Metropolis

métrologie
internet
modélisation
interprétation

LIP6 · CNRS-LAAS · EURECOM Sophia Antipolis · france telecom · get · INRIA · Renater

METROPOLIS

Métrologie Pour L'Internet

Projet RNRT Exploratoire

Livrable
Rapport d'avancement

Coordinateur du projet : Kavé Salamatian (LIP6).

Laboratoire d'Informatique de Paris 6	LIP6
France Télécom Recherche et Développement	FTRD
Groupe des Ecoles de Télécommunications	GET
Groupement d'Intérêt Publique RENATER	RENATER
Institut Eurecom	EURECOM
Institut National de Recherche en Informatique et Automatique	INRIA
Laboratoire d'Analyse et d'Architecture des Systèmes	LAAS

Table des matières

Première partie

Sous Projet 2

Classification et Dimensionnement

METROPOLIS

Métrologie Pour L'Internet

Projet RNRT Exploratoire

Sous Projet 2
Classification et dimensionnement

Livrable
Rapport d'avancement

Coordinateur du sous projet : Fabrice Guillemin (FT R&D).
Contributeurs : FT R&D : Fabrice Guillemin, Philippe Olivier, Thomas Bonald
LIP6 : Kavé Salamatian, Augustin Soule

France Télécom Recherche et Développement FTRD
Laboratoire d'Informatique de Paris 6 LIP6

Chapitre 1

Introduction

L'avènement de l'Internet comme réseau intégrateur de tous les services, de la transmission de données aux services multimédia, en passant par la téléphonie, a bousculé ces dernières années de nombreuses habitudes et certitudes chez les opérateurs de réseaux de télécommunications. Le réseau Internet a été initialement conçu avec des principes simples et parfaitement adaptés au transfert de données dans les réseaux informatiques, éventuellement interconnectés par des liaisons louées. Son déploiement à grande échelle avec une cible grand public ne va pas sans poser de nombreux problèmes pour les opérateurs de réseaux.

Tout d'abord, le protocole IP est par essence même en mode non connecté. L'avantage majeur de ce mode de transfert est qu'il permet de rapatrier de manière très souple des données localisées sur des serveurs différents, souvent géographiquement distants. Le mode non connecté, qui rend toute gestion de trafic à l'intérieur même du réseau très difficile, a comme corollaire le fait que le réseau apparaît comme une boîte noire et que toute l'intelligence est rejetée dans les terminaux. C'est le principe fondamental du protocole TCP qui est capable de transmettre des données dans un environnement parfaitement hostile. Le mode non connecté ne facilitant pas la réservation de ressources suivant un chemin pré-déterminé dans le réseau, les terminaux mettent en œuvre des procédures de récupération de pertes et d'asservissement de débit en fonction de l'état de congestion du réseau. En outre, le réseau IP emploie des paquets de longueur variable, qui peut dépendre de nombreux facteurs : signification sémantique du paquet (un paquet d'acquittement est en général plus petit qu'un paquet de données), le type de raccordement du terminal et certaines fonctionnalités mises en œuvre dans les terminaux comme le mécanisme de MTU path discovery. Une connexion à un réseau local de type Ethernet donne en général naissance à des paquets de 1500 octets. La variabilité de la longueur des paquets rend difficile toute définition d'un débit. Il ne suffit en effet pas de compter le nombre de paquets pour en déduire un nombre de bits par seconde.

Enfin, le réseau Internet dans un système autonome donné n'est en principe pas hiérarchisé. Les paquets des flots de données empruntent des chemins qui sont fixés par l'algorithme de routage (la plupart du temps ISIS dans les réseaux opérationnels), qui calcule des chemins à travers le réseau en utilisant certaines métriques, principalement des poids administratifs. La décision de routage est prise localement par un routeur et il n'y a pas en général de cohérence globale. Chaque routeur construit sa propre représentation du réseau en fonction des informations de routage et choisit paquet par paquet le prochain routeur. En outre, des mécanismes de partage de charge peuvent être activés

pour éviter la surcharge de certains liens. Dans un tel cas, les paquets d'un flot de données peuvent être éparpillés sur plusieurs routes.

Le mode non-connecté, la variabilité de la longueur des paquets, le manque de hiérarchisation rendent difficile toute mise au point de gestion de trafic dans le réseau Internet. Des efforts particuliers ont été cependant faits par l'IETF ces dernières années pour surmonter ce problème. En dehors des premières tentatives liées à l'architecture IntServ, qui a rencontré une certaine hostilité au sein de la communauté Internet à cause de son manque d'extensibilité (scalability), l'IETF a introduit l'architecture DiffServ et le protocole MPLS. DiffServ propose une différenciation dans le traitement des paquets localement à un routeur en prenant en compte un champ dans le l'entête des paquets IP (per hop behavior) sans pour autant assurer une cohérence globale vis à vis de qualité de service. Une telle cohérence nécessiterait des protocoles de contrôle de charge au niveau du réseau, ce qui est loin de faire l'unanimité dans le communauté de l'Internet. Le protocole MPLS repose sur un traitement des paquets sur la base d'un label, qui peut être simplement un préfixe dans les champs d'adresse (source ou destination, voire les deux) ; le routage des paquets se fait sur la base du label et nom pas de l'adresse IP en entier, ce qui permet d'accroître la performance des routeurs, raison invoquée initialement pour l'introduction de MPLS. Aujourd'hui, l'intérêt majeur dans MPLS est que ce protocole peut être utilisé pour gérer la bande passante dans le réseau. En effet, MPLS assure une cohérence globale vis à vis des labels via la mise en œuvre du protocole LDP et rétablit dans une certaine mesure la notion de faisceau dans le réseau : les paquets qui partagent un certain nombre de caractéristiques communes identifiées par un label suivent le même chemin dans le réseau. Des procédures de gestion de trafic (traffic engineering) peuvent être ensuite utilisées pour gérer la bande passante dans le réseau, même si une gestion stricte de la bande passante n'est pas obligatoire par un contrôle des paramètres de trafic. Bien que l'IETF ait consenti des efforts très importants pour doter le réseau Internet de procédures de gestion de trafic, les principaux "gourous" de l'Internet se sont prononcés pour un abandon pur et simple de toutes procédures compliquées de gestion de qualité de service dans les réseaux dorsaux (ou réseaux cœurs). Cette tendance est très marquée dans le milieu universitaire nord-américain aujourd'hui (cf. les recommandations de la NSF) et se retrouve chez de nombreux opérateurs. Pour les principaux concepteurs de l'Internet, la seule planche de salut pour améliorer la qualité de service dans un réseau IP est d'accroître les débits des liens de transmission et de rejeter tous les problèmes de gestion de trafic en périphérie de réseau, quand cela est possible, par exemple quand le réseau d'accès est lui-même doté de fonctions de gestion de trafic. Il faut de plus noter que la mise en œuvre des procédures de gestion de trafic à l'échelle d'un réseau est rendue d'autant plus complexe que les implémentations sont souvent propriétaires ; le style des RFC laisse aux constructeurs toute la latitude pour interpréter à leur guise les normes produites par l'IETF. Il faut néanmoins nuancer le propos pour un opérateur global pour lequel la mise en œuvre de MPLS est tout à fait envisageable pour des offres à des entreprises. Partant de ces constatations, la maîtrise totale d'un réseau IP par un opérateur, comme par exemple dans le cas d'un réseau téléphonique, semble très difficile. L'absence de signalisation, le mode non connecté, les protocoles de routage comme BGP, etc. rendent le trafic extrêmement aléatoire et volatile dans un réseau. En réalité, la seule approche possible pour connaître de manière fine le trafic transitant réellement dans un réseau, ceci afin de qualifier la qualité de service perçue par les clients (sans attendre le retour des clients eux-mêmes) et les usages, est d'effectuer des mesures. Mais pour réaliser de telles mesures, il faut mettre au point une méthode de mesure. C'est le sujet même de la métrologie.

Chapitre 2

Classification des flots

2.1 Introduction

L'Internet est constitué de milliers de liens, au travers desquels des millions d'utilisateurs partagent l'information. Les utilisateurs de ce gigantesque réseau génèrent des paquets qui utilisent des protocoles divers et variés comme TCP, UDP, HTTP, *etc.*, pour arriver à destination. Tout ces paquets sont mélangés au travers des routeurs pour former des flots de plus en plus agrégés.

L'ingénierie de trafic IP à pour but de gérer ces flots agrégés, de manière à ce qu'ils traversent sans encombre le réseau, *i.e.* de manière efficace, sans erreurs et le plus rapidement possible. Un nombre important de contributions de la communauté scientifique ont portés sur ces objectifs. Typiquement les solutions proposées ont été axées sur une caractérisation du trafic que l'on désire gérer afin d'assurer une meilleure qualité de service, *i.e.* faible taux de perte, faible congestion. Parmi ces approches, nous pouvons citer les algorithmes d'équilibrage de charge, la mise en forme du trafic...

Mais tous ces algorithmes nécessitent au préalable une caractérisation des flots ainsi qu'une connaissance de leur comportement. Avant toute chose, il convient de préciser la notion de flot, qui peut paraître floue. Suivant le niveau d'agrégation et le point de vue dans lequel nous nous plaçons, la définition d'un flot peut être différente ; par exemple pour un pare-feu, une simple connexion TCP est un flot. Tandis que pour un site web, une session HTTP peut être considérée comme un flot. A chaque niveaux d'agrégation, les caractéristiques propres des entités des niveaux inférieurs (dépendantes de l'adresse source, du chemin parcouru, *etc.*) sont brassées et le flot présente ainsi un comportement macroscopique qui n'est pas facilement mis en relation avec les comportements microscopiques sous-jacents .

Plutôt que d'essayer d'analyser chaque flot, il est intéressant d'essayer de classer les flots en un petit nombre de classes homogènes, pour ensuite caractériser chaque classe. L'objectif de chapitre consiste à proposer une méthodologie générique de classification du trafic hautement agrégé dans les dorsales de l'Internet. Nos motivations sont multiples. Tout d'abord, notre méthode doit être fondée sur le moins d'hypothèses possibles afin de capturer le plus de comportements possibles. Les approches précédentes reposaient généralement sur les deux premiers moments de la distribution. Ici nous utilisons des histogrammes qui contiennent beaucoup plus d'informations. Ensuite, nous devons être capables de détecter et d'identifier de manière fiable les flots de chaque classe. Ceci

permet alors d'appliquer un traitement particulier aux flots de chaque classe en leurs appliquant un équilibrage de charge, un changement de route ou n'importe quelle opération de gestion de trafic.

La granularité qui apparaît naturelle pour l'ingénierie du trafic est celle qui apparaît dans les tables de routage. Étant donné que le routage dans le cœur du réseaux est fondé sur les préfixes IP annoncés par le protocole BGP (*Border Gateway Protocol* [36], nous adopterons tout au long de ce travail la définition d'un flot comme l'ensemble des paquets allant d'un préfixe BGP vers un autre. Cette granularité semble raisonnable car c'est la plus petite entité manipulable par un routeur de cœur, et donc le moindre changement dans la politique de routage touche un flot dans son intégralité. Cette définition de flot est hautement agrégée et correspond dans la plupart des cas à un ensemble de connexions TCP ou de flux UDP. Ces flots peuvent être mesurés simplement à l'aide de l'outil NETFLOW[9] de Cisco ou par le biais de mesure passives sur le réseau.

Une classification fondée sur ce niveau de granularité de flot est intéressante pour l'ingénierie de trafic, car elle applique le principe de '*diviser pour régner*'. En effet cette classification permet de diviser les flots en plusieurs classes homogènes, ce qui simplifie grandement l'ingénierie de trafic car il suffit de ne traiter qu'un petit nombre d'entités pour résoudre la plupart des problèmes rencontrés dans le réseau. L'exemple le plus courant de ce type de classification est la séparation des flots en éléphants et en souris. Les études précédentes [12, 16, 28, 35] ont montrées que seul un petit pourcentage des flots était responsable de la majeure partie de l'encombrement d'un lien. Ce type de comportement à été observé à tout les niveaux d'aggrégation ; aussi bien au niveau des flots TCP, entre Systèmes Autonomes, ou encore au niveau des flots applicatifs. Une bonne connaissance de ce phénomène permet, en traitant uniquement quelques éléphants, de résoudre la majeur partie des problèmes à l'intérieur du réseau.

D'autre part, cette approche permet la détection des changements survenu dans le réseau, qu'ils soient imputable à un problème physique ou encore à une attaque de type déni de service distribué, en suivant les changements de classes des flots.

Notre recherche a pour objectif d'appliquer des techniques d'inférences statistiques pour identifier les classes de comportement différents ainsi que les flots appartenant à chaque classe. Nous souhaitons tout d'abord trouver une représentation statistique sémantiquement forte ; la classification doit nous permettre de caractériser et de mieux comprendre les caractéristiques d'un ensemble de flots. Nous espérons ensuite aboutir à des classes de comportement stables au cours du temps, c'est à dire des classes qui présenterait des caractéristiques identiques sur une période de temps suffisamment longue (du moins au vu de la période de mesure). Enfin nous souhaiterions que la stabilité soit effective pour chaque flots, c'est à dire que les flots qui sont assignés à une classe restent dans cette classe pour une durée suffisamment longue.

Les diverses publications de ces dernières années ont eu fréquemment recours à des termes issus de la zoologie, comme éléphants, souris, tortues ou encore dragon, pour identifier le comportement des flots. Nous ne dérogerons pas à cette tradition et notre recherche s'apparentera donc à un safari et à une traque où notre gibecière contiendrait différentes espèces de flots ayant des comportements particuliers.

Notre approche est la suivante : tout d'abord nous collectons grâce à une plate-forme de mesure passive des traces de paquets. Ces traces contiennent l'entête IP de tous les paquets qui traversent un lien OC-12 de la dorsale du réseau d'un opérateur commercial américain. Nous calculons ensuite sur une durée de 5 minutes la valeur moyenne de la bande passante de chaque flot BGP. Ces valeurs sont ensuite transformées en histogrammes. La classification est effectuée sur l'ensemble de ces his-

togrammes normalisés. Chacun des histogrammes peut être considéré comme une observation issue d'une distribution aléatoire et tous les membres d'une même classe sont considérés comme issus d'une même distribution aléatoire. Nous utilisons ici une distribution de Dirichlet pour modéliser la distribution aléatoire. En effet, cette distribution est très générale et elle est capable de représenter une grande variété de comportements. Ainsi nous supposons que les histogrammes générés par les flots dans le réseau suivent un mélange de distribution de Dirichlet. Les paramètres de ce mélange ainsi que les paramètres de chaque distribution de Dirichlet du mélange sont obtenus par le biais d'une version stochastique de l'algorithme EM (*Expectation Maximisation*). L'algorithme de classification génère aussi les probabilités *a posteriori*, c'est à dire la probabilité pour chaque flot d'appartenir à chacune des classes. Nous utilisons ensuite un critère de type MAP (*Maximum a Posteriori*) pour assigner chaque flot à la classe qui a la plus grande probabilité de le représenter.

L'approche proposée ici est plus générale que celles qui ont été proposées précédemment [27, 15]. Dans ces approches précédentes un seuil est calculé et chacun des flots est classé suivant qu'il est au dessus ou au dessous de cette valeur. Dans notre approche, un éléphant peut avoir des courts moments où il passe sous ce seuil sans pour cela qu'il change de catégorie. Dans [27], les auteurs ont été confrontés à des oscillations des flots entre les classes. Notre approche échappe à ces oscillations en incorporant l'histogramme en entier, plutôt que les valeurs discrètes.

2.2 Méthode

2.2.1 Caractérisation utilisant des histogrammes

Les études précédentes ont tenté de caractériser les flots à l'aide de la moyenne temporelle [26]. Néanmoins, cette approche ne permet pas de caractériser le comportement du flot. Dans ce travail, nous utilisons la distribution empirique comme critère de classification et comme caractéristique comportementale du flot. En effet, l'histogramme empirique garde toute l'information statistique si le nombre de bin est suffisamment large (voir figure 2.1).

Il convient d'introduire quelques notations qui seront utilisées par la suite. Soit $X_i(n)$ la bande passante moyenne utilisée par un flot vers le préfixe i au cours du n^{i}me instant de mesure. Soit $\mathfrak{B} = \{[b_0 = 0, b_1), [b_1, b_2), \ldots, [b_{l-1}, b_l = +\infty)\}$, avec $0 < b_1 < \ldots < b_{l-1}$ un ensemble de l intervalles et $\mathcal{F}^m(X_i; \mathfrak{B})$ l'histogramme empirique de la bande passante X_i sur l'ensemble des intervalles $\mathfrak{B}$ obtenus après m instants de temps. Ainsi $\mathcal{F}^m(X_i; \mathfrak{B})$ est une séquence de l valeurs $(f_{i1}, \ldots, f_{il})$ où f_{ik} est la proportion de temps pendant lequel la valeur X_i du flot i a été observée dans l'intervalle $[b_{k-1}, b_k)$.

2.2.2 Théorie des distributions aléatoires

Dans notre approche, chaque flot est un histogramme normalisé. Chaque histogramme peut alors être vu comme une réalisation d'une source particulière stochastique générant des histogrammes. Cette source peut être définie de manière formelle.

Définition : Une distribution aléatoire (DA) est une application définie sur $(\Omega, \mathcal{F}, \mathbf{P})$ prenant ses valeurs dans $\mathbf{P}(V)$ qui représente l'ensemble des mesures définies dans un sous-ensemble fini et

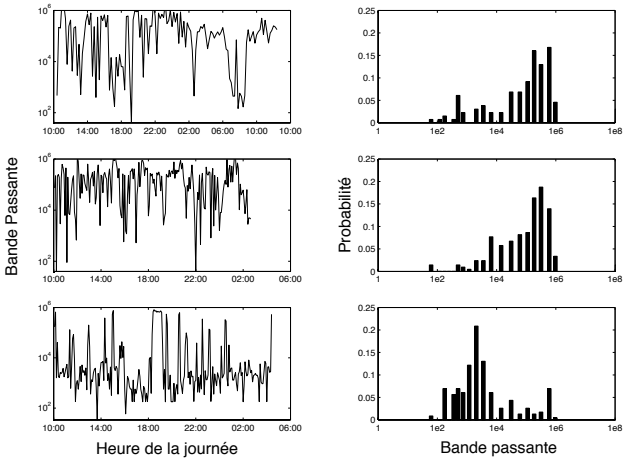


FIG. 2.1 – Variations temporelles d'un flot

mesurable de $(V, \mathcal{V})$. Si $X : \Omega \longrightarrow \mathbf{P}(V)$ est une Distribution Aléatoire alors sa distribution $\mathcal{P}_X$ est une mesure de probabilité sur $\mathbf{P}(V)$.

La notion de distribution aléatoire est intéressante quand elle traite de la distribution d'éléments qui sont eux-même des distributions de probabilité. Dans ce cadre chaque observation est une fonction de densité de probabilité et non plus un vecteur de réels. C'est exactement notre cas car nous étudions des histogrammes normalisés et non plus des suites de valeurs.

Les distributions aléatoires peuvent être obtenues aisément en choisissant les paramètres d'une distribution paramétrique de façon aléatoires. Par exemple, la distribution aléatoire $\lambda(\omega)e^{-\lambda(\omega)x}$, où $\lambda(\omega)$ est lui-même une variable aléatoire, définit une distribution aléatoire exponentielle. La distribution de Dirichlet est souvent utilisée dans le contexte des Distributions Aléatoires.

Distribution de Dirichlet $\mathcal{D}(\alpha_1, ..., \alpha_l)$

Soit $\alpha = (\alpha_1, ..., \alpha_l)$, avec $\alpha_1 > 0, ..., \alpha_l > 0$, et soit $X = (X_1, ..., X_l)$ une distribution aléatoire. La distribution de Dirichlet $\mathcal{D}(\alpha_1, ..., \alpha_l)$ est définie sur l'espace $\mathbf{P}(V)$ de toutes les distributions aléatoires.

$$\mathcal{D}(\alpha_1, ..., \alpha_l) = \frac{\Gamma(\alpha_1 + ... + \alpha_l)}{\Gamma(\alpha_1)...\Gamma(\alpha_l)} X_1^{\alpha_1 - 1} ... X_{l-1}^{\alpha_{l-1} - 1} (1 - \sum_{h=1,...,l-1} X_h)^{\alpha_l - 1} I_{S_l}(X) \qquad (2.1)$$

où S_l représente le simplex :

$$S_l = \{y = (y_1, ..., y_{l-1}), y_j \geq 0, \sum_{j=1}^{l-1} y_j \leq 1\}. \qquad (2.2)$$

La distribution de Dirichlet définit la probabilité d'observer la distribution $\mathcal{F} = (f_1, \ldots, f_l)$ comme le résultat d'un choix aléatoire. Cette distribution est très souple et peut être utilisée pour modéliser un grand nombre de processus stochastique. Un processus stochastiques, ayant une distribution de Dirichlet, est appelé un processus de Dirichlet. Ces processus sont souvent utilisés dans la littérature statistique [30, 13, 14].

Une distribution de Dirichlet a la sympathique propriété suivante : soit $\mathcal{F} = (f_1, \ldots, f_l)$ un processus de Dirichlet suivant la distribution de Dirichlet $\mathcal{D}(\alpha_1, \ldots, \alpha_l)$, *i.e.* $(f_1, \ldots, f_l) \sim \mathcal{D}(\alpha_1, \ldots, \alpha_l)$. Alors la loi marginale de chaque composante f_i est une distribution bêta.

$$f_i \sim \mathcal{B}(\alpha_i, A - \alpha_i)$$

où $A = \sum_{j=1}^{l} \alpha_j$ est le paramètre de dispersion et la valeur moyenne de chaque composante est $E\{f_i\} = \frac{\alpha_i}{\sum_{j=1}^{l} \alpha_j}$. Pour rappel, la distribution bêta $\mathcal{B}(\alpha, \beta)$ est définie par :

$$\mathcal{B}(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}(1 - x)^{\beta - 1} x^{\alpha - 1}$$

La valeur moyenne du processus de Dirichlet est alors définie par : $E\{\mathcal{F}\} = (E\{f_1\}, \ldots, E\{f_l\})$. Le paramètre de dispersion A contrôle la dispersion du processus de Dirichlet autour de sa moyenne.

Les distributions de Dirichlet peuvent être construites par normalisation de distribution gamma. Soient $Z_1, ..., Z_l$, l distributions aléatoires indépendantes, à valeurs réelles, suivant une distribution gamma $\gamma(\alpha_1, 1), ..., \gamma(\alpha_l, 1)$ respectivement, où

$$\gamma(a, b)(x) = \frac{1}{\Gamma(a)} b^a e^{-bx} x^{a-1} I_{(x>0)}.$$

Dans ce cas la distribution du vecteur aléatoire $(\frac{Z_1}{Z}, ..., \frac{Z_l}{Z})$ où $Z = Z_1 + ... + Z_l$ suit une distribution de Dirichlet $\mathcal{D}(\alpha_1, ..., \alpha_l)$. Il faut néanmoins noter que même si la distribution de Dirichlet peut être obtenue d'une telle façon, l'utilisation de telles distributions ne signifie en aucun cas que la distribution est issue de la normalisation de variables issues de loi Gamma.

Mélange fini de processus de Dirichlet

Dans la suite, nous utilisons des mélanges de processus de Dirichlet comme modèle pour les histogrammes normalisés de la bande passante des flots traversant le réseau. L'utilisation de ce type de mélange est justifiée dans [30]. En effet, les mélanges de loi de Dirichlet sont souvent utilisés comme modèles dans le cadre de l'estimation de distribution bayesienne avec *a priori* non paramétriques où l'approche classique de choix de distribution *a priori* impose des contraintes beaucoup trop rigides. Quelques exemples d'applications comme le problème empirique de Bayes [13], la régression non paramétrique [21] ou encore l'estimation d'une fonction de densité [14] montrent la puissance d'une telle approche.

Une des raisons principales de l'utilisation des mélanges de processus de Dirichlet est liée au fait que les mélanges de processus de Dirichlet contiennent un unique paramètre de dispersion qui contrôle la dispersion de la fonction de densité de probabilité autour de sa moyenne. Cette moyenne est elle-même une fonction de probabilité qui peut être modélisée par un modèle paramétrique. En

pratique, si une fonction de densité paramétrique moyenne est spécifiée alors le paramètre de dispersion du mélange de processus de Dirichlet contrôle la divergence maximale autorisée par rapport au comportement moyen. Ainsi le modèle par mélange de processus de Dirichlet permet une flexibilité importante et un *a priori* plus faible.

Dans ce contexte, supposons que les histogrammes normalisés observés suivent une distribution $\mathcal{P}(f_1, \ldots, f_l)$. Alors pour un mélange de K distributions de Dirichlet, nous avons :

$$\mathcal{P}(f_1, \ldots, f_l) = \sum_{k=1,\ldots,K} p_k \mathcal{D}(\alpha_1^k, ..., \alpha_l^k)$$

Les paramètres à estimer sont donc $p_1, \ldots, p_K$ et $\alpha_1^k, ..., \alpha_l^k$ pour $k = 1, \ldots, K$.

2.2.3 Méthode d'estimation

Avant de présenter la méthode d'estimation des paramètres du mélange, nous rappelons l'estimation des paramètres du mélange de loi dans le cas où les observations sont des vecteur de réels. Soit $x_1, x_2, ..., x_n \in \mathbb{R}^p$, n observations d'un échantillon de n vecteurs aléatoires de dimension p. Le problème consiste à estimer la distribution $\mathcal{P}_X$ de X quand $\mathcal{P}_X$ est supposée être une combinaison linéaire de distribution appartenant à une famille quelconque de fonctions paramétriques, comme par exemple la famille exponentielle.

$$\mathcal{P}_X = \sum_{k=1,\ldots,K} p_k P_k$$

Plusieurs méthodes ont été proposées pour estimer les poids du mélange p_k et les paramètres de chacune des composantes P_k. Nous utilisons ici une des méthodes parmi les plus efficaces, le *Simulated Annealing Expectation Maximisation (SAEM)*[8], qui est très robuste face aux problème des minima locaux.

L'algorithme EM est une méthode générale pour trouver le maximum de la fonction de vraisemblance d'une distribution fixée quand les données sont incomplètes ou que des paramètres ne sont pas directement mesurables. La méthode EM repose sur une itération de deux étapes, l'Estimation, la Maximisation. Les détails de l'application de l'algorithme EM dans le cas des mélanges est décrite dans [4].

L'algorithme SAEM, introduit par Celeux et Dielbot dans [8], est une modification de l'algorithme EM permettant d'éviter les problèmes de convergence lente et de minima locaux. Dans cette approche plutôt que d'utiliser une distribution *a priori* pour le paramètre inconnu, une étape de simulation stochastique est ajoutée. Cette étape génère les données manquantes pour se retrouver dans le cas de données complètes sans variables cachées. Une suite $\{\gamma_q\}$ de réels positifs décroissant vers zéro permet de réguler l'importance de l'étape Stochastique. Cela permet de passer en douceur d'un pur algorithme Stochatic EM au début, à un pur EM à la fin des itérations.

SAEM pour des mélanges de Dirichlet

Une formulation de l'algorithme SAEM à été écrite dans le cas des mélanges où la fonction de densité appartient à la famille exponentielle, *ie* de la forme :

$$d(x, a) = d(a)e(x) \exp < a^T . b(x) >$$

où le paramètre a est un vecteur de transposé a^T, $d(a)$ est un facteur de normalisation, e et b sont des fonctions connues mais arbitraires et $<>$ est le produit scalaire standard. Dans le cas de Dirichlet, nous avons $a = (1 - \alpha)$ et $b(x) = \log(x)$. Les observations sont les N vecteurs $f_i, i = 1, ..., N$ où chaque observation représente un histogramme normalisé. Le nombre de composantes du mélange K est supposé connu. Soit $\{\gamma_q\}$ une suite de réels positifs, strictement décroissant avec $\gamma_0 = 1$.. Soit $c(n)$ une valeur seuil telle que $0 < c(n) < 1$ et $\lim_{n\to\infty} c(n) = 0$.

Etape d'Initialisation :
Assigner aléatoirement chaque f_i à une classe.
Etape de Simulation :
Générer aléatoirement $t_{ik}^{(0)}$ $(i = 1, \ldots, n)$ représentant la probabilité *a posteriori* initiale qu'un flot f_i appartienne à la classe $k = 1, \ldots, K$.
pour $q = 0$ *à* Q **faire**
Etape Stochastique :
générer un vecteur multinomial $e_{qi} = (e_{qi}^k)$ suivant la distribution de probabilité $\{t_{i1k}^q\}$ où tout les e_{qi}^k sont 0 sauf un seul égal à 1.
si $\frac{\sum_{i=1,\ldots,N} e_{qi}^k}{N} < c(n)$*pour un* k *quelconque* **alors**
Revenir à l'étape d'initialisation.
fin
Etape de Maximisation :
Estimer les poids du mélange $p_{(q+1)k} = \frac{1}{n}[(1 - \gamma_q) \sum_{i=1,\ldots,n} t_{ik}^q + \gamma_q \sum_{i=1,\ldots,n} e_{qi}^k]$.
et les paramètres $a_k^{q+1} = (1 - \gamma_q) \frac{\sum_{i=1,\ldots,n} t_{ik}^q b(f_i)}{\sum_{i=1,\ldots,n} t_{ik}^q} + \gamma_q \frac{\sum_{i=1,\ldots,n} e_{qi}^k b(f_i)}{\sum_{i=1,\ldots,n} e_{qi}^k}$
Etape d'Estimation :
mettre à jour les probabilités *a posteriori* qu'un flot i appartienne à la classe k : (t_{ik}^{q+1})
avec $t_{ik}^{q+1} = \frac{p_{(q+1)k} h_{(q+1)k}(f_i)}{\sum_{r=1\ldots K} p_{(q+1)r} h_{(q+1)r}(f_i)}$
fin

Algorithm 1 – Algorithme SAEM pour l'estimation des paramètres d'un mélange de processus de Dirichlet

Cet algorithme contient trois étapes principales :
– Une étape de Simulation, qui ajoute du bruit au système en assignant les classes de manière probabiliste. C'est ce bruit qui aide l'algorithme à sortir des minima locaux. A mesure que γ_q décroît, le bruit diminue est l'estimation se stabilise.
– Une étape de Maximisation qui met à jour les paramètres a_k^{q+1} et $p_{(q+1)k}$ en maximisant la vraisemblance.
– Une étape d'Estimation qui actualise la probabilité *a posteriori* t_{ik}^q que le flot i appartienne à la classe C^k à l'itération q de l'algorithme.

Cet algorithme estime de manière asymptotique le mélange car p_{qk}, t_{ik}^q et la densité convergent quand $q \to \infty$.

Une propriété intéressante de cet algorithme est qu'il donne une probabilité *a posteriori* t_{ik} d'appartenance à une classe qui permet de nuancer l'appartenance à une classe. Dans la partie suivante, nous allons confronter notre nouvelle méthode de classification à des données de synthèse puis à des données réelles issues d'une dorsale de l'Internet.

2.3 Applications et résultats

Dans cette partie nous allons commencer par appliquer notre algorithme sur des données de synthèse pour illustrer la capacité de notre algorithme à distinguer précisément les différents comportements. Nous appliquerons ensuite la méthode présentée sur des données réelles provenant de la capture d'une dorsale OC-12.

Notre objectif dans cette section n'est définitivement pas de présenter des résultats expliquant comment effectuer la classification dans un contexte réaliste dans l'Internet, mais plutôt une illustration de la méthode proposée avec des exemples concrets. Nous sommes pleinement conscients que les résultats présentés ici ne sont pas suffisants pour donner une description comportementale détaillée des flots. L'application de cette méthode sur des traces beaucoup plus variées est en cours de réalisation et en dehors de l'étendu de ce livrable. Malgré cela, les résultats obtenus sur une trace de 3 jours sont suffisamment intéressants pour illustrer la puissance de cette méthode.

2.3.1 Données utilisées pour la classification

Les données utilisées dans ce papier proviennent de traces collectées sur un lien au cœur d'un grand opérateur de réseau (Sprint). Des équipements de mesure passive ont été utilisés pour capturer les 44 premiers octets de chaque paquet IP traversant les liens. Pour cette étude nous avons utilisé des traces provenant de deux liens OC-12 situés aux Etats Unis, un sur la côte Est, l'autre sur la côte Ouest. La capture a commencé le 24 juillet 2001 et a duré 3 jours et demi. Ces liens sont à deux sauts de la bordure de l'internet ce qui permet d'atteindre un degré d'aggrégation suffisant pour l'étude que nous souhaitons faire. La collecte de paquets s'est accompagnée de la capture de tables BGP. Ces tables sont considérées sans défaut et représentent environ 120 000 entrées. Nous avons alors relevé le volume de trafic entre chaque préfixe BGP et ainsi calculé la bande passante moyenne utilisée sur des intervalles de 5 minutes. Au cours de cette campagne de mesure, nous avons pu remarquer qu'environ 90% des préfixes BGP ne recevaient pas de trafic. Nous définissons alors un flot comme *actif* s'il transmet au moins 1 paquet durant la période de mesure. Nous avons trouvé qu'environ 2000 flots étaient actifs durant la période de mesure.

2.3.2 Transformation en histogrammes et taille des bins

La première étape de l'algorithme est de transformer les flots en un ensemble d'histogrammes normalisés. Pour cela, nous avons partagé nos 3 journées de traces en 3 parties et effectué la classification sur une journée[1]. Chaque période de 24 heures générant 288 mesures de la bande passante (étant donné que nous avons un intervalle de 5 minutes entre chaque mesure) pour chaque flot. Ces 288 valeurs sont alors regroupées en 20 bins. Un des point dur de la transformation est le choix du nombre des bins et aussi celui du centre des bins $\mathfrak{B}$. S'ils sont mal choisis, quelques-uns sont vides et empêchent l'algorithme de converger. Pour éviter ce problème, nous avons choisi de trouver des centres qui permettent d'avoir des bins contenant à peu près autant d'instants de mesure chacun. Dans le cas de 20 bins, cette approche nous donne des centres distribués de manière quasi logarithmique.

[1] Ceci principalement pour examiner le comportement de l'algorithme sur plusieurs jours consécutifs

En utilisant ces bins, nous transformons chacun des flots en histogramme normalisé. L'objectif suivant étant de trouver K classes qui représentent le mieux possible l'ensemble de ces histogrammes. K définit alors le nombre de processus de Dirichlet que nous allons utiliser par la suite.

2.3.3 Validation à partir de données de synthèse

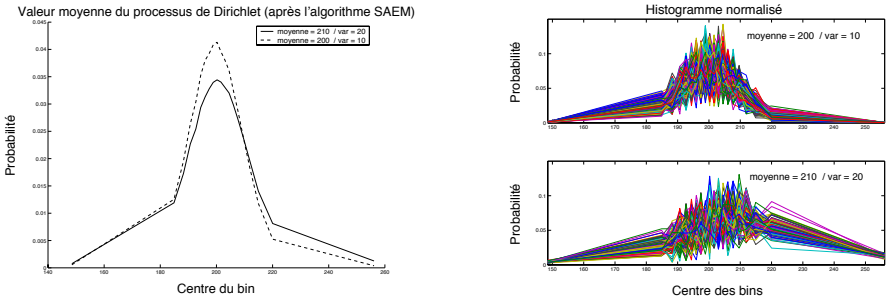


FIG. 2.2 – Classification sur des données synthétiques

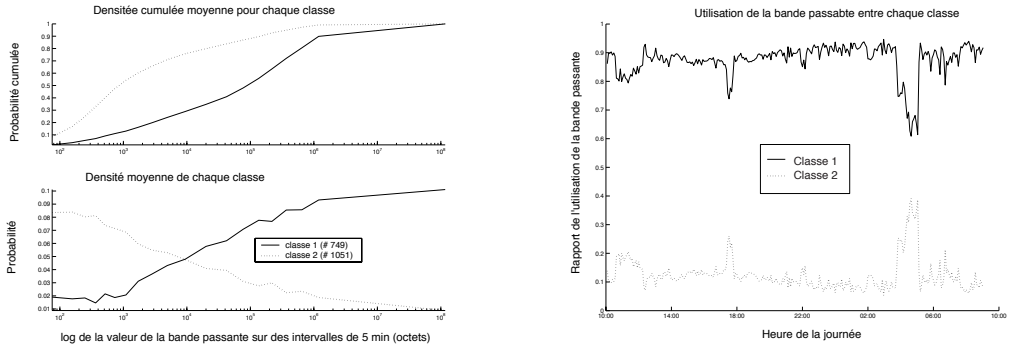
En classant des données réelles, nous obtenons des informations sur les données, mais cela ne nous donne aucune information sur la méthode de classification car nous ne savons pas ce que nous devrions trouver.

Ainsi, nous avons testé des données spécialement créées pour l'occasion. Nous avons généré deux classes. Chacune suivant une distribution normale de paramètres différents ($\mu_1 = 200$, $\sigma_1 = 10$ et $\mu_2 = 210$, $\sigma_2 = 20$). Nous avons alors généré 500 flots de la première catégorie, et 100 de la seconde pour un total de 600 flots. Nous avons délibérément choisi deux classes dont la moyenne est proche car cela crée un cas relativement dur et notre méthode doit être capable de distinguer deux classes quasiment similaires.

Nous avons ensuite appliqué l'algorithme sur les histogrammes normalisés comme décris précédemment. Les résultats sont présentés dans la figure 2.2. Tout les flots ont été classés avec succès.

2.3.4 Classification avec deux classes

Nous présentons maintenant le résultat de classification en deux classes. La figure 2.3-a montre la distribution (cumulée et non cumulée) de la moyenne de chaque classe. Nous avons 749 flots (41%) dans la première classe et 1051 (59%) dans la seconde. Dans la seconde classe la plupart des flots ont une grande probabilité d'avoir de petites valeurs. Le comportement moyen de la classe 2 ressemble à un comportement exponentiel mis en évidence par l'alignement sur une droite entre 100 octets/s et 1 Moctets/s. Dans la première classe, la majorité des flots ont une grande probabilité d'avoir de grandes valeurs et n'ont quasiment jamais des valeurs nulles. Cette classification empirique correspond au comportement classique des éléphants et des souris. Dans la figure 2.3-b, on peut voir que les flots de la classe 1 contribuent à quasiment 90% de la bande passante tout au long de la journée. Nous



(a) (b)

FIG. 2.3 – (a) Ditribution/Distribution cumulé moyennes dans le cas de deux classes (b) Contribution de chaque classe au trafic total durant 24h.

appellerons donc la première classe des éléphants et la seconde des souris. Bien que nous ayons forcé l'utilisation de deux classes, la méthode présenté à permis de créer deux classes distinctes et ne s'est pas laissée tromper par des flots mi-éléphant mi-souris.

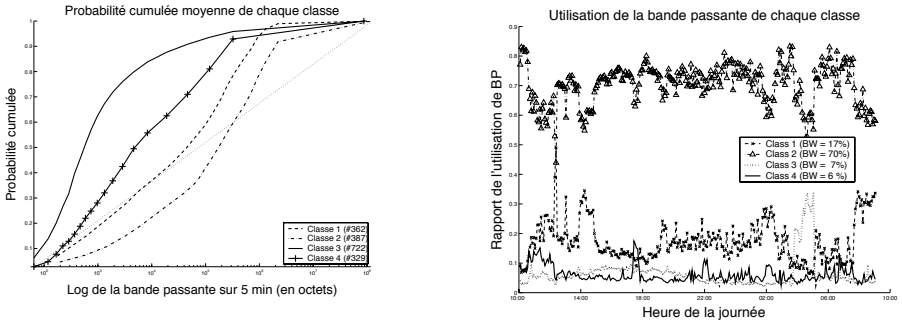
Il faut noter qu'il reste dans la catégorie des éléphants des flots qui ont une faible valeur moyenne. Ces flots peuvent avoir une faible valeur moyenne, mais aussi avoir quelques grosses rafales. Le phénomène inverse s'applique aux souris. Nous avons ainsi trouver un moyen d'isoler (chasser !) les éléphants et les souris.

Nous pouvons aussi observer un comportement étrange autour de 3h du matin dans la figure 2.3-b. Nous ne pouvons pas l'expliquer. Pour cela, nous allons essayer de classer les flots à l'aide de plus de classes.

2.3.5 Classification avec plus de classes

Nous classifions à présent en utilisant 4 classes. Le comportement moyen est montré dans la figure 2.4. Cette fois-ci, nous avons montré le comportement des fonctions de probabilité cumulatives moyennes des classes (fig. 2.4-a). Nous avons aussi ajouté une ligne pointillée pour permettre de comparer les valeurs plus aisément. Les classes 1 et 2 correspondent maintenant aux éléphants dans le cas de deux classes tandis que les classes 3 et 4 sont les souris telles que précédemment définies. Les classes 3 et 4 croissent rapidement vers 1 (fig. 2.4-a) ne donnant pas de signes d'une quelconque queue lourde de distribution. La classe 1 quand à elle est linéaire sur une grande partie de la courbe, montrant un comportement de queue lourde de distribution.

En regardant la figure 2.4-b, nous pouvons voir que la classe 1 contient 20% des flots tandis qu'elle est responsable de plus d'environ 70% de la bande passante. Les classes 3 et 4 contiennent 58% des flots pour seulement 13% du trafic total. Si l'on regarde précisément la classe 3, elle semble responsable du comportement étrange vers 3h du matin tandis que la classe 4 n'y contribue pas. La



(a) (b)

FIG. 2.4 – Résultat pour 4 classes : (a) PDF moyenne de chaque classe (b) Contribution de chaque classe au trafic total

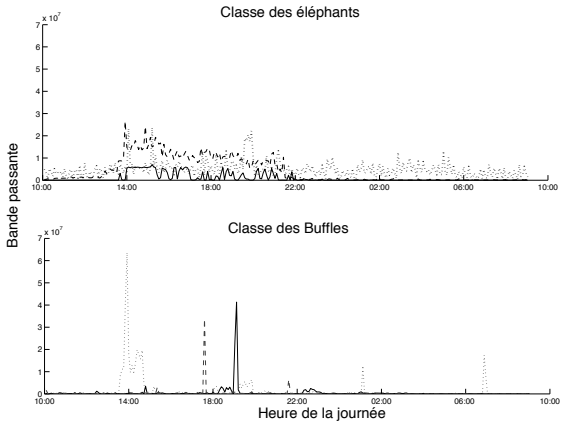
méthode que nous proposons ici nous permet de détecter des comportements étranges. Il est fort probable que la classe 3 contienne les flots qui font des opération de maintenance ou encore des sauvegardes la nuit.

Nous pouvons aussi faire une différence entre les classe 1 et 2, chacune contribuant différemment au trafic total sur le lien. En regardant de plus près les variations temporelles de chacune des classes, nous avons trouvé que les flots de la classe 2 sont beaucoup moins constants que la classe 1 (voir fig 2.5 pour un exemple de flots de la classe 1 et 2).

Nous avons donc 4 classes de flots qui exhibent chacune des comportements qui les distinguent, nous nous permettons de nommer ces «espèces», la classe 1 représentant les éléphants, la classe 2 les buffles, la classe 3 les souris et enfin la classe 4 les libellules.

Notre classification est capable de faire une distinction fine entre les flots car elle prend en compte tout l'histogramme des valeur des flots et qu'elle utilise un modèle généraliste. Tous les jours de toutes les traces que nous avons à notre disposition mènent à des comportements similaires, mais nous sommes bien conscients que de tels résultats doivent être étudiés à l'aide de plus de traces, pour permettre de valider cette classification.

Un point important qui n'a pas été discuté car il est en dehors du sujet de l'article concerne le mode de décision du nombre de classes nécessaires pour décrire le lien observé. La réponse est relativement simple : clairement, plus le nombre de classes est important, meilleure est la description. Le degré de précision nécessaire dépend clairement de l'application des résultats de la classification. Néanmoins quelques critères plus objectifs peuvent être appliqués. Une approche prometteuse est celle basée sur l'entropie. Cette approche est détaillée dans [31], il y est exposé que le logarithme de la fonction de vraisemblance est bornée par l'entropie de la source générant les observations. Au cours de la section suivante, nous allons regarder le problème important de la stabilité des classes au cours du temps.



(a)

FIG. 2.5 – Comportement typique sur 24 heures des Élephants et des Buffles

2.4 Analyse de stabilité

Un des problèmes classiques de la classification de flots est celui de la stabilité de la classification. Une classification effectuée à différentes heures de la journée classifie souvent les flots dans des classes différentes, avec une oscillation des flots entre les classes à des instantes différents [26].

Nous avons essayé d'évaluer la sensibilité de notre approche à ce phénomène d'oscillation. Pour cela nous avons utilisé 4 classes, sur trois jours consécutifs. Les classes moyennes sont représentées sur la figure 2.6 (Un graphique par classe).

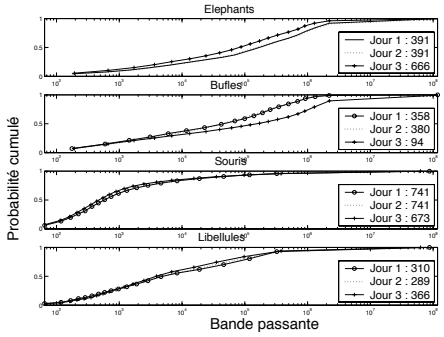


FIG. 2.6 – PDF moyenne des classes sur trois jours différents.

Nous pouvons voir que les classes 3 et 1 sont très stables et leur moyenne ne changent quasiment pas au cours du temps. Pour la classe 2 (les buffles), nous pouvons voir qu'ils sont très stables les deux premier jours, mais qu'ils disparaissent le 3^e jour (ils passent de 380 à 94). Cette extinction reflète un comportement hebdomadaire car le troisième jour est un samedi. Si l'on regarde les changements, entre le premier et le second jour, on peut voir que seulement 90 flots changent de classe, ce qui est très peu comparé au 1800 flots au total. Ce graphique semble montrer que l'algorithme est relativement stable. Bien sûr, il faudrait des périodes beaucoup plus longues pour l'affirmer.

2.5 Spécificité du trafic ADSL

Alors que la majorité des résultats publiés jusqu'à présent dans la littérature technique concernent essentiellement les réseaux locaux ou des réseaux cœurs peu chargés (cf. les sections précédentes), on montre dans cette section les spécificités du trafic Internet commercial, par l'intermédiaire de l'analyse du trafic ADSL transitant sur des liens avec des niveaux de charge non négligeables. Le point le plus marquant est certainement l'importance du trafic peer-to-peer (p2p), qui représente une majorité écrasante du trafic. Ce trafic présente des particularités très importantes qui nécessitent une analyse poussée en termes de classification de flux. Il se caractérise par des transferts de volume très importants et par des débits par connexion TCP assez modestes.

Pour illustrer le propos, on analyse une trace de trafic sur un lien du backbone IP de France Telecom, à savoir un lien à 1Gbit/s entre un routeur du réseau IP et un commutateur Gigabit Ethernet desservant plusieurs plaques ADSL ; les traces de trafic sont capturées avec les sondes NetVCR. Seul le trafic TCP est considéré dans la suite de ce document. Les résultats présentés ci-après sont relatifs à une capture de trafic effectuée en Décembre 2002 entre 13h27 et 14h51, ce qui correspond à une activité modérée de la part des clients ADSL. On montrera par la suite sur des captures effectuées en novembre 2003 que si les valeurs numériques changent, les conclusions qualitatives sur le trafic restent les mêmes. Dans toute la suite, les débits sont évalués sur des intervalles de temps de $\Delta = 100$ ms.

Le but de cette section est d'introduire une zoologie relativement simple et une méthode d'aggrégation pour regrouper plusieurs entités élémentaires afin de constituer de nouvelles entités qui ont une interprétation plus intuitive et qui possèdent de bonnes propriétés pour une modélisation ultérieure (cf. le livrable du Sous-projet 5).

2.5.1 Premières observations

La répartition du volume de données en fonction des applications est fournie par le Tableau 2.1. Seul le trafic peer-to-peer (p2p) observable via les numéros de port est indiqué dans cette figure. De nombreux protocoles p2p utilisent aujourd'hui des ports dynamiques qu'il n'est pas possible de connaître sans remonter à la couche application. Cependant, même avec cette restriction, on peut constater qu'un volume important du trafic est dû aux applications p2p. Dans la suite, on essaie de dégager une typologie simplifiée des flux et il est en fait inutile d'avoir une connaissance très précise des applications qui les engendrent ; on s'intéresse par la suite seulement à la différence entre flots longs et flots courts (notion à définir). Les petits transferts de signalisation sont en principe identifiables via le numéro de port.

application	pourcentage
non p2p	
http	14.6
ftp	2.1
nntp	1.9
autres	31.8
Total du trafic "non p2p" (du moins supposé tel)	50.4
p2p	
Edonkey	37.5
Kazaa&Morpheus	7.8
Napster	3.8
Gnutella	0.5
Total du trafic p2p (identifiable par les numéros de port)	49.6

TAB. 2.1 – Composition du trafic et contribution des flots courts.

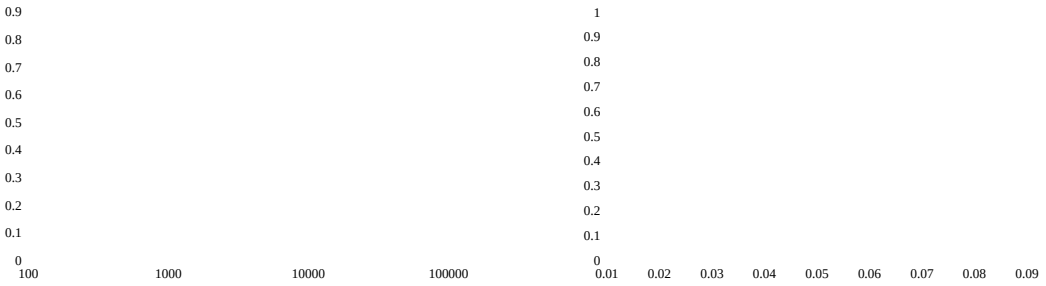
Pour analyser le trafic sur la base de flots, il est nécessaire en premier lieu de définir la notion de flot. Dans ce document, un flot est un ensemble de paquets qui partagent des caractéristiques communes, à savoir les mêmes adresses IP source et destination ainsi que les mêmes numéros de port TCP, en outre bien sûr que les paquets aient pour type de protocole TCP. Par ailleurs, comme l'objectif est de décrire le débit des flots, il est nécessaire de se concentrer sur la partie active des flots. Ainsi, de manière arbitraire, un flot sera déclaré comme terminé si aucun paquet du flot n'est observé pendant une certaine période de temps ($\theta = 5$ secondes pour les flots courts et $\Theta = 20$ secondes pour les flots longs qui seront définis par la suite).

Cette temporisation, en particulier celle de 5 secondes pour les flots courts, permet d'éviter un étalement temporel trop important des flots. Un tel étalement peut être dû au fait que certains paquets (par exemple les segments SYN, FIN ou RESET) peuvent arriver, et ce de manière tout à fait naturelle, loin des paquets engendrés pendant la phase de transfert du flot. Un étalement temporel trop important des flots peut introduire un biais significatif dans l'estimation du débit des flots. La temporisation de 5 secondes évite ce problème mais en contrepartie, certains flots peuvent être artificiellement fragmentés, provoquant l'apparition artificielle de flots constitués d'un seul paquet. Dans la suite, il sera admis que ces flots mono-paquet donnent naissance à un trafic résiduel modélisable par un bruit blanc.

La densité de probabilité de la taille des flots en octets est illustrée par la Figure 2.7(a). Il apparaît clairement que les flots sont majoritairement de petite taille ; la plupart des flots contiennent moins de 1.000 octets. Du point de vue du réseau, il est naturel de faire une distinction entre les petits et les grands flots :

- si un flot est suffisamment petit, il ne sort pas ou très peu de la phase de «slow start» et il est alors insensible au partage de bande passante susceptible d'être imposé par TCP,
- si un flot est suffisamment long, il peut entrer dans la phase de "congestion avoidance" de TCP et on peut alors espérer qu'il partage avec les autres flots longs la bande passante du réseau suivant un certain critère d'équité, idéalement max-min si l'on fait abstraction des temps d'aller et retour (RTT).

La figure 2.7(b) montre par ailleurs que la contribution des flots de moins de 20 paquets au trafic global est faible.



(a) Taille des flots en octets

(b) PDF de la contribution des flots de moins de 20 paquets au débit total

FIG. 2.7 – Taille des flots et contribution des petits flots.

Dans l'analyse effectuée ci-après, on adopte les définitions arbitraires suivantes :

Flot court : un flot court est un flot de moins de $L = 20$ paquets.

Flot long : un flot long est un flot de plus de $L = 20$ paquets.

Les définitions ci-dessus sont purement arbitraires mais elles permettent d'effectuer un premier tri entre les deux types de flots. Comme on peut le vérifier expérimentalement (cf. section 2.5.4 pour les détails), une modification du seuil de 20 paquets ne change pas fondamentalement les conclusions. Enfin, bien qu'il n'y ait pas de définition précise pour la terminologie souris et éléphants, les petits et les longs flots seront appelés respectivement souris et éléphants, même si les définitions retenues dans cette étude ne correspondent pas exactement aux coutumes habituelles dans la communauté de la métrologie du trafic Internet.

2.5.2 Trafic des souris

Intuitivement, les souris correspondent à de petits transferts de données qui peuvent être de plusieurs types :

- des petits volumes de données «utiles», typiquement des pages Web,
- des messages de signalisation liés à des transferts de fichiers beaucoup plus gros (par exemple des connexions de contrôle liées à une session ftp),
- des messages de signalisation et d'échange d'information de routage ou de dissémination de contenus dans les réseaux p2p.

La liste ci-dessus n'est bien évidemment pas exhaustive mais permet immédiatement de faire une distinction entre les souris p2p et les souris non p2p. Ces dernières sont associées aux applications classiques du type ftp, http, etc. Elles ont ceci de commun que pour une même adresse IP destination, un certain nombre de souris arrivent très proches les unes des autres. En effet, lors de la consultation d'une page Web par exemple, un certain nombre de connexions TCP peuvent être ouvertes de manière quasiment simultanée. Ceci peut être dû au rapatriement d'objets localisés sur des serveurs différents ou simplement au fait que certains navigateurs ouvrent toujours plusieurs connexions TCP,

bien que ce dernier point soit de moins en moins vrai avec les nouvelles spécifications de http (cf. la spécification de HTTP 1.1).

Les arguments présentés ci-dessus conduisent naturellement à essayer de grouper les souris non p2p sur la base de l'adresse IP de destination. De manière plus précise, on définit la notion de macro-souris non p2p de la manière suivante

Définition 1 (Macro-souris non p2p) *Une macro-souris non p2p est un ensemble de souris non p2p avec la même adresse IP de destination et arrivant dans un intervalle de δ secondes ; on impose par ailleurs qu'une macro-souris contienne plus de un paquet.*

Dans les expérimentations reportées ci-dessous, la constante δ a été prise égale à 1 seconde. Il est important de noter que par construction, les souris de un paquet isolées, qui peuvent apparaître artificiellement à cause de la temporisation de 5 secondes utilisée pour couper les flots, peuvent ne pas être incluses dans les macro-souris non p2p. Ces souris donnent naissance à un trafic résiduel.

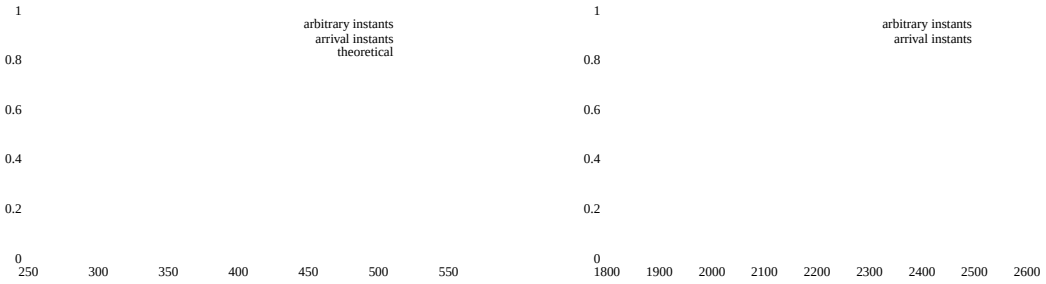
En ce qui concerne les souris p2p, il est a priori raisonnable de regrouper les souris p2p suivant l'adresse IP de la source. En effet, un membre d'un réseau p2p cherchant un contenu envoie un certain nombre de requêtes aux nœuds du réseau p2p pour le trouver. Mais ces requêtes donnent elles-mêmes naissance à des réponses de la part des nœuds en question. Il apparaît alors nécessaire d'introduire un deuxième niveau d'agrégation sur la base de l'adresse IP de destination. Il semble cependant difficile de corréler les adresses IP sources de requête avec les adresses IP destination des réponses. La manière dont répond un nœud à une requête dépend du protocole et le temps de réponse est difficilement maîtrisable. Ceci conduit à définir les macro-souris p2p de la manière suivante

Définition 2 (Macro-souris p2p) *Les souris p2p avec la même adresse IP source arrivant dans un intervalle de δ secondes puis les souris p2p avec la même adresse de destination arrivant elles aussi dans un intervalle de δ secondes sont agrégées pour donner naissance à des macro-souris p2p. On impose par ailleurs que les macro-souris p2p contiennent plus de un paquet.*

Comme précédemment, les souris isolées de un paquet peuvent ne pas être incluses dans les macro-souris et donnent naissance à un trafic résiduel.

Les macro-souris, qu'elles soient liées à des protocoles p2p ou non, correspondent à l'ensemble des messages en relation avec de la signalisation ou des transactions courtes ; ces messages sont engendrés lors de ce que l'on peut qualifier un «appel», bien que cette notion soit un peu étrange dans le cadre de l'Internet. Il est usuel de vérifier que les appels dans un réseau de télécommunications se produisent comme un processus de Poisson. Pour vérifier cette hypothèse, on calcule les distributions aux instants d'arrivée et à un instant arbitraire du nombre de macro-souris actives. Ces distributions sont données par la Figure 2.8.

Ces deux figures montrent clairement que les distributions aux instants d'arrivée et à un instant arbitraire sont quasiment confondues. Cette propriété est connue dans la théorie des processus ponctuels comme la propriété ASTA (Arrivals See Time Averages). Une telle propriété est satisfaite par le processus de Poisson, connue sous le nom de propriété PASTA (Poisson Arrivals See Time Averages). Par contre, la propriété ASTA à elle seule ne suffit pas à garantir que le processus est effectivement Poisson. Pour qu'il en soit ainsi, il faut que le processus d'arrivée vérifie des conditions plus fortes (avec des dépendances markoviennes par exemple). Dans la suite, on supposera que les processus d'arrivée des macro-souris p2p et non p2p sont Poisson ; ceci reste pour l'instant de



(a) Macro-souris non p2p

(b) Macro-souris p2p

FIG. 2.8 – Distribution aux instants d'arrivée et distribution stationnaires du nombre de macro-souris actives.

l'ordre de la conjecture mais comme on le verra par la suite dans la partie modélisation (cf. Livrable du Sous-projet 5), cette conjecture est cohérente avec certaines grandeurs mesurées. Les données empiriques montrent que les taux d'arrivée de ces processus sont égaux à $\lambda_m = 178$ arrivée/seconde pour les macro-souris non p2p et à $\lambda_\mu = 574$ arrivée/seconde pour les macro-souris p2p.

Les distributions complémentaires des durées des macro-souris p2p et non p2p sont données par la figure 2.9. Il ressort de cette figure que les durées peuvent être très bien approximées par des lois de Weibull à deux paramètres, ce qui signifie que la durée S est telle que

$$\mathbb{P}\left(S > x\right) = \exp\left(-\left(\frac{x}{\eta}\right)^{\beta}\right), \tag{2.3}$$

où η est le paramètre d'échelle (exprimé en secondes) et β celui de forme (sans unité). Les paramètres calculés empiriquement sont donnés par le Tableau 2.2.

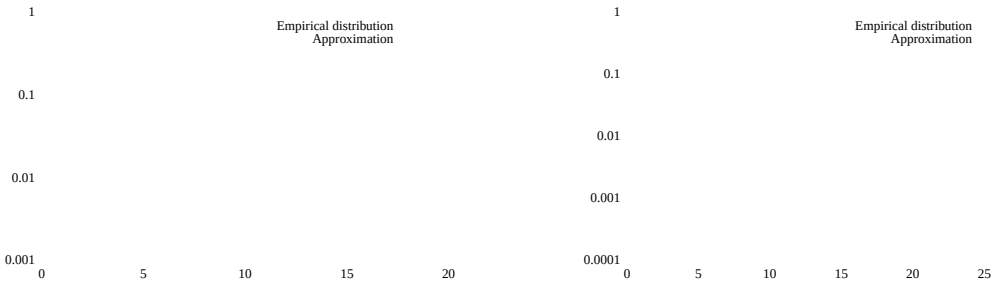
type de macro-souris	λ	β	η
non p2p	$\lambda_m \approx 178$	$\beta_m = 0.8$	$\eta_m = 1.78$
p2p	$\lambda_\mu \approx 574$	$\beta_\mu = 1.2$	$\eta_\mu = 3.9$

TAB. 2.2 – Paramètres des macro-souris.

Enfin, une macro-souris contient un nombre aléatoire de souris. Le nombre moyen de souris dans une macro-souris non p2p est de 2.27 et de 3.43 pour une macro-souris p2p. La densité du nombre de souris dans une macro-souris est donnée par la figure 2.10.

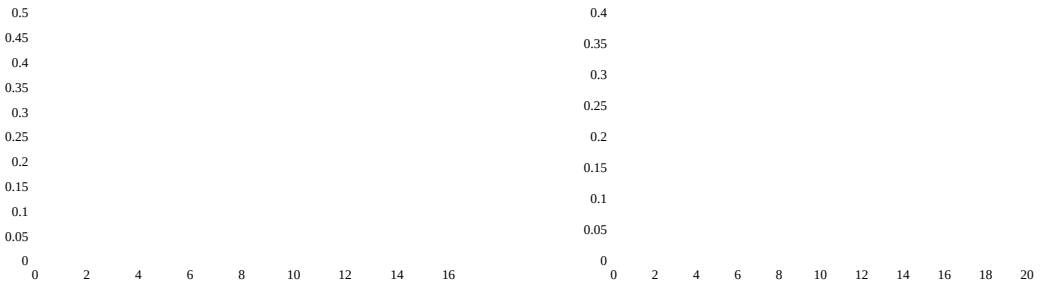
2.5.3 Caractéristiques des éléphants

Pour décrire le trafic des éléphants, il est essentiel de noter quelques points élémentaires. Les éléphants correspondent aux flots qui comprennent plus que 20 paquets, mais tous ces flots ne sont



(a) Macro-souris non p2p (b) Macro-souris p2p

FIG. 2.9 – Distributions complémentaires des durées des macro-souris (en secondes).



(a) Macro-souris non p2p (b) Macro-souris p2p

FIG. 2.10 – Densité du nombre de souris dans une macro-souris.

pas équivalents. En effet, certains éléphants sont naturellement composés de paquets de petite taille et ont un débit très faible. Un éléphant de ce type peut correspondre par exemple au flot d'acquittements (ACK segments) envoyés par un terminal qui rapatrie un fichier de très grande taille. Avec l'émergence des protocoles p2p, ce cas de figure se produit de plus en plus souvent. Des clients ADSL télé- chargent des documents de très grande taille (plusieurs giga octets) sur des périodes de temps très longues. Le fonctionnement même de TCP implique alors qu'un très grand nombre d'acquittements est engendré par le terminal rapatriant le document.

Pour faire la distinction entre les éléphants qui sont composés essentiellement de segments d'acquittement et ceux qui transportent de la "vraie" information, on introduit un seuil pour la taille moyenne des paquets. Si cette taille est trop faible, alors cela indique vraisemblablement que le flot est composé de messages d'acquittement. Dans les mesures reportées ci-après, le seuil a été fixé à $\bar{P} = 80$ octets. Un éléphant avec une taille moyenne de paquet plus petite que 80 octets sera considéré comme un éléphant d'acquittements.

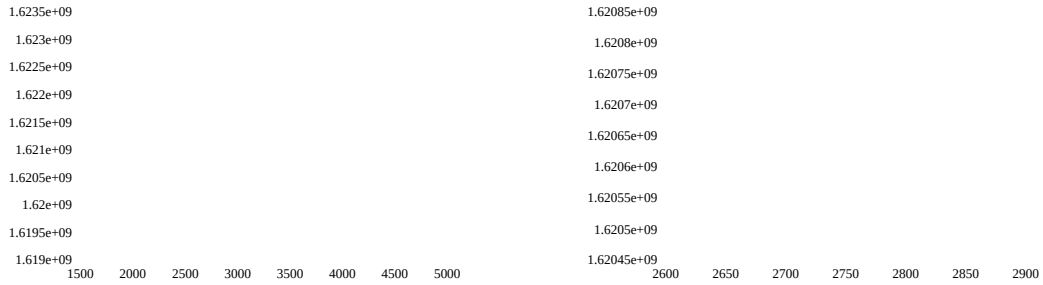
Le trafic engendré par les éléphants d'acquittements résulte de la superposition de flux de paquets plus ou moins espacés. On peut dès lors s'attendre à ce que le débit instantané soit un bruit blanc. La figure 2.11 donne la densité spectrale de la série chronologique décrivant le débit des éléphants d'acquittement. On peut constater que cette densité spectrale est relativement constante et donc que la série chronologique correspondante est un bruit blanc (non centré). La densité spectrale n'est pas tout à fait constante près de l'origine ; cela peut correspondre à une légère non stationnarité. Le débit moyen empirique de ce type d'éléphants est d'environ 1 Mbit/s, ce qui est très faible par rapport aux 100 Mbit/s pour le trafic des autres éléphants.

FIG. 2.11 – Densité spectrale du débit des éléphants d'acquittement.

Dans la suite, on s'intéresse aux éléphants composés de segments véhiculant des données, c'est à dire, les éléphants avec une taille moyenne de paquet plus grande que 80 octets. Alors que les hypothèses habituelles consistent à supposer que ce type d'éléphant est en permanence actif et que les éléphants se partagent le débit du goulot d'étranglement dans le réseau, on constate sur quelques trajectoires que les phases de transmission des éléphants ne sont pas continues.

La figure 2.12 représente l'évolution au cours du temps du numéro de séquence des paquets pour un éléphant particulier. On constate que le numéro de séquence croît linéairement sauf sur quelques intervalles de temps où des pertes se produisent. La figure 2.12 montre également le comportement du numéro de séquence lors de pertes. Certaines sont accompagnées de périodes de "slow start" et d'autres non. Bien que cet éléphant subisse des pertes, son comportement est conforme à ce qui est prévu par la théorie pour des connexions TCP semi-permanentes : de longues périodes de

transmission interrompues par des épisodes de pertes. Plusieurs études traitent de cette situation et permettent de prédire le débit moyen de la connexion TCP (cf. Padhye *et al*), conduisant aux formules en $c/\sqrt{p}$, pour une certaine constante c, la grandeur p désignant la probabilité de perte d'un paquet.

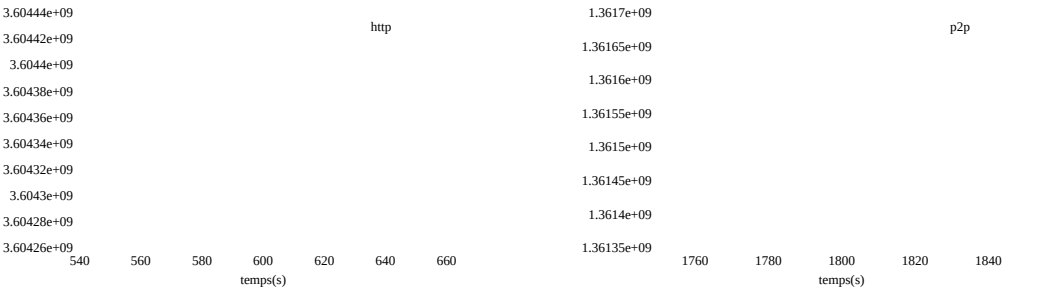


(a) Sur toute la durée de l'éléphant

(b) Pendant des périodes de pertes

FIG. 2.12 – Evolution du numéro de séquence d'un éléphant de longue durée (en secondes).

La figure 2.13 présente des éléphants avec des comportements beaucoup moins réguliers. Le premier est associé à une transaction HTTP, qui a consisté à rapatrier une quantité d'information assez importante suivie de petits rapatriements. Le second éléphant est associé à un protocole p2p. Il apparaît que cet éléphant est essentiellement composé de transfert de petites tailles. En réalité cet éléphant est une suite de souris séparées par des périodes de temps assez longues. Ce phénomène est assez typique des éléphants p2p observés et doit être dû au fonctionnement même des protocoles p2p ou au type d'usage (en l'occurrence l'usage de l'éléphant montré par la figure 2.13 correspond à du *chat*).



(a) Eléphant HTTP

(b) Eléphant p2p

FIG. 2.13 – Eléphants avec des comportements irréguliers.

Toujours dans l'optique de décrire le débit sur le lien de transmission, il est naturel de se ramener sur les phases de transmission en découpant les éléphants en mini-élephants et en souris. De manière plus précise, un mini-éléphant est un groupe de paquets de plus de 20 paquets ; un mini-éléphant est terminé quand aucun paquet de celui-ci n'est observé pendant un période de 20 secondes. Une souris dans le contexte des éléphants est un groupe de moins de 20 paquets n'appartenant pas à un mini-éléphant.

En adoptant cette découpe des éléphants, on peut analyser les deux types de trafic séparément en utilisant les mêmes méthodes que pour les souris habituelles étudiées dans la section précédente. On trouve alors que les souris et les mini-éléphants arrivent suivant des processus de Poisson et que leurs durées sont distribuées suivant des lois de Weibull. Le tableau 2.3 donne les valeurs calculées empiriquement.

	mini-éléphants	souris (éléphants)
Taux d'arrivée λ (en s-1)	26.4	23.35
Facteur de forme β	0.49	0.76
Facteur d'échelle η	122.94	16.94

TAB. 2.3 – Caractéristiques des mini-éléphants et des souris (associées aux éléphants).

2.5.4 Sensibilité aux paramètres

Les différents paramètres utilisés dans les sections précédentes pour introduire les notions de souris et de macro-souris ainsi que les éléphants sont rappelés dans le tableau 2.4.

Symbole	Interprétation
Δ	Période d'intégration
L	Nombre maximal de paquets dans une souris
θ	Valeur de la temporisation pour terminer un flot court
Θ	Valeur de la temporisation pour terminer un flot long
δ	Longueur de l'intervalle pour agréger les souris
$\bar{P}$	Seuil pour la longueur moyenne des paquets pour détecter un flot d'acquittements

TAB. 2.4 – Paramètres utilisés pour décomposer le trafic.

Les résultats décrits jusqu'à présent concernent une capture de trafic effectuée en décembre 2002. Le tableau 2.5 donnent les résultats obtenus pour une capture de trafic effectuée en décembre 2003 entre 21h00 et 23h00. Cette période correspond à une activité soutenue de la part des clients ; la charge moyenne du lien observée est d'environ 45% et seul le trafic TCP descendant est analysé. Qualitativement les résultats restent inchangés par rapport à la capture de décembre 2002. Evidemment, les valeurs numériques changent mais les formes des distributions (sous forme de lois de Weibull) restent les mêmes.

Les paramètre L pour la taille limite en paquet d'une souris peut apparaître en premier abord très critique. Augmenter L peut éventuellement casser le caractère Weibull de la loi des durée des souris. Le tableau 2.6 montre en fait qu'il n'en est rien et que d'augmenter le seuil L entraîne une

Type de flot	Paramètre	Décembre 2002	Novembre 2003
Souris non p2p	$\lambda(s^{-1})$	279.486	594.884
	η (in second)	1.035	1.953
	β	0.673	0.726
	$\mathbb{E}[\mathcal{S}]$(s)	1.390	2.399
Macro-souris non p2p	λ(in s^{-1})	177.936	326.464
	$\eta(s)$	1.780	3.172
	β	0.800	0.873
	$\mathbb{E}[\mathcal{S}]$(s)	2.136	3.249
	mean number of mice	2.270	2.419
Souris p2p	$\lambda(s^{-1})$	1039.277	2390.600
	$\eta(s)$	2.800	3.789
	β	1.030	0.969
	$\mathbb{E}[\mathcal{S}]$(s)	2.798	3.757
Macro-souris p2p	λ(in s^{-1})	574.070	903.342
	$\eta(s)$	3.898	6.362
	β	1.193	1.207
	$\mathbb{E}[\mathcal{S}]$(s)	3.550	5.695
	Number of mice	3.430	4.316
Mini-éléphants	$\lambda(s^{-1})$	26.400	40.010
	$\eta(s)$	122.947	64.043
	β	0.494	0.399
Souris d'éléphant	$\lambda(s^{-1})$	23.350	44.349
	$\eta(s)$	16.940	15.568
	β	0.760	0.842
	$\mathbb{E}[\mathcal{S}]$(s)	18.506	14.288

TAB. 2.5 – Valeurs numériques pour les traces de décembre 2002 et november 2003.

simple modification des valeur numérique mais pas des propriétés qualitatives des flots. Les résultats reportés dans le tableau 2.6 concernent la trace capturée en novembre 2003. De même, les éphants ne sont altérés par une variation modeste du paramètre L.

nombre de paquets		10	20	100
Souris non p2p	λ	274.435	279.486	354.957
	η	1.415	1.035	0.800
	β	0.846	0.673	0.485
	$\mathbb{E}[\mathcal{S}]$(s)	1.237	1.390	2.096
Macro-souris non p2p	λ	163.577	177.936	217.351
	η	1.946	1.780	1.288
	β	0.938	0.800	0.529
	$\mathbb{E}[\mathcal{S}]$(s)	1.902	2.136	3.052
Souris p2p	λ	1034.657	1039.277	1074.519
	η	2.981	2.800	2.854/0.100
	β	1.110	1.030	1.002/0.345
	$\mathbb{E}[\mathcal{S}]$(s)	2.722	2.798	2.852
Macro-souris p2p	λ	566.966	574.070	588.471
	η	4.066	3.898	3.828/0.200
	β	1.273	1.193	1.077/0.375
	$\mathbb{E}[\mathcal{S}]$(s)	3.519	3.550	3.723

TAB. 2.6 – Impact de la variation du paramètre L.

Chapitre 3

Dimensionnement en accès de réseau

3.1 Modèles de files d'attente PS à sources finies

3.1.1 Introduction

Le dimensionnement des ressources d'un réseau nécessite une bonne compréhension de la relation de performance qui lie la capacité d'un lien, le trafic qui lui est offert (demande exprimée par les utilisateurs) et la Qualité de Service (QoS) qui en résulte. Les fameuses formules d'Erlang et d'Engset sont des exemples typiques d'une telle relation, fournissant la probabilité de blocage des appels dans le cas des réseaux téléphoniques à commutation de circuit. Le fait que ces formules aient été utilisées avec succès depuis plusieurs décennies, malgré les évolutions significatives d'usage des réseaux téléphoniques, est en grande partie dû à leur propriété fondamentale dite "d'insensibilité" : la performance ne dépend pas des caractéristiques détaillées du trafic (telles que la distribution statistique des durées d'appel) mais seulement de la valeur moyenne de son intensité. Nous montrons dans ce chapitre qu'une telle relation peut être obtenue pour du trafic de données dans un réseau d'accès. La notion d'appels téléphoniques doit alors être transposée en celles de "flots" et de "sessions" que l'on définira par la suite.

Le trafic de données sous contrôle du protocole TCP constitue encore à l'heure actuelle, malgré l'émergence attendue des services audio/vidéo temps-réel, la grande majorité (environ 95% du débit) du trafic Internet [20]. Ce trafic est élastique dans le sens où son débit peut s'ajuster flexiblement aux conditions de congestion du réseau pour utiliser autant que possible la bande passante disponible. La performance d'un flux de données, telle que perçue par l'utilisateur, se mesure alors d'une manière globale par la durée de son transfert ou, de manière équivalente, par le débit moyen effectivement réalisé.

On considère donc dans ce qui suit un lien d'accès à un réseau IP ne transportant que du trafic élastique de données. La performance y est analysée au niveau flots, en supposant que les mécanismes de TCP au niveau paquets permettent d'atteindre un certain degré de partage équitable de la capacité du lien entre les différents flots en concurrence. Sous des hypothèses réalistes de structure des sessions utilisateurs, des expressions du débit effectif moyen sont obtenues en fonction de paramètres systèmes et d'une mesure de la demande de trafic par utilisateur. L'un des objectifs prin-

cipaux recherchés dans ce travail est de tirer les enseignements pratiques d'une telle modélisation de la performance et d'en dégager des règles simples de dimensionnement des liens d'accès, afin de permmettre à un opérateur d'atteindre de manière économique ses objectifs de fourniture de QoS.

L'approche de modélisation au niveau flots conduit à analyser la performance du trafic élastique dans le cadre des files d'attente à discipline de service "Processor Sharing" (PS) [11]. Nabe et al. [22] ont par des simulations extensives confirmé la validité de ces modèles PS pour dimensionner des liens d'accès à l'Internet. Des modèles PS généralisés, tenant compte d'une limitation de débit à l'accès, ont été étudiés notamment dans [1] et [29] ; cependant, ces articles ne traitent pas le problème d'une population source de taille finie puisqu'ils supposent un processus Poissonnien des arrivées de flot.

3.1.2 Modèle de trafic

On considère un lien de capacité (débit) C transportant du trafic de données et partagé par N utilisateurs ayant tous le même débit d'accès c. Il peut s'agir typiquement d'un lien connectant un ensemble de lignes xDSL au premier routeur d'un réseau IP. Le trafic généré par un utilisateur (en session active) est composé d'une succession de flots (périodes de transmission) séparés par des périodes de silence (inactivité) aussi appelées "temps de réflexion", comme l'illustre typiquement la Figure 3.1. Chaque flot correspond au transfert d'un document numérique (une page Web, un e-mail, une séquence vidéo non temps-réel, ...) ou plus généralement d'une séquence de documents émis successivement ou en parallèle. On pourra se référer par exemple à [20], [24] pour une discussion plus précise de la structure en flots du trafic IP.

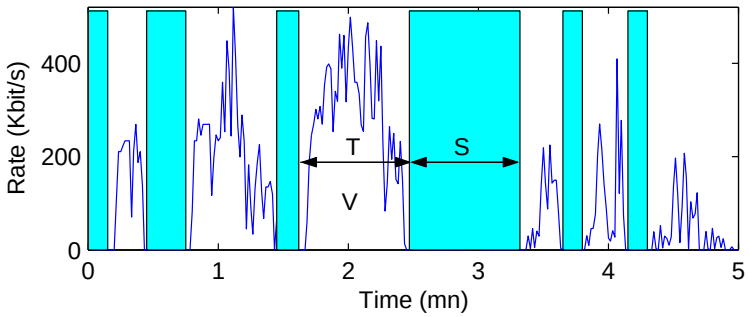


FIG. 3.1 – Exemple de session d'un usager ADSL

Il est clair qu'en général la longueur des flots, leur durée et la durée des périodes de silence sont des grandeurs aléatoires extrêmement variables et corrélées entre elles. Néanmoins, comme il sera montré plus loin, seule la connaissance de leurs valeurs moyennes, notées respectivement V, T et S, est indispensable. A noter que V et S sont des paramètres intrinsèques de la demande de trafic, tandis que T est déterminé par l'état du système.

La QoS perçue par les usagers dépend essentiellement de la durée moyenne de transfert d'un document (flot) de taille donnée, c'est-à-dire d'un débit moyen effectivement réalisé. Pour des raisons de tractabilité analytique, il est commode de caractériser ce dernier paramètre de performance par un *débit utile* d défini par : $d = V/T$ (différent en général du débit effectif moyen).

Le *trafic écoulé* par utilisateur résulte de la charge réellement observée durant un certain intervalle de temps et dépend bien sûr de la QoS fournie par le réseau. En valeur moyenne, il se définit par : $b = V/(T + S)$. Au contraire, le *trafic offert* par utilisateur correspond au trafic qu'un usager écoulerait si le débit d'accès c était toujours disponible ; c'est donc une caractéristique intrinsèque indépendante des autres utilisateurs et des conditions d'écoulement du trafic. En valeur moyenne, il s'écrit : $a = V/(V/c + S)$.

De ces définitions on déduit l'identité suivante, très simple, entre les quatre paramètres de débit, valable pour chaque usager ou pour chaque groupe d'usagers aux caractéristiques homogènes (même trafic offert et même débit d'accès) :

$$\frac{1}{a} + \frac{1}{d} = \frac{1}{b} + \frac{1}{c}. \tag{3.1}$$

3.1.3 Performance et dimensionnement

Chaque session utilisateur est modélisée comme une succession aléatoire et stationnaire de transferts de flot et de temps de réflexion, ce qui correspond à l'hypothèse habituelle de trafic stationnaire à l'heure chargée. On suppose par ailleurs qu'à tout instant les flots simultanément en présence se partagent équitablement la bande passante : lorsque x flots sont présents, chacun d'eux dispose d'un débit instantané $\min\{C/x, c\}$. Ce partage idéalement équitable peut être vu comme résultant de modèles de performance de TCP bien connus, tels que [25], dans le cas où les connexions individuelles sont soumises à des conditions identiques de transport de bout en bout (fenêtres d'émission TCP, temps de propagation aller-retour (RTT), probabilité de perte de paquets, etc.).

Modèle de performance

Considérons le cas général où la population source est formée de K classes d'usagers, de populations N_k, offrant chacune un trafic par usager a_k, $k = 1, \ldots, K$. Introduisons ici une mesure auxiliaire du trafic offert, $r_k = V_k/S_k$, qui simplifie l'écriture des expressions à suivre. Reliée au trafic offert a_k par la relation $a_k = r_k c/(r_k + c)$, cette expression du débit représente en fait le trafic offert par utilisateur en l'absence de contrainte de débit d'accès. Soit $x_k(t)$ le nombre de flots de classe k en cours de transfert au temps t. Sous les hypothèse formulées ci-dessus, le vecteur $x(t) = (x_1(t), \ldots, x_K(t))$ converge vers un état stationnaire x de loi de probabilité :

$$\pi(x) = \pi(0) \prod_{k=1}^{K} \binom{N_k}{x_k} \left(\frac{r_k}{c}\right)^{x_k} \cdot \prod_{i=1}^{L(x)} \frac{i}{\min(i, C/c)}. \tag{3.2}$$

où $L(x)$ désigne la longueur du vecteur d'état (nombre total de flots en cours) : $L(x) = \sum_k x_k$. La relation de normalisation à 1 de la somme des probabilités d'état détermine la probabilité $\pi(0)$.

Cette distribution stationnaire du vecteur d'état ne dépend que des intensités moyennes de trafic offert par classe, par l'intermédiaire des paramètres r_k. Elle est insensible aux distributions de longueur de flot et de durée de silence et à leurs éventuelles corrélations [17], comme déjà identifié par [3] et [18] dans le cas de sources homogènes. On peut établir simplement l'expression (3.2) à partir des équations de balance locale si l'on suppose un modèle markovien (distributions exponentielles et indépendantes des longueurs de flot et des durées de silence). Le système forme alors un processus de naissance et de mort dont les équations d'évolution s'écrivent en régime stationnaire :

- taux d'arrivée des flots de classe k lorsque x flots sont présents : $\lambda_{x,k} = (N_k - x_k)/S_k$, $x_k \leq N_k$
- taux de départ des flots de classe k lorsque x flots sont présents : $\mu_{x,k} = c/V_k \min(x_k, x_k/L(x).C/c)$
- équations de balance locale entre les états $x - e_k$ et x, pour toute direction de transition e_k, $k = 1, K$: $\lambda_{x-e_k,k}\pi(x - e_k) = \mu_{x,k}\pi(x)$, $x_k \leq N_k$

Il s'agit en fait des mêmes équations d'équilibre que dans le modèle d'Engset (multi-classes) en téléphonie à commutation de circuit [10], où C/c représente un nombre de serveurs équivalent ; c/V_k un taux de service équivalent proposé par chaque serveur, et où une attente est possible (pas de rejet) lorsque tous les serveurs sont occupés.

Le débit utile par classe d_k s'obtient aisément en fonction du nombre moyen $E[x_k]$ de flots de classe k, par application de la formule de Little aussi bien aux usagers de chaque classe en phase active (transfert d'un flot) qu'à ceux en phase inactive (période de réflexion) :

$$d_k = r_k \frac{N_k - E[x_k]}{E[x_k]}. \tag{3.3}$$

Combinant cette expression avec la relation (3.1) appliquée à chaque classe, on obtient une relation de conservation très simple entre débit écoulé et débit utile : $N_k b_k = E[x_k] d_k$.

Modèle approché

En pratique, lorsque le trafic de chaque usager est faible au regard de la demande totale, le débit utile par classe d_k peut être approximé par un débit utile moyen (sur tous les flots de tous les utilisateurs) s'écrivant sous la forme :

$$d \equiv V/T = \frac{\sum_k r_k(N_k - E[x_k])}{\sum_k E[x_k]}. \tag{3.4}$$

Cette approximation se justifie par application de la propriété MUSTA ("Moving Units See Time Averages") [32] que l'on explicite plus en détail dans [5]. Le modèle de performance peut être grandement simplifié au moyen d'approximations supplémentaires. On suppose qu'il existe deux régimes opératoires limites : un régime saturé où le lien est en permanence pleinement utilisé et un régime transparent où chaque flot dispose à tout instant du débit d'accès.

Régime transparent - Lorsque le trafic offert total $\sum_{k=1}^{K} N_k a_k$ esr inférieur à la capacité C, le débit utile moyen d est proche du débit d'accès c. Il s'en déduit par la relation (3.1) que les trafics offert et écoulé, par classe, sont eux aussi approximativement égaux : $a_k \approx b_k$.

Régime saturé - A l'opposé, lorsque $\sum_{k=1}^{K} N_k a_k > C$, le trafic écoulé total $\sum_{k=1}^{K} N_k b_k$ est pratiquement égal à la capacité C. A nouveau à l'aide de (3.1) appliquée à chaque classe k, on en déduit la relation approchée suivante entre capacité du lien et débit utile moyen :

$$C \approx \sum_{k=1}^{K} \frac{N_k}{1/a_k + 1/d - 1/c}. \tag{3.5}$$

La Figure 3.2 illustre la qualité des approximations faites dans le cas d'une demande homogène de trafic ($K = 1$), pour différentes valeurs du trafic offert par usager et une population de taille

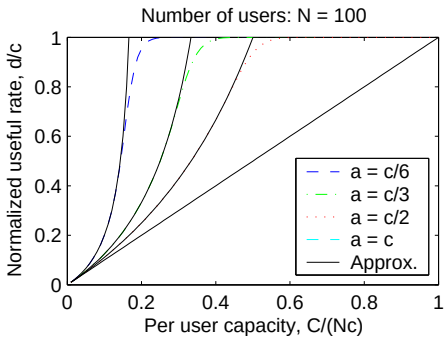


FIG. 3.2 – Relation entre capacité et débit utile pour différents trafics offerts a

$N = 100$. L'approximation est d'autant meilleure que a est proche de c et, comme montré dans [5], que N est grand. Par ailleurs, une application du modèle exact (3.3) au cas de deux classes d'usagers aux trafics très hétérogènes [6] montre que l'approximation de l'identité des débits utiles par classe reste précise tant que le rapport C/c est d'un ordre de grandeur supérieur à 10.

Application au dimensionnement

La relation approchée (3.5) permet de dimensionner le lien si l'on souhaite opérer en régime saturé : la capacité à installer C s'obtient en fonction d'un débit utile cible d que l'on se fixe (objectif de performance). Observant que la fonction f définie par $f(a) \equiv 1/(1/a + 1/d - 1/c)$ est concave, on remarque que la capacité requise par utilisateur est maximale lorsque la demande de trafic est homogène au sein des utilisateurs, pour un trafic offert total donné. On en déduit qu'une stratégie conservative de dimensionnement consisterait en l'installation d'une capacité $C = Nf(a)$ où a est le trafic offert par utilisateur moyenné sur l'ensemble des classes : $a = \frac{1}{N}\sum_{k=1}^{K} N_k a_k$.

La Figure 3.3 illustre ce principe de dimensionnement pour un débit d'accès $c = 500$ Kbit/s et un trafic offert moyen par usager $a = 50$ Kbit/s. Par exemple, une capacité par utilisateur de 36 Kbit/s est nécessaire dans ce cas pour garantir un débit utile de 100 Kbit/s. On note que ces résultats, exprimés par utilisateur, ne dépendent pas du nombre N d'usagers ; en d'autres termes, il n'y a pas d'économie d'échelle (tant que l'on ne met pas en oeuvre de politique de rejet des demandes et que l'on ne se base pas, comme en téléphonie, sur un objectif de probabilité de blocage).

La procédure ci-dessus s'appuie sur une valeur moyenne du trafic offert par uitilisateur. On montre dans [5] qu'il est possible de raffiner la formule de dimensionnement, toujours dans le cadre d'un modèle approché, pour tenir compte de déséquilibres importants dans la répartition du trafic. Sont en particulier évalués les impacts respectifs d'une segmentation entre usagers actifs et non actifs d'une part, et de la présence d'utilisateurs insatiables (au sens où leur trafic offert est proche du débit d'accès) d'autre part. Dans les deux cas, la concavité de la fonction f induit une amélioration de la performance globale lorsque le degré d'hétérogénéité du trafic augmente.

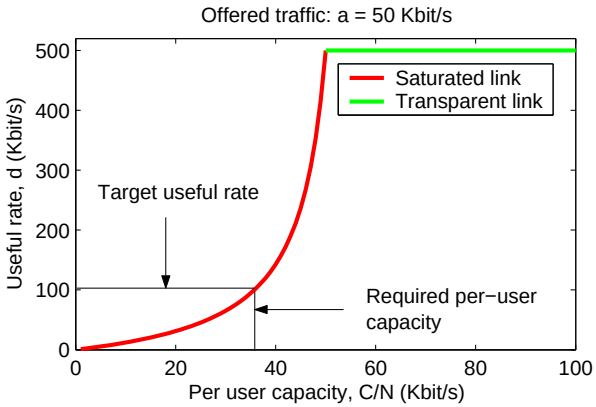


FIG. 3.3 – Fonction de dimensionnement basée sur le modèle approché

3.1.4 Généralisation

Partage de débit inéquitable

En réalité, le partage de la capacité réalisé par TCP n'est pas équitable dans la plupart des cas. Et ce pour plusieurs raisons, parmi lesquelles : un flot peut être composé de plusieurs connexions TCP en parallèle ; la valeur de RTT a un impact fort sur le débit réalisé [25] et peut être très variable d'une connexion à une autre (en fonction de la distance géographique et du nombre de noeuds traversés) ; les terminaux d'usagers sont susceptibles d'utiliser différentes versions de TCP ; ...

Pour évaluer l'impact d'un partage inéquitable, on étudie à présent un cas extrême où une fraction des usagers (classe 1) bénéficie d'une priorité sur les autres usagers (classe 2). Cette priorité est à entendre ici en termes de consommation de bande passante : les flots de classe 1 utilisent autant de débit que possible dans la limite de leur débit d'accès, n'étant aucunement contraints par la présence des flots de classe 2. Nous ne donnons ci-dessous que les équations spécifiques qui diffèrent du modèle général multi-classes (cf. §3.1.3) :

- taux de départ des flots de classe 1 : $\mu_{x,1} = c/V_1 \min(x_1, C/c)$
- taux de départ des flots de classe 2 : $\mu_{x,2} = c/V_2 \min(x_2, C/c - x_1)$ si $x_1 \leq C/c$ (= 0 sinon)

Ce système à priorité ne vérifie pas les équations de balance locale (et la performance n'est plus insensible). Il est donc nécessaire, sous des hypothèses markoviennes, de résoudre sous forme numérique le système linéaire formé par les équations de balance globale (en notant que l'une de ces équations doit être remplacée par la relation de normalisation $\sum_x \pi(x) = 1$) :

$$\sum_{k=1}^{K} \lambda_{x-e_k,k}\pi(x-e_k) - \sum_{k=1}^{K} \mu_{x,k}\pi(x) = \sum_{k=1}^{K} \lambda_{x,k}\pi(x) - \sum_{k=1}^{K} \mu_{x+e_k,k}\pi(x+e_k). \qquad (3.6)$$

Les probabilités d'état étant ainsi obtenues, les formules de performance exactes (3.3) peuvent être appliquées. On peut aussi, de même que précédemment, raisonner en termes d'approximations pour obtenir les résultats suivants :

Régime transparent - $N_1a_1 + N_2a_2 < C$: les débits utiles d_1 et d_2 sont proches de c.

Régime saturé - $N_1a_1 + N_2a_2 > C$: le trafic écoulé total $N_1b_1 + N_2b_2$ est pratiquement égal à la capacité C. Si $N_1a_1 \geq C$, le débit utile des flots de classe 2 est pratiquement nul et celui des flots de classe 1 vérifie la relation :

$$C \approx \frac{N_1}{1/a_1 + 1/d_1 - 1/c}. \tag{3.7}$$

Si $N_1a_1 < C < N_1a_1 + N_2a_2$, d_1 est proche de c et d_2 satisfait à :

$$C \approx N_1a_1 + \frac{N_2}{1/a_2 + 1/d_2 - 1/c}. \tag{3.8}$$

La Figure 3.4 présente les résultats du modèle approché pour une proportion de 10% d'usagers de classe 1 et un trafic offert uniforme de 50 Kbit/s par utilisateur de chaque classe. La performance des usagers favorisés est bien entendu grandement améliorée, mais une observation plus intéressante est que celle des usagers de classe 2 ne souffre que très marginalement par rapport à la situation de partage équitable. Ce comportement reste remarquablement robuste si l'on applique le modèle exact ci-dessus, sauf à considérer des situations extrêmement défavorables, et peu réalistes, où priorité est donnée aux utilisateurs les plus gourmands [6].

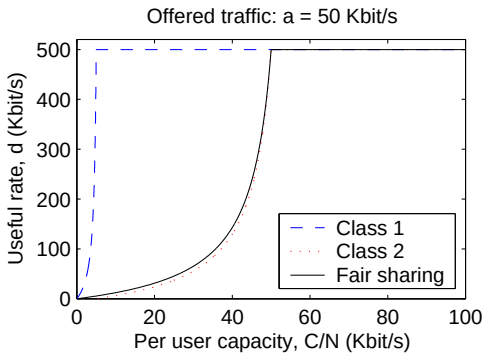


FIG. 3.4 – Performance approchée dans le cas d'un partage de débit inéquitable

On traite de manière similaire un autre type de généralisation du modèle, à savoir l'effet d'un multiplexage d'offres de service différentes (débits d'accès différenciés) [6].

3.1.5 Conclusion

Le trait essentiel qui ressort des analyses ci-dessus est que le trafic offert moyen par utilisateur constitue le paramètre clé pour dimensionner des liens d'accès à un réseau IP. La connaissance de

caractéristiques plus détaillées de la structure du trafic n'est pas nécessaire puisque la performance des flots élastiques y est dans une large mesure insensible. Par ailleurs, la prise en compte d'un trafic moyen conduit à un dimensionnement conservatif en cas de forte hétérogénéité de la demande. Soulignons pour terminer que parmi les deux stratégies possibles de dimensionnement, assurer la transparence du lien ou opérer en régime saturé avec un objectif de débit utile garanti, la seconde peut se révéler délicate à manier en raison de la forte sensibilité de l'indicateur de performance, le débit utile, à la précision avec laquelle le trafic offert a été estimé.

3.2 Modèle de TCP en surcharge

Alors que dans la section précédente, le nombre de clients était fini, on s'intéresse à présent à un nombre infini de sources. On étudie plus particulièrement le comportement de TCP quand le lien sature.

Il a été montré dans la littérature technique que pour le modèle processor sharing, la dégradation des performances en cas de surcharge est progressive. Nous allons voir dans cette section que c'est également le cas avec TCP, protocole transportant la quasi-totalité du trafic internet. Cette dégradation dépend bien sûr du trafic offert et de la durée de la congestion, mais également de la distribution de la taille des transferts : la détérioration est d'autant plus lente que la variance des tailles de flux est élevée. Par contre, si avec processor sharing les performances des petits transferts se dégradent moins vite que celles des gros transferts, ce n'est pas le cas avec TCP. L'idée proposée est alors de prioriser les petits flux pour ralentir encore cette dégradation. Nous verrons enfin pour conclure comment appliquer ces résultats au dimensionnement des réseaux.

3.2.1 Performances de TCP en surcharge

Bien que le modèle stationnaire de partage de charge (Processor Sharing - PS) prédise des dysfonctionnements lorsque la charge est supérieure à 1, il a été montré dans [**?**] que cette dégradation est progressive : les connexions continuent à être servies, et le nombre $N(t)$ de connexions actives augmente linéairement en fonction du temps t écoulé : $\lim_{t\to\infty} \frac{N(t)}{t} = \alpha$, avec la pente α définie par la relation : $\alpha = \lambda(1 - \int e^{-\alpha y} F(dy))$, où F est la distribution des temps de service.

Nous avons observé dans [**?**] le même type de dégradation pour TCP, see Figure 3.5. Comme pour PS, cette dégradation est d'autant plus lente que la taille des transferts est variable. Or la variance de la distribution de ces tailles de transfert est effectivement très élevée. On peut en pratique la modèliser par une distribution de Pareto, dont l'estimation du paramètre est relativement variable, en général entre 1.1 et 1.5 selon les analyses (nous avons trouvé 1.5 sur les mesures que nous avons faites sur un PoP ADSL), ce qui donne dans tous les cas une variance infinie.

Par ailleurs, pendant la période de surcharge, pour PS, les connexions sont d'autant plus impactées que leur taille est grande. $\frac{W_k}{k}$ converge en distribution vers $\frac{e^{\alpha y}-1}{\lambda} F(dy)$ lorsque t tend vers l'infini, où W_k est le temps de transfert du k-ième transfert arrivé. Le partage de charge permet au système de continuer de servir les petites connexions malgré la présence de gros transferts en cours.

Par contre les modèles PS et TCP divergent quant à l'estimation de l'influence de la taille sur l'impact perçu par les connexions. Nous observons en effet pour TCP, dans [**?**] et cf. Figure 3.6, des temps de transferts quasi-linéaires en fonction de la taille des transferts, donc aucun avantage pour

les petites connexions par rapport aux grandes, au moins tant que la surcharge ne devient pas très importante.

Nous montrons dans [**?**] et cf. Figure 3, où g représente l'estimation de la pente de $E(W_t/t)$ que ce phénomène peut être expliqué par le fait que TCP régule l'émission des connexions en fonction des pertes de paquets détectées, que ce soit la taille de la fenêtre d'émission ou l'intervalle de temps entre deux réémissions d'un paquet perdu. Les pertes de paquets, dues aux saturations du buffer du noeud congestionné, touchent aléatoirement les connexions, indépendamment de leur taille lorsque le nombre de connexions actives est grand comparé à la taille du buffer en paquets, car toutes les connexions ont des tailles de fenêtre de 1 ou 2 paquets. Les petites connexions sont donc autant impactées que les grosses.

3.2.2 Classification des flux selon leur taille

Suite aux observations précédentes, nous avons donc été conduits à rechercher un mécanisme permettant d'améliorer les performances des petites connexions sur Internet, de manière à se rapprocher du fonctionnement PS. En effet, d'une part la majeure partie du trafic est composée de petites connexions et donc les avantager permet de maximiser la satisfaction des utilisateurs, d'autre part les temps de transferts des grosses connexions sont de toute manière importants, et celles-ci vont donc se prolonger au delà de la période transitoire de congestion, donc les grosses connexions sont, en moyenne sur leur durée, peu impactées par la priorisation des petits transferts.

A titre d'illustration, nous avons représenté sur la Figure 3.8 la répartition des transferts (en noir) et la répartition du trafic (en rouge) en fonction de la taille des transferts, en octets. La mesure a été effectuée sur un point de présence ADSL, sur tous les transferts d'une journée, les transferts s'étalant sur plusieurs jours étant donc tronqués, les gros transferts sont donc sous-estimés. Nous constatons tout de même comme il a déjà souvent été observé, qu'un très grand nombre de petits transferts représente une part minime du trafic, et parallèlement qu'un nombre réduit de gros transferts représente une part importante du trafic.

Pour la mise en oeuvre de la priorisation des petits transferts, un des problèmes est de reconnaître au vol ces transferts alors qu'ils sont en cours de transmission et que par conséquent les équipements traversés ne peuvent pas en connaître la taille finale. Nous proposons dans [**?**] un mécanisme de priorisation permettant de résoudre ce problème, sans avoir à maintenir d'état par connexion dans les équipements.

Le système résultant peut être approché par une file PS+PS, i.e. une file PS prioritaire pour les petits flux, et une seconde file PS non-prioritaire pour les autres flux. L'analyse de ce système, présentée dans [**?**], montre que l'asymptote du temps de réponse en fonction de la taille des transferts a la même pente pour les systèmes PS et PS+PS.

Par conséquent l'écart en débit tend vers 0 pour les grandes connexions, elles sont donc peu impactées par la priorisation des petits flux. En effet, donner la priorité aux petits flux diminue le nombre de connexions actives en cas de congestion, donc les connexions restantes peuvent être servies plus rapidement. Les grosses connexions, de type peer-to-peer, sont d'autant moins impactées par des congestions transitoires qu'elle peuvent débuter avant et se terminer après la congestion.

3.2.3 Application au dimensionnement

La mise en oeuvre de mécanismes de priorisation des flux en fonction de leur taille, tels que celui présenté par exemple dans [**?**], permet ainsi d'optimiser les performances des utilisateurs en cas de surcharge, plus précisément d'accroître la durée des périodes de surcharge que peut absorber le réseau avant que les performances des petites connexions soient vraiment dégradées.

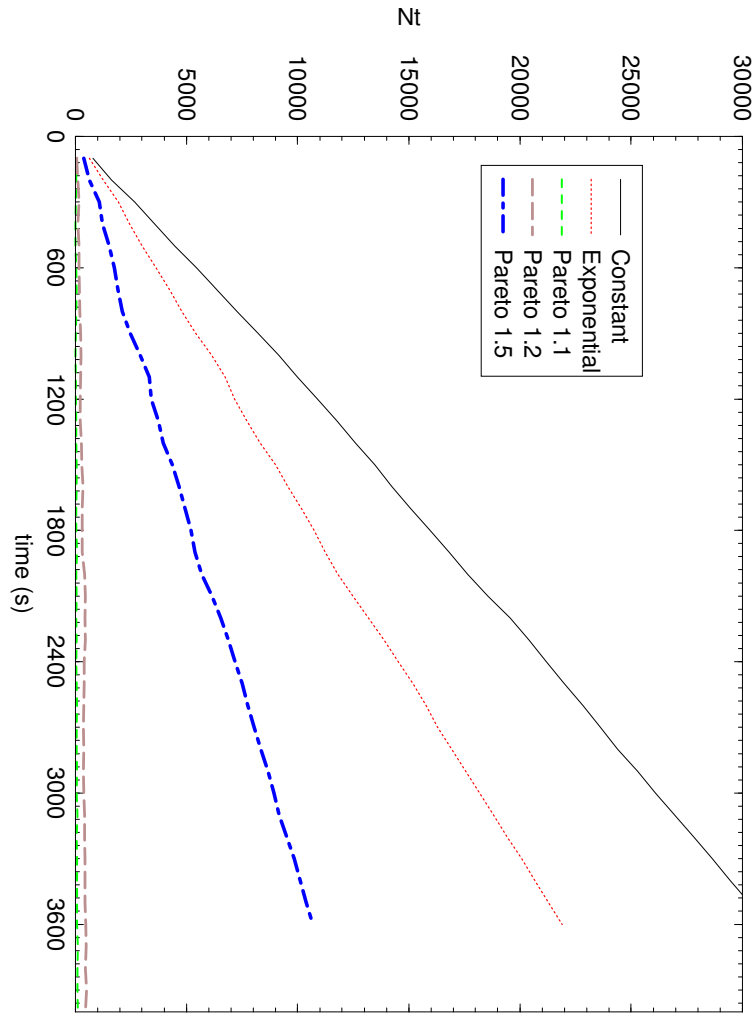


FIG. 3.5 – Nombre de connexions actives en fonction du délai écoulé depuis le début de la surcharge et pour diverses distributions de tailles des transferts.

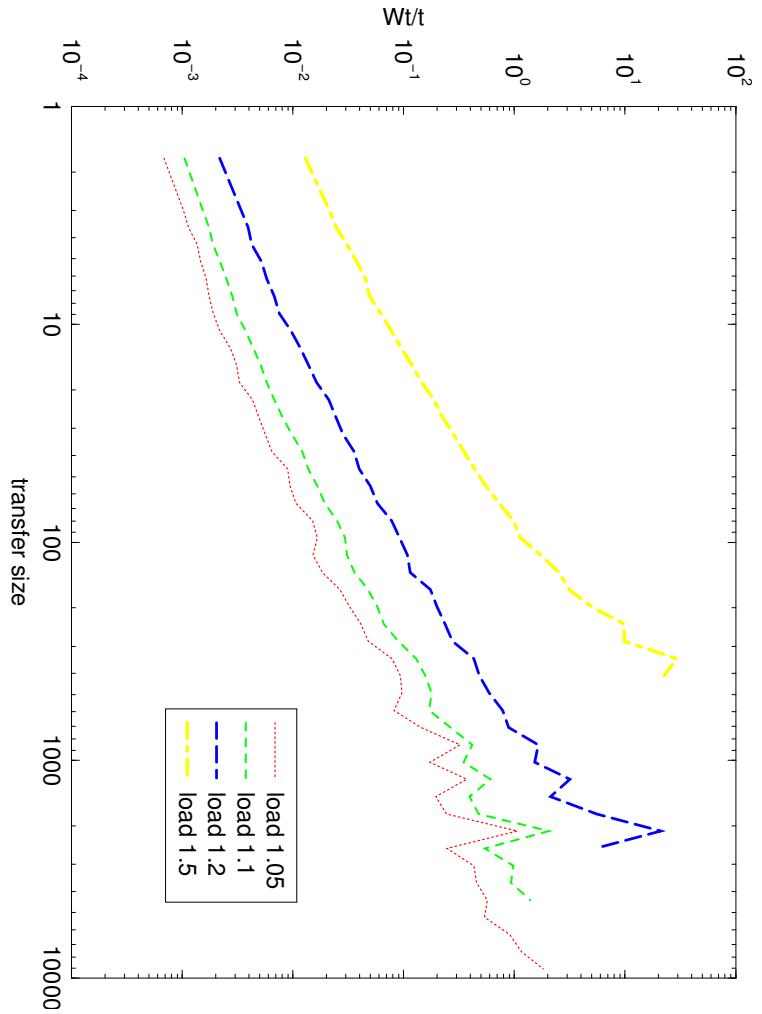


FIG. 3.6 – Comparaison des temps de transfert moyennés sur 10 mn, et divisés par la date d'arrivée des connexions, en fonction du temps de transfert, et pour des tailles de transfert suivant une loi de Pareto de paramètre 1.2 et diverses charges offertes.

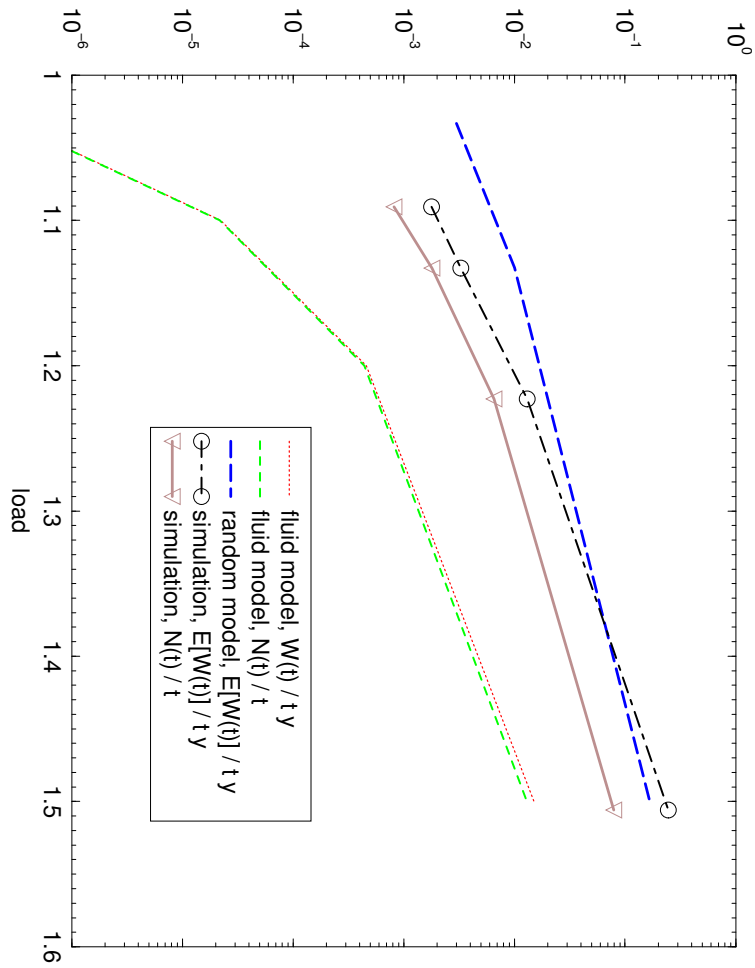


FIG. 3.7 – Comparaison des estimations de a et g pour une charge de 1.2 et des tailles de transfert suivant une loi de Pareto de paramètre 1.2.

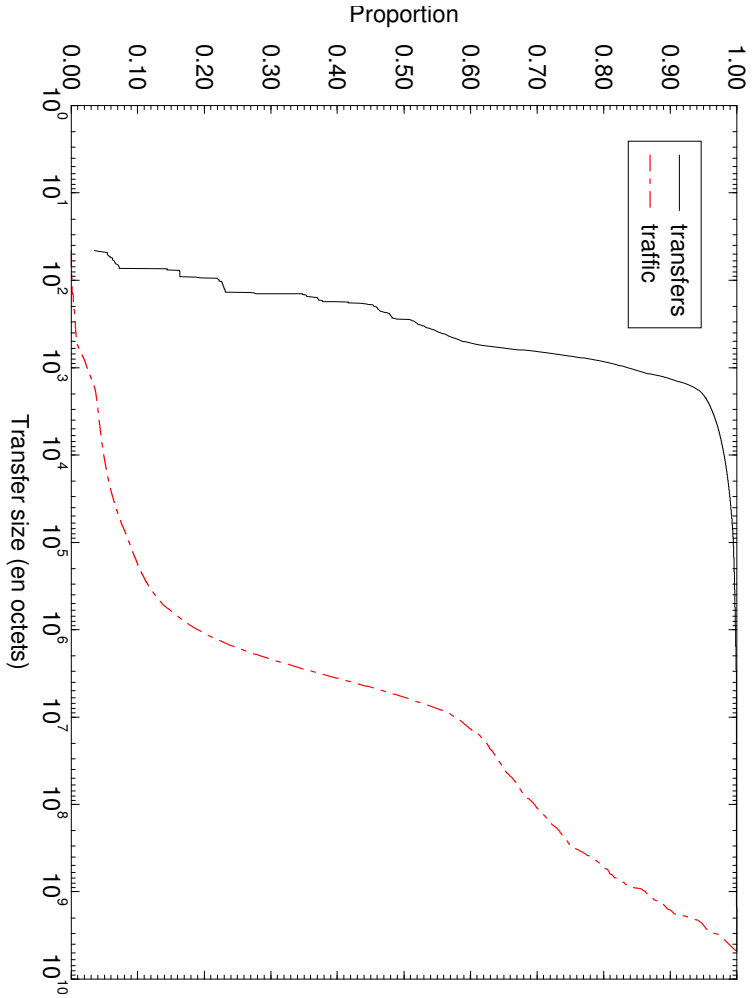


FIG. 3.8 – Répartition des transferts et du trafic associé en fonction de la taille des transferts.

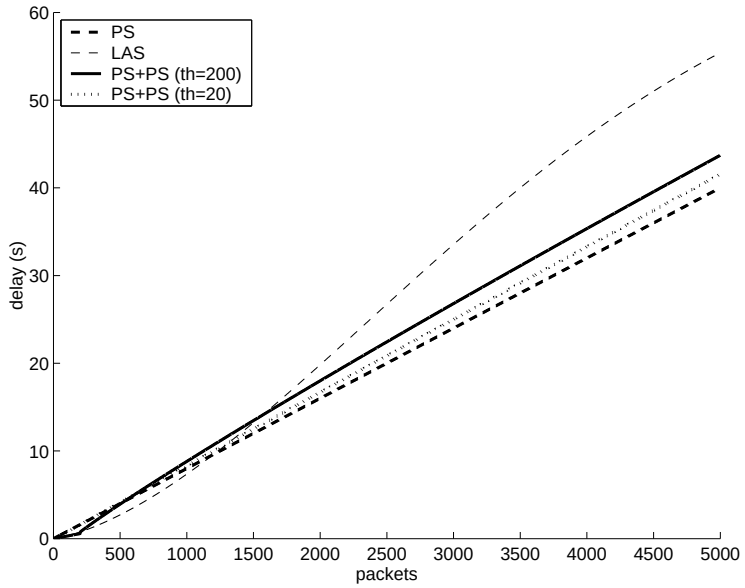


FIG. 3.9 – Comparaison du temps de réponse en fonction de la taille des transferts pour PS, PS+PS et LAS (Least-Attained Service)

Chapitre 4

Estimation de matrices de trafic

4.1 Introduction

La matrice de trafic est la représentation du volume de données échangées entre tout les couples (paires Origine-Destination) possibles de nœuds dans un réseau de communication. Dans le contexte de l'Internet, les nœuds peuvent être des routeurs, ou un ensemble de routeurs situés physiquement au même endroit comme les Points De Présence.

Il est relativement compliqué à l'heure actuelle de procéder à une mesure directe de cette matrice de trafic. Pourtant la matrice de trafic d'un opérateur est quasiment indispensable pour les ingénieurs. C'est pourquoi beaucoup d'opérateurs tentent à l'heure actuelle de trouver une méthodologie pour extraire cette matrice de trafic des informations directement disponibles.

La matrice de trafic apporte une meilleure compréhension de la dynamique interne du réseau d'un opérateur. Elle permet notamment de répondre simplement à des questions du type : "Que se passe-t-il si ... ?". Il est ainsi possible d'utiliser la matrice de trafic pour planifier l'évolution de la topologie logique d'un réseau de grand opérateur, de planifier la capacité nécessaire pour chaque lien, ou encore de s'approcher un peu plus près de la matrice de routage optimale.

La méthode la plus simple serait de faire mesurer cette matrice par chaque nœud du réseau. Malheureusement de telles approches (tel NetFlow de Cisco) soufrent de quelques problèmes techniques qui freinent encore leur déploiement à grande échelle. Comme ce type de solution ne sera pas disponible avant quelques années, il est important pour les fournisseurs d'accès à Internet de trouver une méthodologie de mesure applicable dès maintenant. Pour cela l'ensemble de la communauté de recherche à développé des outils d'inférence statistique.

La relation entre la matrice de trafic, la table de routage et le volume de données transitant sur chaque lien peut être représentée sous forme d'un système d'équations linéaires : $Y = AX$ où Y est la matrice du volume de données transitant sur chaque lien, X le volume de données entre chaque paire Origine-Destination (paire OD) et A, la matrice de routage avec a_{ij} (l'élément de A à la i^{e} ligne j^{e} colonne) la fraction de la paire OD j passant sur le lien i. Il est courant de pouvoir mesurer directement Y, notamment à l'aide du protocole SNMP, et la matrice de routage A est simple à dériver à l'aide d'un peu de connaissance de la topologie et de différents poids des liens entre les

nœuds. Le problème est alors assez simple, il suffit de retrouver X connaissant A et Y. En d'autres termes il suffit d'inverser la matrice A. Malheureusement la matrice A n'est pas carrée, ni même de rang complet.

Les premières générations d'algorithme utilisaient la méthode de génération de moment, l'inférence baysienne, ou encore la maximisation de la vraisemblance pour estimer X. L'idée principale de ces méthode était d'utiliser les moments du second ordre pour ajouter des contraintes au système. Après avoir ajouté ces contraintes, on peut de manière itérative ou non résoudre le système. Malheureusement, pour ajouter ces contraintes, il faut formuler des hypothèses sur la relation entre la moyenne et la variance. Par exemple dans [37, 34], les auteurs posent comme a priori que la distribution des valeurs de la matrice de trafic suit une distribution de poisson. Comme la moyenne et la variance d'une distribution de poisson sont identiques, l'estimation de la variance des paires OD permet d'estimer la moyenne. D'autres distributions peuvent être utilisées comme dans [7] où la distribution n'est plus poisson mais gaussienne avec une relation de puissance entre la moyenne et la variance.

Il a été montré dans [19] que toutes ces méthodes sont très sensibles au point de départ de la procédure d'estimation. Cette démonstration à donné naissance à une nouvelle génération de techniques [19, 38, 39] qui proposent diverses techniques pour trouver un bon point de départ.

En pratique toutes ces estimations souffrent d'importants problèmes de précision. Il est en effet impossible de donner une précision lors de la mesure, et les écarts constatés lors de simulation sont très importants, de l'ordre de 10 à 25 % avec quelques paires OD dépassant 100 % d'erreur relative. Il nous semble un peu utopiste de vouloir, sans apporter plus d'information, faire descendre de manière significative ces taux d'erreur. De plus certains fournisseurs d'accès à l'Internet ont signalé leur volonté de ne pas utiliser les techniques d'estimation de matrice de trafic si le *taux d'erreur moyen* ne descendait pas en dessous de la barrière des 10 %.

La difficulté principale rencontré est la diversité des comportements agrégés dans la matrice de trafic. Les modèles utilisés ne permettent pas de capturer toute la diversité des utilisateurs. C'est pourquoi tout modèle à un impact relativement conséquent sur la précision des estimations. Pour développer des modèles réalistes, nous avons choisi de collecter des données NetFlow d'un grand opérateur pour ensuite étudier les principales caractéristiques du trafic. A partir de ces observations, nous avons trouvé deux composants importants, la variation journalière du trafic et un processus de fluctuation. Nous avons aussi tester l'hypothèse "loi de puissance" entre la moyenne et la variance. Cette hypothèse ne nous a pas complètement convaincu, c'est pourquoi nous avons développé un modèle sans a priori sur la relation entre la moyenne et la variance.

En plus d'utiliser des modèles provenant de mesures, nous pensons qu'il faut absolument injecter de l'information dans le système pour ajouter des contraintes au problème. Nous avons exploré dans ce travail deux nouvelles méthodes d'estimation de la matrice de trafic toutes deux utilisant des changements dans la matrice de routage pour injecter de l'information. Nous avons ainsi des heuristiques pour étudier le premier et second moment indépendamment. En couplant ces approches, nous pouvons étudier la variation journalière et le processus de fluctuation sans utiliser de modèle contraignant.

Notre méthode repose sur un ensemble de mécanismes qui combinés donnent une procédure complète d'estimation de matrice de trafic. L'idée principale est d'utiliser une technique simple d'inférence statistique couplée à quelques étapes supplémentaires, chaque étape étant relativement

indépendante des autres. Il est ainsi possible de changer la méthode d'une des étapes pour améliorer l'ensemble de l'estimation.

L'idée principale de notre méthode est d'ajouter à des techniques d'inférence statistique standard quelques étapes pour améliorer cette inférence. En particulier, nous avons ajouté un estimateur de la variance et un algorithme, proposé dans [23], pour signaler quels changement des poids des liens seront utiles pour augmenter le rang de la matrice A. Nous avons prouvé dans [33] que, dans l'hypothèse non restrictive de routage selon le chemin de moindre coût et des coûts strictement positif, l'estimation de la matrice de covariance ne nécessite aucun changements dans la matrice de routage. Ce résultat est très intéressant car il permet d'être utilisé pour améliorer l'estimation de la moyenne, et ce sans aucun a priori sur la relation entre moyenne et variance. Il est aussi possible d'utiliser la variance pour identifier les flots les plus larges. Or nous avons observé que 30 % des flots représentent plus de 95 % du trafic. Il est alors possible de concentrer la méthode de changement des poids des liens pour se focaliser uniquement sur les paires OD les plus importantes.

4.2 Formalisation du problème

Soit $\mathcal{V} = \{1, \ldots, V\}$ un ensemble de V nœuds. Ces nœuds représentent l'ensemble des routeurs d'un réseau à étudier, ou un ensemble de routeurs physiquement localisés au sein d'un même PDP. Les différents types de matrices de trafic sont décrites dans [2].

Soit un ensemble de L liens directionnels avec $\mathcal{L} \in \mathcal{V} \times \mathcal{V}$, représentant la topologie, supposée connue, du réseau. Nous considérons aussi un ensemble de K ensembles de mesures disjoints. Ces ensembles de mesures serons appelés **images** . Entre chaque image , le routage est modifié, ce qui a pour effet de modifier le chemin suivi par quelques paires OD. Pour changer le routage, nous utilisons la méthode de [23]. Il faut noter que nous changeons 1 ou 2 poids de lien, et donc que nous ne nous éloignons pas trop de l'état stable.

Pendant une image , nous collectons N_s **échantilons** du volume de données transitant sur chacun des liens. Le protocole couramment utilisé pour récolter ces échantilons étant SNMP, chaque image dure donc $N_s \times 5$ minutes. Nous choisirons par la suite un nombre d'échantillon par image constant, mais cette contrainte peut peut être aisément relaxée.

Considérons la paire OD $p = (v_1, v_2)$, et notons $X_p(n, k)$ le volume d'information échangé par la paire OD p pendant l' image k à l'instant n. Si $\mathcal{P}$ est l'ensemble des paires OD, alors card$(\mathcal{P}) = V^2 - V$. Les paires OD sont ordonnées de manière à former le vecteur colonne $X(n, k)$ à partir des $X_p(n, k)$. De même, à partir de $l \in \mathcal{L}$, on peut former le vecteur colonne $Y(n, k)$ dont les éléments sont les $Y_l(n, k)$. Définissons $A_{l,p}(k, n)$ comme étant la proportion du trafic traversant le lien l imputable à la paire OD p au cours de l'image k à l'instant n. Il faut noter que le routage étant constant lors d'une image , nous avons : $A_{l,p}(k, 1) = A_{l,p}(k, 2) = \ldots = A_{l,p}(k, N_s) = A_{l,p}(k)$.

Nous pouvons alors écrire l'équation : $Y_l(n, k) = \sum_{p \in \mathcal{P}} A_{l,p}(n) X_p(n, k)$ ou encore sous forme matricielle :

$$Y(n, k) = A(k)X(n, k) \tag{4.1}$$

Avec $n \in \{0, \ldots, N_s - 1\}$ et $k \in \{0, \ldots, K - 1\}$

Le problème classique d'estimation de matrice de trafic est de résoudre l'équation 4.1 sans changer le routage (ie : $A(1) = A(2) = \ldots = A(K)$), et en considérant $X(n, k)$ comme un échantillon

d'un processus stationnaire. Ce problème n'est pas simple car le rang de la matrice A est au plus L avec $L \ll P$ ie, pour chaque k, le système d'équation est largement sous-contraint. L'idée que nous allons présenter ici est l'utilisation possible des différentes matrices de routages que l'on peut voir au cours d'une expérience.

L'objectif que nous suivrons est l'estimation du premier et du second moment de la matrice de trafic à l'aide de la méthode proposée dans [23]. Nous allons présenter dans la partie suivante les premiers résultats de collecte de toutes les paires OD d'un réseau commercial pour ensuite proposer deux méthode d'estimation de la matrice de trafic.

4.3 Dynamique du trafic

Dans cette partie, nous allons étudier la méthode que nous avons utilisée pour mesurer la matrice de trafic du réseau cœur européen de Sprint. Nous avons utilisé une version de NetFlow appelée *sampled NetFlow* qui collecte de manière déterministe pour chaque flot un paquet sur 250. Comme la définition des flots selon Cisco est le classique 5-uplet, nous avons pu, à l'aide des des tables BGP, et des informations de topologie, déterminer le point d'entrée et le point de sortie de chacun des flots. Le trafic est ensuite agrégé par lien pour créer la matrice de trafic de type PDP à PDP. Nous avons ainsi collecté environ un mois de données.

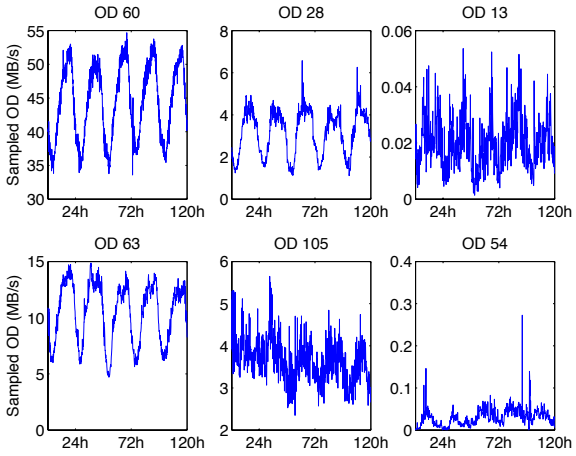


FIG. 4.1 – Quelques exemples de paires OD.

La figure 4.1 montre l'évolution temporelle sur 5 jours de 6 paires OD représentatives des 132 paires que nous avons pu observer sur le réseau européen. Les paires OD de gauche sont parmi les plus gros débits. Ces paires exhibent un comportement diurne relativement régulier et ne semblent pas poser trop de problème à estimer. Tandis que pour les paires moyennes (au centre), il y a essentiellement deux comportements, l'un comme la paire 28 qui présente un comportement diurne stable et l'autre comme la paire 105 qui ressemble beaucoup plus à un bruit blanc qu'à un phénomène interprétable. Les petites paires OD (à droite) présentent presque toutes un comportement quasiment imprévisible.

Nous avons pu observer que les flots les plus petits présentaientt ce genre de comportements erratiques, de plus nous avons aussi observé que plus de 95 % du trafic est inclus dans seulement 30 % des paires OD. Les paires OD correspondant à la partie la plus importante du trafic total observé ont un comportement relativement stable. De plus la différence entre les gros flots et les petits flots sont de plusieurs ordres de grandeur (50 Mo/s contre 100 o/s). Cette différence nous a permis de décider de nous préoccuper de l'estimation des plus grosses paires OD, représentant tout de même 95 % du trafic total. C'est pourquoi dans quasiment tout les graphiques qui vont suivre, nous ne nous intéresserons qu'aux paires les plus importantes.

4.4 Méthodes

Avant de détailler la procédure d'inférence de la matrice de trafic, nous allons décrire l'ensemble de des deux méthodes. Nous allons proposer dans ce rapport deux méthodes. La première méthode est composée de 4 étapes :

1. Collecter les données SNMP et estimer la variance des flots.
2. Utiliser l'algorithme de [23] pour trouver la séquence d'images nécessaire.
3. Appliquer les changements dans les poids des liens tout en continuant la collecte de données SNMP.
4. Estimer, à partir des différentes matrices de routage, de l'estimation de la variances des paires OD, et des données SNMP collectése, la matrice de trafic. Pour cela, nous pouvons utiliser des techniques d'inférence classiques comme la *pseudo inverse* ou encore la formule de *gauss markov*.

Cette méthode permet d'estimer *toutes les paires Origine-Destination* de la matrice de trafic. Dans cette méthode, l'étape n^o 1 n'est pas utile dans le cas de l'utilisation de la pseudo-inverse.

La seconde méthode consiste à tout d'abord identifier les paires OD les plus importantes pour ensuite se focaliser uniquement sur leur estimation, et ainsi réduire considérablement le nombre d'Images nécessaires. Pour ce faire, nous utilisons le fait que l'ordre des paires OD rangées suivant leur moyenne ou leur variance est a peu près le même. Cela est évident dans le cas de l'utilisation d'un modèle en loi de puissance pour le rapport entre la variance et la moyenne des paires OD. Le modèle ici est beaucoup moins contraignant que celui de la loi de puissance, surtout qu'il n'est pas indispensable d'avoir une stricte égalité entre les deux ordres, nous pouvons toujours estimer un peu plus de paires OD que nécessaire. Il faut juste que les grosses paires OD aient une grosse variance. Nous avons vérifié empiriquement, à partir des données disponibles, que cette condition est respectée.

La méthode 2 peut donc être résumée comme suis :

1. Collecter les données SNMP et estimer la variance des paires OD.
2. Sélectionner, à l'aide de l'estimation de la variance, les paires OD qui doivent être estimées.
3. Utiliser l'algorithme de [23] pour trouver la séquence d'images nécessaire à l'estimation du sous ensemble de paires OD.
4. Appliquer les changements dans les poids des liens tout en continuant la collecte de données SNMP.
5. Estimer, à partir de toutes ces informations, la matrice de trafic en utilisant des techniques d'inférence classiques comme la *pseudo inverse* ou encore la formule de *gauss markov*.

En comparant ces deux méthodes, nous avons un choix à faire. D'un côté nous avons une bonne estimation, mais au prix d'un grand nombre de changements de la matrice de routage tandis que dans l'autre, les changements sont moins importants, mais seulement une partie de la matrice de trafic est estimée. La seconde méthode permet à l'opérateur de spécifier un nombre maximal de changements dans la matrice de routage. La méthode n° 2 permet alors d'identifier et d'estimer les paires OD les plus importantes et de laisser l'estimation des autres paires à des méthodes d'inférence statistiques plus classiques.

4.5 Modèle

4.5.1 Scénario stationnaire

Dans ce paragraphe, nous supposons que $\{X(n,k) \forall k \in \{0,\ldots,K-1\}$ et $n \in \{0,\ldots,N_s-1\}\}$ est la réalisation d'un processus stationnaire à temps discret. En particulier, nous pouvons représenter les paires OD sous la forme :

$$X(k,n) = x + W(k,n) \tag{4.2}$$

Où x est un vecteur colonne déterministe représentant la valeur moyenne de chaque paire OD et $W(k,n)$ représente le bruit, avec $E[W] = 0$. Dans la pratique, il se peut que certaines données manquent à cause d'erreurs humaines ou de pannes diverses. Toutes ces erreurs se retrouvent dans ce qui est appelé le bruit (représenté par $W(k,n)$). De plus, pour l' image k, quelques équations du système 4.1 peuvent être redondantes mais pour des raisons de simplicité de notation, nous garderons toutes les équations. Nous avons donc :

$$Y(n,k) = A(k)X(n,k) \tag{4.3}$$

Avec $n \in \{0,\ldots,N_s-1\}$ et $k \in \{0,\ldots,K-1\}$.

Par définition le rang$\big(A(k)\big) = M_k$ avec $M_k \leqslant L < P$. Il est donc impossible de retrouver X à partir de Y, même s'il n'y a pas eu de problème (pertes de données) lors de la mesure.

Si nous changeons le routage entre deux images , et en définissant les matrices suivantes :

$$Y = \begin{bmatrix} Y(0,0) \\ \vdots \\ Y(0,N_s-1) \\ \vdots \\ Y(K-1,N_s-1) \end{bmatrix}, \; A = \begin{bmatrix} A(0) \\ \vdots \\ A(0) \\ \vdots \\ A(K-1) \end{bmatrix}$$

$$W = \begin{bmatrix} W(0,0) \\ \vdots \\ W(0,N_s) \\ \vdots \\ W(K-1,N_s-1) \end{bmatrix}, \; C = \begin{bmatrix} A(0) & 0 & \ldots & 0 \\ 0 & A(0) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & A(K-1) \end{bmatrix}$$

Dans la nouvelle matrice A, les blocs $A(k)$ sont répétés N_s fois, car la matrice de routage est supposée stable lors des différentes images . Les dimensions des nouvelles matrices sont donc :

– $Y : 1 \times LN_sK$
– $A : LN_s \times KP$
– $W : 1 \times KN_sP$
– $C : LN_sK \times KN_sP$

Avec ces notations et en combinant l'équation 4.2 et 4.3, nous obtenons :

$$Y = Ax + CW \tag{4.4}$$

L'objectif est d'estimer x à partir des $A(k)$ et des $Y(n,k)$. Nous pouvons alors utiliser x comme estimation de $X(n,k)$. Tout a long de ce rapport, nous utiliserons $\hat{x}$ comme étant l'estimation de x. Nous supposerons aussi que nous avons appliqué les bons changements dans le routage pour que la matrice A soit de rang complet.

L'approche la plus simple pour estimer x est de minimiser la norme euclidienne de $Y - AZ$, ie :

$$\hat{x} = \arg\min_z \{(Y - Az)^T (Y - Az)\} . \tag{4.5}$$

4.5.2 Estimation de la matrice de trafic connaissant la matrice de covariance

Dans cette partie, nous allons regarder comment des estimateurs simples comme la *pseudo-inverse* et les équations de *Gauss-Markov*, peuvent être appliqués à notre problème pour résoudre l'équation 4.5.

Pour cela définissons B la matrice de covariance du bruit, ie :

$$B = E[WW^T] . \tag{4.6}$$

Pour des raisons de simplicité des notations, nous utiliserons t avec $t \in \{0, \ldots, T-1\}$ et $T = KN_S$ pour dénoter le temps. T est le nombre total d'échantillons que nous avons collectés au cours de notre expérience. Notez qu'il existe une bijection entre le temps noté sous la forme t et celui noté par (k,n). La matrice B peut alors être écrite sous la forme :

$$B = \begin{bmatrix} R(0) & R(1) & \ldots & R(T-1) \\ R(1) & R(0) & \ldots & R(T-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(T-1) & R(T-2) & \ldots & R(0) \end{bmatrix} , \tag{4.7}$$

Où $R(t)$ est la matrice de taille $P \times P$définie par :

$$R(t) = [r_p(t)] = \text{diag}(E[W_p(\tau) W_p(\tau + t)]). \tag{4.8}$$

Avec cette définition, la matrice de covariance de Y est CBC^T. Dans un premier temps, nous supposerons la matrice B comme connue. Nous verrons comment l'estimer dans la section 4.5.3.

Le meilleur estimateur de x conaissant Y et A, au sens de la minimisation de $E[(Y - Az)^T (Y - Az)]$ suivant z est connu sous le nom d'Estimation des Moindre Carrés. En utilisant le théorème de Gauss-Markov, on obtient :

$$\hat{x} = \hat{x}(Y, B) = \left(A^T (CBC^T)^{-1} A\right)^{-1} A^T (CBC^T)^{-1} Y . \tag{4.9}$$

Il faut noter que l'équation 4.9 se réduit à une pseudo-inverse dans le cas où B est la matrice identité. Un des avantages d'utiliser la pseudo-inverse est qu'aucune connaissance des statistiques de W est nécessaire. Bien évidemment, en ayant des idées sur les statistiques de W et en appliquant la formule de Gauss-Markov, l'estimation de x est meilleure. Notamment, il n'est pas évident que la variance de chacune des composantes de Y soit identique. Par exemple, si une des composantes de Y à une faible variance, la composante correspondante de $A\hat{x}$ doit être fortement pondérée pour donner une meilleure estimation de x.

4.5.3 Estimation de la matrice de covariance

Intéressons nous à l'étude de la matrice de covariance des paires OD. Comme vu dans la partie précédente, cette estimation peut-être utilisée pour trouver une meilleure estimation de x en utilisant la formule de Gauss-Markov (equation 4.9).

Nous tenons à mettre en exergue deux points importants de notre méthode d'estimation de la matrice de covariance. Premièrement, elle est totalement indépendante de l'estimation du premier moment de la matrice de trafic. Dans les méthodes précédentes, les méthodes supposaient une relation entre la variance et la moyenne (type loi de puissance). Ensuite, l'estimation de la matrice de covariance ne nécessite aucun changement de la matrice de routage. Il suffit juste de collecter un grand nombre de mesures. Dans [33], une preuve justifiant le fait qu'aucun changement n'est nécessaire est présentée.

Nous allons réutiliser la notation temporelle introduite dans la section 4.5.2. Définissons la matrice de corrélation de liens $C_Y(t) = E[Y(\tau)Y^T(\tau+t)]$. Pour le couple de liens (l, m), en supposant les paires OD indépendantes, nous pouvons l'écrire sous la forme :

$$E\left[\sum_{i=1}^{P}\sum_{j=1}^{P} A_{l,i}[X_i + W_i(\tau)]A_{m,j}[X_j + W_j(\tau+t)]\right]$$
$$= \sum_{i=1}^{P} A_{l,i}A_{m,i}r_i(t) + \sum_{i=1,j=1,j\neq i}^{P} A_{l,i}A_{m,j}E[X_i]E[X_j].$$

Soit encore, en notation matricielle :

$$C_Y(t) = AR(t)A^T + E[AX]E[AX]^T = AR(t)A^T + E[Y]E[Y]^T. \tag{4.10}$$

Nous pouvons donc estimer $R(t)$ comme :

$$\hat{R}(t) = \arg\min_{Z} ||C_Y(t) - AZA^T - E[Y]E[Y]^T||_2^2.$$

et à l'aide de quelques manipulations simples de matrice, nous pouvons écrire l'équation (4.10) comme $\gamma_y(t) = \Gamma r_w(t)$, où $\gamma_y(t)$ est $C_Y(t) - E[Y]E[Y]^T$ ordonné en vecteur de taille L^2, et Γ la matrice de taille $L^2 \times P$ dont chaque élément est le produit terme à terme de toutes les paires possibles de lignes de A, et enfin $r_w(t)$ est un vecteur de taille P dont les éléments sont les $r_p(t)$.

Il est donc possible, en utilisant la pseudo-inverse, d'estimer $r_w(t)$:

$$\hat{r}_w(t) = (\Gamma^T\Gamma)^{-1}\Gamma^T\gamma_y(t). \tag{4.11}$$

4.5.4 Scénario cyclo stationnaire

Nous allons maintenant présenter un modèle cyclo stationnaire de matrice de trafic. Dans cette partie, nous n'utiliserons plus l'hypothèse que X est composé d'une moyenne et de fluctuations autours de cette moyenne. A la place nous allons modéliser X par un processus cyclo stationnaire :

$$X(t) = x(t) + W(t) \ \ , \forall t \in [0, T-1]. \tag{4.12}$$

Nous supposons que $X(t)$ est cyclo stationnaire de période N. $X(T)$ et $X(T+N)$ ont donc la même distribution de probabilité. De manière plus spécifique, nous étudierons le cas où $x(t)$ est déterministe de période N et $W(t)$ est de moyenne nulle. L'objectif est alors d'estimer $x(t)$ (la matrice de trafic) pour tout les instants t observant $Y(t)$. En décomposant $x(t)$ dans une base de $2N_b + 1$ fonctions génératrices, ie :

$$x(t) = \sum_{h=0}^{2N_b} \theta_h b_h(t) \tag{4.13}$$

Où $\forall k$, θ_h est un vecteur de P éléments (les coefficients) et que $b_h(.)$ est la valeur de la fonction génératrice de période N. En particulier, nous regarderons le cas de l'expansion de Fourrier :

$$b_h(t) = \begin{cases} \cos(2\pi t h/N) & \text{, si } 0 \leq h \leq N_b \\ \sin(2\pi t(h-N_b)/N) & \text{, si } N_b + 1 \leq h \leq 2N_b \end{cases} \tag{4.14}$$

En substituant (4.13) dans (4.12) et (4.3), nous obtenons :

$$\begin{aligned} Y(t) &= A(k)(\sum_{h=0}^{2N_b} \theta_h b_h(t)) + A(k)W(t) \\ &= A'(t)\theta + A(k)W(t), \end{aligned}$$

où nous définissons le vecteur θ de taille $(2N_b+1)P \times 1$ suivant :

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{2N_b} \end{bmatrix}, \tag{4.15}$$

et la matrice $A'(t)$ de taille $L \times (2N_b+1)P$ comme :

$$A'(t) = \begin{bmatrix} A(k)b_0(t) & A(k)b_1(t) & \cdots & A(k)b_{2N_b}(t) \end{bmatrix}. \tag{4.16}$$

Et enfin en redéfinissant la matrice A pour être de dimension $LT \times (2N_b+1)P$:

$$A = \begin{bmatrix} A'(0) \\ A'(1) \\ \vdots \\ A'(T-1) \end{bmatrix}. \tag{4.17}$$

Les matrices C et W étant définies comme auparavant, nous obtenons une équation similaire à l'équation (4.3), soit :

$$Y = A\theta + CW \ . \tag{4.18}$$

Nous pouvons donc utiliser les deux méthodes présentées plus haut pour estimer θ.

4.6 Résultats

Dans cette partie, nous allons nous intéresser à la validation de notre méthode en utilisant des données réelles présentés dans la partie 4.3. Nous commencerons par présenter le premier modèle pour valider l'approche dans le cas où toutes les paires OD sont estimées. Puis nous nous focaliserons sur l'estimation des paires OD générant plus de 95 % du trafic.

4.6.1 Méthode 1 : Estimation de toutes les paires Origine-Destination

Dans cette section, nous allons étudier le cas où nous estimons toutes les paires OD. Nous avons utilisé l'heuristique présentée dans [23] pour générer une séquence d'images (une séquence de changement de la matrice de routage) pour obtenir une matrice de rang complet.

L'algorithme à décidé d'utiliser $K = 24$ images. Dans ces 24 images, 22 impliquaient un changement d'un seul lien à la fois, les deux autres images impliquaient 2 changements de liens.

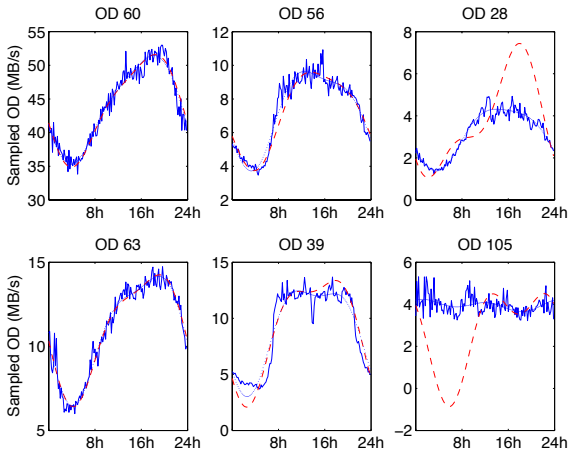
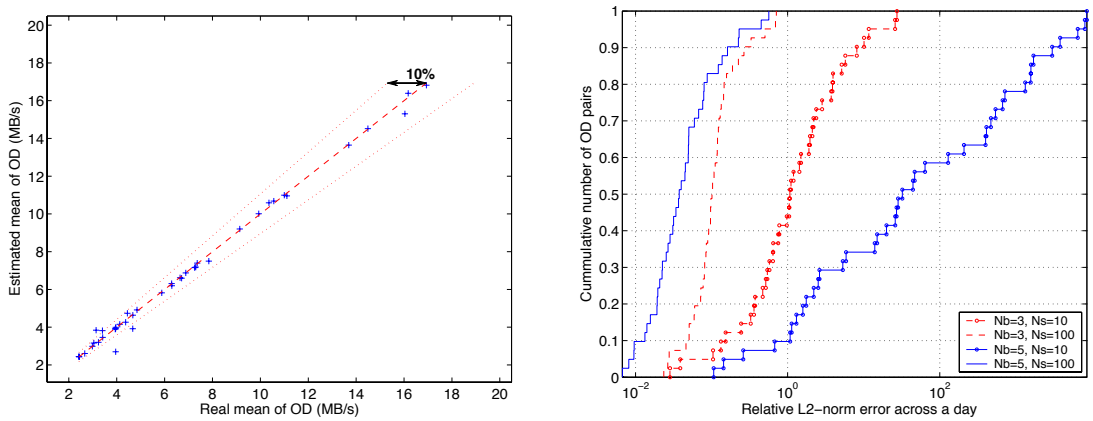


FIG. 4.2 – Exemple d'estimation pour 6 paires Origine-Destination.

Avant de montrer des statistiques plus complètes, nous montrons le résultat pour 6 paires OD. Les paires OD de la figure 4.2 sont parmi les 30 % qui contribuent à 95 % du trafic total. Pour chaque paire, nous avons tracé la paire OD réelle en bleu plein, la paires OD telle que nous essayons de l'estimer (les premiers coefficients de Fourrier) en bleu pointillé et notre estimation en ligne mixte trait point.

Les deux paires de droite (OD 28 et OD 105) sont les pires estimations que nous ayons pour les paires importantes. Il est intéressant de noter que si nous ne sommes pas capables de suivre le comportement temporel de ces paires OD, nous obtenons tout de même une idée le leur valeur moyenne. Les deux paires du milieu (OD 56 et OD 39) sont des estimations typiques, tandis que nous avons montré de bonnes estimations à gauche (OD 60 et OD 63). Au regard de ce graphique, nous nous sommes demandés comment mesurer la qualité de notre estimation. Pour ce faire, nous avons décidé d'étudier deux paramètres, la valeur moyenne, et la différence d'énergie entre notre estimation et le signal à estimer. La valeur moyenne permet de résumer en quelques chiffres notre estimation, mais n'est pas représentative du fait que l'on estime maintenant les paires OD de manière temporelle. Pour la différence d'energie, nous avons utilisé la norme-2 ie : $\|X_p(t) - \hat{X}_p(t)\|_2^2$, elle permet de donner une idée de la différence temporelle des deux signaux, mais les erreurs ne peuvent pas être comparées aux différents articles sur l'estimation de matrice de trafic car aucun article ne considère les paires OD de manière temporelle.



(a) (b)

FIG. 4.3 – (a) Erreur relative dans l'estimation de la valeur moyenne des paires OD représentant 95 % du trafic. (b) Distribution cumulative de la norme-2 de la différence entre les paires OD réelles et leur estimation en fonction du nombre de fonctions de génératrice et du nombre d'échantillons.

Dans la figure 4.3-(a), l'erreur relative maximale dans l'estimation de la valeur moyenne est de 28 %, de plus seulement 4 paires OD ont une erreur relative de plus de 10 %. Ces paires sont toutes de moins de 5 Mo/s. De plus, pour 90 % de ces flots, l'erreur moyenne est en dessous de la barre de 2 %. Les paires OD de moins de 3 Mo/s ne sont pas représentées sur le graphique, mais sont bien sûr prise en compte lors des calculs.

Dans la figure 4.3-(b) nous avons tracé la distribution cumulative de la norme-2 de l'erreur entre entre la paire origine destination réelle et notre estimation. Un des point intéressant est de montrer qu'il doit y avoir une relation entre le nombre d'échantillons et le nombre de fonctions génératrices. On peut voir en effet que si le nombre d'échantillons par image est petit, il vaut mieux utiliser moins de fonctions génératrice, car sinon l'erreur est plus importante. Tandis que si le nombre d'échantillons est grand, utiliser plus de fonctions génératrice permet de mieux « coller » à la courbe. Les valeurs numériques de de la norme-2 ne s'interprètent pas de la même façon que ceux de l'erreur

relative de la moyenne. Par exemple, dans la figure 4.2, la paire OD 28 correspond à une erreur relative en norme-2 de 56 % (44 % pour la paire 105).

A travers ces trois figures, nous avons montré que la méthode n^o 1 donne de bons résultats quand il s'agit d'estimer toutes les paires OD. Nous allons maintenant essayer de réduire le nombre de paires OD à estimer.

4.6.2 Méthode 2 : Identification et Estimation des Paires OD les plus larges

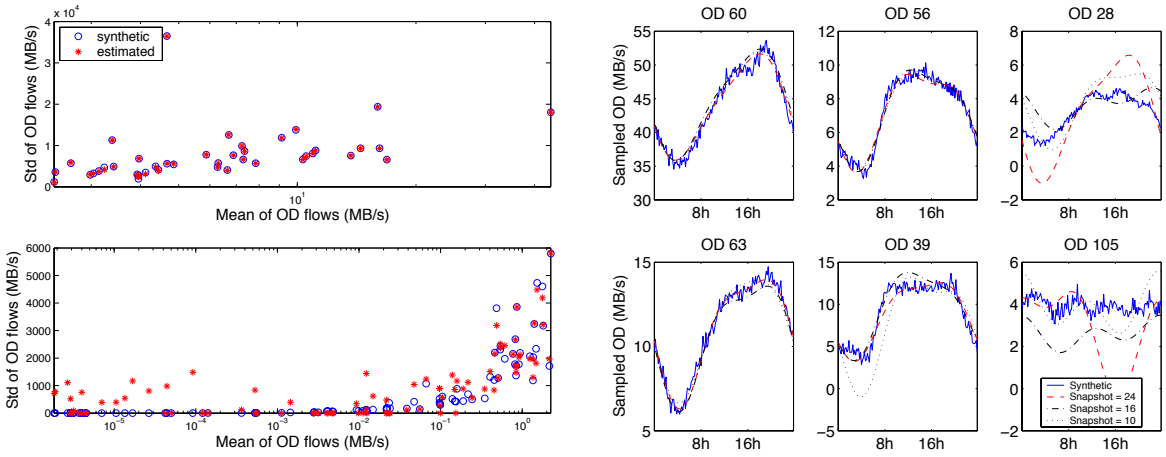
Dans cette section, nous commençons par tenter d'estimer la variance des flots pour ensuite les classer et ainsi estimer uniquement les paires OD les plus larges. Après quelques expériences sur l'estimation de la variance, nous avons trouvé qu'il nous faut un grand nombre d'échantillons, de l'ordre de 25 000 échantillons. Cela représente environ 3 mois de capture de données SNMP. Bien qu'ayant accès à des années de données SNMP, nous ne disposons pas des 3 mois de traces NetFlow. Nous avons donc décidé d'utiliser des traces pseudo synthétiques générées à partir des traces réelles. Pour ce faire, nous avons isolé les basses fréquences du signal et ajouté un bruit gaussien de moyenne 0 et avec une loi de puissance, dont la valeur à été issue de la trace réelle, pour la variance. L'idée étant de rendre ces traces quasiment identiques aux traces réelles que nous avions à notre disposition. Pour se donner une idée, il suffit de comparer la figure 4.4-(b), utilisant des paires OD pseudo synthétiques, à la trace réelle de la figure 4.2.

Pour montrer l'estimation de l'écart type, nous avons tracé sur la figure 4.4-(a) en haut l'estimation des paires OD les plus larges et en dessous les paires OD les moins significatives. Il est possible de voir que les paires contribuant le plus aux trafic sont relativement bien estimées. Nous ne voulons pas connaître la valeur exacte de la valeur de l'écart type des paires OD, mais plutôt une estimation permettant de les classer. Or comme nous pouvons le voir sur la figure 4.4-(a), les paires OD sont plutôt bien ordonnées.

Dans la figure 4.5, nous montrons le nombre d'images nécessaires pour isoler uniquement une partie des paires OD en fonction du nombre des paires OD à estimer. Nous avons aussi tracé en ligne pointillée la valeur moyenne de chaque paire OD pour donner une idée du volume des paires OD. Nous pouvons voir que seul 5 images sont nécessaires pour estimer les paires OD à l'origine de plus de 95 % du trafic. Comme nous ne pouvons pas connaître la qualité de l'estimation, ni même si le classement par écart type est le même qu'à l'aide de la moyenne, nous avons décidé de ne pas descendre au dessous de 10 images.

Nous avons ainsi tracé dans la figure 4.4-(b) le résultat de l'estimation des paires OD en utilisant uniquement 24, 16 ou 10 changements de la matrice de routage. Il faut noter que les paires OD les plus grosses sont toujours bien estimées quel que soit le nombre d'images utilisées (à gauche) tandis que les paires les plus petites ont une erreur importante, surtout pour le cas où le nombre d'images est 10.

Les statistiques résumées dans le tableau 4.1 montrent que l'on a réussi à atteindre l'objectif de rester en dessous des 10 % nécessaires à la mise en place de la méthode d'estimation de matrice de trafic.



(a) (b)

FIG. 4.4 – (a) Estimation de la variance avec 3 mois de données SNMP. (b) Exemple d'estimation de 6 paires OD

OD [%]	Images = 24		Images = 16		Images = 10	
	EM	*PC*	*EM*	*PC*	*EM*	*PC*
100%	3.79	28.04	8.96	57.31	10.88	58.40
95%	2.34	15.48	5.79	31.23	7.99	31.20
90%	1.72	9.30	4.54	22.51	6.78	24.85

TAB. 4.1 – Erreur Moyenne (EM) et Pire Cas (PC) de l'erreur dans la moyenne pour 100 %, 95 % et 90 % des paires Origine Destination contribuant pour 95 % du trafic.

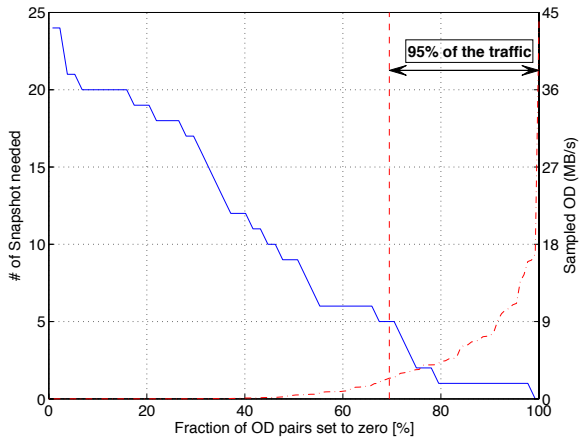


FIG. 4.5 – Nombre d'images nécessaires en fonction du nombre de paires OD à estimer. (b)

Chapitre 5

Perspectives

Pour la dernière année du projet Metropolis, les points suivants seront abordés :

- A l'aide de l'estimation des matrices de trafic, vérifier l'existence de trunks dans le réseau, c'est à dire des paires OD prépondérantes, et examiner comment ces trunks se fusionnent dans le réseau.
- Eclaircir le phénomène des buffles (attaques ?) et celui des libellules (opérations de sauvegarde ?).
- Etude des usages des clients ADSL et rapprochement des résultats avec l'utilisation de distributions aléatoires (cf. Section 2.2).
- Reconstruction d'une matrice de trafic à partir de données NetFlow.

Bibliographie

[1] J. V. L. Beckers, I. Hendrawan, R. E. Kooij, and R. D. van der Mei. Generalized processor sharing performance models for internet access lines. In *Proceedings of 9th IFIP conference on Performance modelling and evaluation of ATM & IP networks*, Budapest, Hungary, 2001.

[2] R. E.K̃ooij Beckers, I.H̃endrawan and
R.D̃. van der Mei. A Taxonomy of IP Traffic Matrices. In *SPIE ITCOM : Scalability and Traffic Control in IP Networks II*, Boston, 2002.

[3] A. Berger and Y. Kogan. Dimensioning bandwidth for elastic traffic in high-speed data networks. *IEEE/ACM Trans. on Networking*, 8(5) :643–654.

[4] Jeff Bilmes. A gentle tutorial on the EM algorithm including gaussian mixtures and baum-welch. Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, 1997.

[5] T. Bonald, P. Olivier, and J. Roberts. Dimensioning high speed ip access networks. In *Proceedings of ITC'18*, Berlin, Germany, Sept. 2003.

[6] T. Bonald, P. Olivier, and J. Roberts. Dimensioning ip access links carrying data traffic. *Soumis aux Annales des Télécommunications*, 2004.

[7] J. Cao, D. Davis, S. Vander Weil, and B. Yu. Time-Varying Network Tomography. Journal of the American Statistical Association, 2000.

[8] Gilles Celeux, Didier Chauveau, and Jean Diebolt. On stochastic versions of the EM algorithm. Technical Report RR-2514, INRIA, 1995.

[9] CISCO. Netflow services and applications, 2002.

[10] J. W. Cohen. The generalized engset formulae. *Philips Telecomm. Review*, 18 :158–170.

[11] J. W. Cohen. The multiple phase service network with generalized processor sharing. *Acta Informatica*, 12 :245–284.

[12] M. Crovella. Performance evaluation with heavy tailed distributions. *Lectures Notes in computer Science 1786*, March 2000.

[13] Michael D. Escobar. Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, 89(425) :268–277, march 1994.

[14] Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430) :577–588, 1995.

[15] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop*, pages 75–80. ACM Press, 2001.

[16] W. Fang and L.Peterson. Inter-as traffic patterns and their implications. In *Proc. IEEE Globecom*, Brazil, December 1999.

[17] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J.W. Roberts. Statistical bandwidth sharing : a study of congestion at flow level. In *Proceedings of ACM SIGCOMM*, 2001.

[18] D. P. Heyman, T. V. Lakshman, and A. L. Neidhardt. A new method for analysing feedback-based protocols with applications to engineering web traffic over the internet. In *Proceedings of ACM SIGMETRICS'97*, 1997.

[19] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and Diot C. Traffic Matrix Estimation : Existing Techniques and New Directions. In *ACM SIGCOMM*, Pittsburgh, USA, 2002.

[20] Metropolis. Métrologie pour l'internet et les services - sous-projet 1 : Rapport d'état de l'art. Technical report, Projet RNRT, Jan. 2003.

[21] P. Muller, A. Erkanli, and M. West. Bayesian curve-fitting using multivariate normal mixtures. *Biometrika*, 83(1) :67–79, 1996.

[22] M. Nabe, M. Murata, and H. Miyahara. Analysis and modeling of world wide web traffic for capacity dimensioning of internet access lines. *Performance Evaluation*, 34 :249–271.

[23] Antonio Nucci, Rene Cruz, Nina Taft, and Christophe Diot. Design of IGP link weight changes for estimation of traffic matrices. In *To appear in IEEE Infocom*, Hong Kong, 2004.

[24] P. Olivier and N. Benameur. Flow level ip traffic characterization. In *Proceedings of ITC'17*, Salvador da Bahia, Brazil, Dec. 2001.

[25] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling tcp reno performance : a simple model and its empirical validation. *IEEE/ACM Trans. on Networking*, 8(2) :133–145.

[26] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot. A pragmatic definition of elephants in internet backbone traffic, 2002.

[27] K. Papagiannaki, N. Taft, and C. Diot. Temporal behavior of network prefixes on internet backbone links. Technical Report TR03-ATL-020527, Sprint, 2002.

[28] S. Sarvotham J. Rexford and K.Shin. Load-sensitive routing of ling-lived flows. In *Proc. ACM SIGCOMM*, september 1999.

[29] A. Riedl, T. Bauschert, M. Perske, and A. Probst. Investigation of the m/g/r processor sharing model for dimensioning of ip access networks with elastic traffic. In *Proceedings of 1st Polish-German Teletraffic Symposium*, Dresden, Germany, 2000.

[30] Christian P. Robert. *The Bayesian Choice*. Springer, 2001.

[31] Kavé Salamatian and Sandrine Vaton. Hidden markov modeling for network communication channels. In *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 92–101. ACM Press, 2001.

[32] R. Serfozo. *Introduction to Stochastic Networks*. Springer Verlag, 1999.

[33] Augustin Soule, Antonio Nucci, Emilio Leonardi, Rene Cruz, and Nina Taft. How to identify and estimate the largest traffic matrix element in a dynamic environement, submitted.

[34] C. Tebaldi and M. West. Bayesian Inference of Network Traffic Using Link Count Data. Journal of the American Statistical Association, 1998.

[35] S. Uhlig and O. Bonaventure. On the cost of using mpls for interdomain traffic. In *Quality of future Internet Services*, Berlin, September 2000.

[36] Iljitsch van Beijnum. *BGP Building Reliable Networks with the Border Gateway Protocol*. O'Reilly, 2002.

[37] Y. Vardi. Estimating Source-Destination Traffic Intensities from Link Data. Journal of the American Statistical Association, 1996.

[38] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greeberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proceedings of the 2003 ACM SIGMETRICS*, 2003.

[39] Yin Zhang, Matthew Roughan, Carsten Lund, and David Donoho. An information-theoretic approach to traffic matrix estimation. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 301–312. ACM Press, 2003.

Deuxième partie

Sous Projet 3

Analyse de mesure de réseaux

METROPOLIS

Métrologie Pour L'Internet

Projet RNRT Exploratoire

Sous Projet 3
Analyse de mesure de réseau

Livrable
Rapport d'avancement

Coordinateur du sous-projet : Philippe Owezarski (LAAS).

Auteurs : LAAS : Nicolas Larrieu, Philippe Owezarski
Eurecom Ernst Biersack, Guillaume Urvoy-Keller,. . .
LIP6 : Augustin Soule

Laboratoire d'automatique et d'analyse des systèmes	LAAS
Institut Eurecom	Eurecom
Laboratoire d'Informatique de Paris 6	LIP6

Chapitre 6

Introduction

L'objectif du sous-projet 3 de METROPOLIS intitulé "Analyse" consiste à préparer des logiciels pour exploiter les traces de trafic produites par les équipements de métrologie. Toutefois, ce rapport ne traitera que de l'analyse des traces passives DAG[1] [23], l'analyse des traces actives faisant partie du rapport davancement du sous-projet 4 [12]. Les logiciels développés dans ce sous-projet vont des outils de base pour la manipulation de traces brutes (et ce afin de les convertir à un format facile à exploiter) jusqu'à des outils de plus haut niveau pour la caractérisation du trafic et dont les résultats sont des bases notamment pour les travaux de modélisation.

Les outils développés dans ce sous-projet et qui vont être décrits dans ce rapport ont donc trait à :

- La mise en forme des traces : en effet, le format DAG (ERF) est difficile à exploiter, en particulier car il est variable. Il a donc fallu développer les scripts nécessaires pour permettre aux sondes DAG de capturer des traces avec les bons formats, c'est à dire incluant les entêtes IP et TCP (mais pas les entêtes applicatifs pour rester dans les limites imposées par la CNIL) et une estampille précise et exploitable. En particulier, il a fallu développer des macros pour manipuler les estampilles dans des grandeurs humainement compréhensibles (secondes, ms, μs, etc.) et pas en nombre de tics d'horloges comme cela est fait par la carte DAG. Cette famille de logiciel inclut également les procédure d'anonymisation d'adresses ;
- L'extraction de données de granularité différentes des traces de trafic microscopiques. En effet, les analyses peuvent porter sur des octets, des paquets ou des flux [26]. Les traces DAG ne considèrent que le niveau paquet (ces traces sont constituées de l'enregistrement de l'entête de tous les paquets qui passent sur le lien supervisé, avec pour chacun d'eux une estampille associée a leur date de passage). Ces outils permettent donc de générer soit des traces en nombres d'octets ou de bits, soit des traces de flux ;
- La caractérisation simple du trafic : cela concerne l'étude de nombreux paramètres comme le débit au cours du temps, la détection de pertes, le calcul des distributions de tailles de flux, taille de paquets, durées des flux, etc.
- La caractérisation avancée du trafic : En effet, un certains nombre de caractéristiques qui sont apparues dans le trafic lors des premières phases d'analyse n'ont pu être décrites par les cri-

[1] Quelques résultats obtenus avec les sondes QoSMOS, notamment pour la classification applicative des paquets seront également présentés.

tères simples de l'item précédent. Aussi, des outils de caractérisation plus avancés, notamment des outils pour calculer les moments statistiques d'ordre 2 comme l'auto-corrélation, ou évaluer les phénomènes de dépendance dans le trafic ont été développés ou intégrés. A noter que certains de ces outils sont issus de travaux de la communauté traitement du signal.

Naturellement, cette tâche de développement d'outils d'analyse n'est pas encore achevée puisque le sous-projet analyse perdure jusqu'à la fin du projet METROPOLIS. Toutefois, tous ces outils ont été intégrés dans une boite à outils d'analyse des traces. Cette boite à outils a été baptisée ZOO. Ce nom peut surprendre de prime abord, mais nous espérons dans ce rapport convaincre le lecteur de la pertinence de ce nom. En effet, nous verrons dans la suite que souvent la caractérisation du trafic ne peut se faire de façon globale, et qu'il faut passer par des sous-classes du trafic. Deux des plus célèbres classes de flux portent les noms de "souris" et "éléphants", suivant que ces flux comportent respectivement un petit ou un grand nombre de paquets. En s'inscrivant dans cette même démarche, nous avons donc défini de nouvelles classes de flux qui présentent des caractéristiques communes et que nous avons choisi de désigner par des noms d'animaux. Ainsi, ce rapport est plein de tortues, libellules, buffalos, etc., et ZOO est donc un outil qui permet d'analyser très simplement chacun de ces types de flux et leur trafic associé.

Ce rapport va donc décrire tous ces outils d'analyse développés au cours des deux premières années du projet METROPOLIS ainsi que la boite à outils ZOO. Ces outils seront présentés dans l'ordre de l'énumération précédente. Naturellement, des résultats d'analyse de traces capturées sur les réseaux de collecte de Renater seront présentés pour illustrer chacun de ces outils logiciels. Enfin, ce rapport montrera aussi des méthodes permettant d'expliquer le lien qui existe entre certains phénomènes observés sur le trafic et des mécanismes protocolaires, et surtout de montrer le lien qui existe entre ces phénomènes et des caractéristiques statistiques du trafic. L'exemple qui sera décrit dans la partie 5 de ce rapport part du phénomène oscillatoire qui a été observé sur toutes les traces de tous les liens que nous avons étudiées, et au travers d'un certain nombre d'expériences met en évidence les raisons d'un tel phénomène. Il sera ainsi montré que les mécanismes de contrôle de congestion ne sont pas adaptés à la transmission des flux éléphants, et que cela se traduit justement par une augmentation du phénomène oscillatoire du trafic avec tout ce que ça a de néfaste pour la performance du réseau. D'autre part, cette même étude nous permettra aussi de mettre en évidence que l'une des caractéristiques du trafic – la dépendance longue – qui a été mise en évidence systématiquement est complètement liée aux oscillations du trafic (et donc à TCP), et donc que la dépendance est un bon moyen de qualifier et quantifier le caractère oscillatoire du trafic. Enfin, la partie 6 conclura ce rapport.

Chapitre 7

Mise en forme des traces

Cette section est dédiée à l'ensemble des réglages et des calibrages qui ont été nécessaires pour exploiter les captures de trafic réalisées sur la plate-forme de mesures passives déployée dans le cadre du projet Métropolis. En particulier, nous traiterons des problèmes de la taille des enregistrements, de l'estampillage temporel des traces ainsi que de leur anonymisation, problèmes que nous avons rencontrés lors des phases de déploiements matériel et logiciel du projet Métropolis.

7.1 Taille des enregistrements

La taille des enregistrements pour les captures microscopiques passives est un problème crucial dans le projet Métropolis. En effet, cette taille conditionne le degré de complexité des analyses qui pourront être menées a posteriori sur une trace. En effet, une capture n'enregistrant que l'en-tête TCP/IP des paquets circulant sur le réseau ou une capture permettant de stocker l'ensemble des données contenues dans un paquet TCP/IP n'offriront pas les mêmes possibilités en terme de caractérisation. Par exemple, prenons le cas d'un ingénieur réseau qui souhaite analyser le trafic qui passe sur son réseau. Une capture au niveau TCP/IP lui apportera par l'intermédiaire des numéros de port des informations quantitatives sur les différentes applications qui transitent sur son réseau. Mais cette information sera de plus en plus incomplète car de plus en plus d'applications, notamment pair à pair (P2P), n'utilisent pas un numéro de port fixé à l'avance pour échanger leurs informations. Ainsi, leur détection nécessite de faire une reconnaissance syntaxique de ces flux. Cette tâche nécessite de pouvoir avoir accès au contenu applicatif de chacun des paquets capturés. Malheureusement, le stockage de la totalité de l'en-tête applicatif de chaque paquet ne nous est pas autorisé par la CNIL. Pour ces raisons, nous utiliserons le logiciel de classification applicatif QoSMOS [31] pour réaliser ce type de caractérisation car il permet d'analyser à la volée le contenu de chaque paquet et de ne stocker qu'une partie agrégée de ces informations (par exemple la répartition en volume des données circulant sur le réseau classées par type d'application que nous présenterons dans la section 8).

Pour garantir un degré d'analyse suffisant, nous avons fixé la valeur minimale d'un paquet capturé à 70 octets (16 octets pour l'en-tête DAG + 14 octets pour l'en-tête Ethernet (sans les 4 octets du CRC) + 20 octets pour l'en-tête IP + 20 octets pour l'en-tête TCP (hors option)).

Les cartes DAG utilisées pour réaliser les captures passives proposent plusieurs formats de stockage pour les paquets capturés. La mise à disposition de ces différents formats par la société Endace [1], provient de la nécessité de garder une compatibilité ascendante avec les différents outils développés pour chacune des générations de cartes DAG. Etant en possession de la dernière génération, nous avions le choix entre deux variantes[1] du format le plus récent appelé ERF[2] (format préconisé par la société Endace). Après plusieurs expérimentations, nous avons choisi d'utiliser la version du format ERF avec une taille d'en-tête fixe. En effet, c'est ce format qui nous a permis de simplifier au maximum le développement des outils de traduction de format qui ont été nécessaires pour permettre à chacun des partenaires de disposer de traces dans un format dont il avait l'habitude (par exemple le format PCAP [30] produit par le logiciel TCPDUMP). Le détail des différents champs contenus dans un paquet capturé dans le format ERF avec un en-tête fixe est disponible sur la figure 7.1.

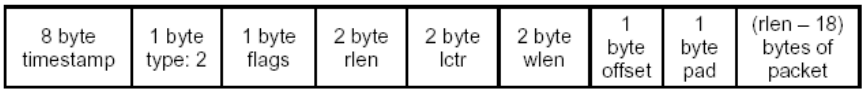


FIG. 7.1 – Détail du format ERF des paquets capturés par une sonde DAG (pour une architecture TCP/IP basée sur Ethernet 10/100 Mbps)

Le détail des différents champs est donné ci-après :
- timestamp : date d'arrivée du paquet,
- type : type de trame de niveau liaison (Eth, ATM, PoS)
- flags : informations diverses sur l'état de la capture (numéro de l'interface, enregistrement tronqué, etc),
- rlen (record length) : longueur totale de l'enregistrement transféré entre la carte de capture et le périphérique de stockage,
- lctr (loss conter) : nombre de paquets perdus entre la carte DAG et le périphérique de stockage (dans le cas d'une surcharge du bus PCI).
- wlen (wire length) : longueur du paquet capturé (rlen moins les informations rajoutés par la carte DAG),
- offset / pad : nombre d'octets qui n'ont pas été capturés au début de la trame (pour le moment cette fonctionnalité n'est pas implémentée sur les cartes DAG).

La valeur maximale d'un paquet capturé est fixé par le protocole de niveau liaison sur lequel la carte réalise la capture. Dans le cas qui nous intéresse pour le présent document, les captures sont réalisées sur un réseau local de type Fast-Ethernet. Etant donné qu'une trame Ethernet ne peut transporter plus de 1500 octets (valeur de la MTU[3] de données, la taille maximale d'une trame capturée est limitée à 1532 octets : 1500 octets pour les données utile du paquet Ethernet + 14 octets pour l'en-tête Ethernet (sans les 4 octets du CRC) + 18 octets pour l'en-tête DAG).

Cette contrainte de taille maximale a une incidence sur les trames provenant de réseaux avec une MTU supérieure à celle du réseau Ethernet et qui sont donc nécessairement fragmentés. Les différentes trames fragmentés sont donc capturées et stockées dans des enregistrements différents, le traitement de ces derniers devant être réalisé par le logiciel d'analyse des traces.

[1] La différence entre les deux variantes porte sur la taille des en-têtes qui dans un cas peut être fixe et dans l'autre variable. Cette spécificité permet une optimisation de l'espace utilisé pour le stockage de la trace capturée.

[2] ERF : Extensible Record Format

[3] MTU : Maximum Transmission Unit

L'intervalle pour la taille de chaque enregistrement dont nous disposons au moment de la capture est donc relativement important en théorie. En pratique, les captures sont soit réalisées sur les en-têtes TCP/IP seuls, soit sur l'ensemble du paquet TCP/IP. En effet, la capture partielle d'un paquet de données serait un inconvénient pour son analyse a posteriori. L'information capturée ne serait pas complète ce qui pourrait nuire à l'analyse des informations. Considérons pour illustrer cette notion l'exemple de l'analyse d'un trafic généré par une application de VoIP. Il arrive de plus en plus souvent que ce type de trafic soit encapsulé dans du trafic HTTP. Si la capture de l'en-tête applicatif du protocole HTTP est incomplète, il sera impossible de faire la différence entre un paquet HTTP contenant les données d'une page Web et un paquet HTTP servant à véhiculer du trafic VoIP.

La taille des paquets capturés aura un impact sur la complexité de l'algorithme à mettre en oeuvre dans le cadre de la procédure d'anonymisation que nous aborderons dans la section 7.3.

A titre de remarque, signalons que les cartes DAG capturent l'ensemble du trafic qui circule sur le réseau, ainsi tous les paquets générés par des protocoles de niveaux réseau et transport sont aussi capturés (ICMP, ARP, UDP...). Nous n'illustreront pas quantitativement l'impact de ce type de paquet sur la taille des enregistrements réalisés par les cartes DAG étant donné que le trafic TCP/IP est le trafic majoritaire sur les réseaux analysés dans Métropolis[4]. Nous tenons juste à signaler que ces différents paquets entraineront pour la plupart une modification de la taille de l'en-tête de niveau transport de la trame capturée sur le réseau.

7.2 Estampillage temporel

L'ensemble de la plate-forme de mesure Métropolis est synchronisée temporellement par des horloges GPS (à l'aide de cartes GPS installées dans les machines de capture). Ainsi, tout paquet qui est capturé sur le réseau est aussitôt estampillé par l'ajout d'un champ de 64 bits (cf. le champ "timestamp" de la figure 7.1). Dès lors, cette datation universelle permet de réaliser des calculs extrèmement précis (cette précision étant requise dans certains cas, par exemple, dans le cadre d'un calcul du délai de propagation de routeur en routeur pour des paquets IP dans un réseau de type MAN ou WAN).

Précisons que l'information temporelle délivrée par les cartes DAG est exprimée en "tics" d'horloge. Il a donc été nécessaire de développer des macros pour permettre de manipuler ces valeurs dans des unités plus courantes (microseconde, milliseconde ou seconde).

7.3 Anonymisation

La capture de trace réelles dans un réseau en production pose quelques problèmes. D'une part, les utilisateurs ne veulent pas que l'on puisse savoir comment ils utilisent leur connexion Internet de manière nominative. Mais surtout étant donné que les traces sont destinées à être rendues publiquement accessibles, il ne faut pas que l'on puisse piocher des informations comme des bouts de courriel, des numéros de carte bleue, ou encore de mots de passe qui circulent en clair sur le réseau.

Pour rassurer les utilisateurs, il suffit simplement de brouiller l'adresse IP source et destination. Tandis que pour les informations, le plus simple et surtout le plus sûr est de retirer tout les données.

[4] En effet, dans l'Internet plus de 80 % du trafic, quelle que soit la granularité d'étude, est représenté par le trafic TCP/IP.

Dans cette sous-section, nous allons ainsi présenter la solution retenue pour l'anonymisation des adresses.

7.3.1 Objectif

L'approche la plus simple pour anonymiser l'adresse IP_i est de lui associer de manière aléatoire une adresse IP_i^a prise dans l'ensemble des adresses IP. La seule contrainte est alors d'assurer que la relation entre les deux ensemble est bijective. Garantir le caractère anonyme des adresses consiste alors à ne pas rendre publique la table de correspondance.

La principale limite de cette méthode est de casser la relation entre les adresses IP. C'est à dire que deux adresses proches avant l'anonymisation ne seront pas nécessairement proche dans la trace anonyme. Cela peut se révéler gênant si l'on veut travailler sur des flots agrégés.

Il est donc nécessaire que la procédure d'anonymisation conserve les préfixes. C'est à dire que si IP_1 et IP_2 partagent k bits de préfixe dans la trace capturée alors IP_1^a et IP_2^a partagent k bits de préfixe dans la trace anonymisée. Il existe deux solutions simples et publiques pour réussir cela : TCPdpriv [2] et une approche cryptographique [32].

Le principe de l'approche TCPdpriv est le suivant. TCPdpriv peut être vu comme une table qui à chaque adresse IP non anonyme associe de manière aléatoire une autre IP. La table des adresses IP anonymes commence à 1 et à chaque nouvelle adresse lue dans la trace, on associe la prochaine IP anonyme disponible. Ainsi la première adresse IP vue dans la trace sera anonymisée en 0.0.0.1, la seconde en 0.0.0.2. Etant donné que l'ordre d'arrivée des adresses IP est inconnu, il est impossible, à partir de l'adresse anonyme de revenir à l'adresse originale.

L'approche de TCPdpriv a vite été écartée pour son manque de scalabilité. En effet pour garantir que deux sites distincts anonymisent les traces de la même manière, il faut que chacun ait une copie de l'ensemble de la table d'anonymisation. Comme cette table contient beaucoup d'adresses IP, et qu'elle grandit à chaque nouvelle adresse IP rencontrée, cela devient vite problématique de partager cette table. En effet, si chaque trace est anonymisée différemment, il est impossible de fusionner des traces anonymisées à partir de deux sites différents. Il est alors impossible d'étudier le passage de paquets sondes à travers différents points du réseau.

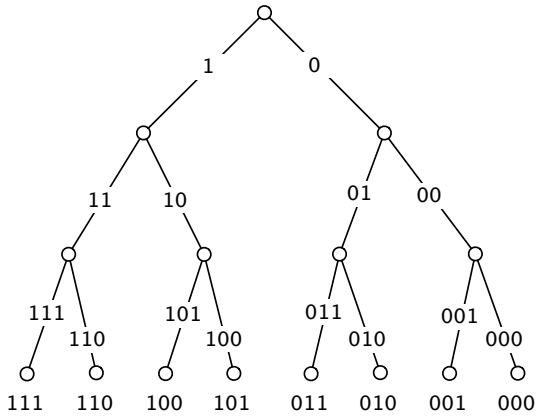
7.3.2 Anonymisation cryptographique

Pour l'explication de la procédure d'anonymisation, nous allons regarder l'ensemble des adresses IPv4[5] comme un arbre binaire équilibré de profondeur 32. Nous ne nous occuperons ici que de la description de l'anonymisation des adresses IPv4. Mais l'algorithme est transposable aisément à des adresses plus longues.

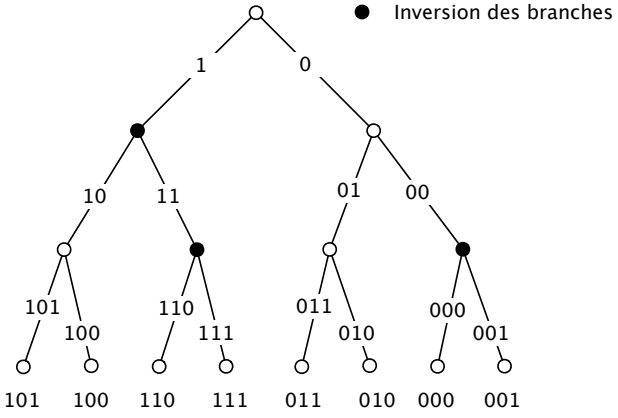
La procédure d'anonymisation conservant les préfixes peut être vue comme une fonction qui à chaque nœud (autre que les feuilles) inverse ou non les branches. En permutant certains nœuds, on obtiens l'arbre de la figure 7.2-b. Ainsi l'adresse 000 est transformée en 001, 101 en 111,...

Une solution cryptographique à été développée dans [32] pour que, à l'aide d'une clef relativement courte (par rapport à la taille de la table d'association de toutes les adresses IP), on puisse

[5] en IPv4, les adresses sont codées sur 32 bits



(a)



(b)

FIG. 7.2 – Les adresses IP vues comme un arbre : (a) avant anonymisation, (b) après anonymisation.

anonymiser une adresse IP. La fonction qui à chaque nœud i associe une permutation peut-être écrite comme :

$$f_i(a_1 a_2 \dots a_i) = L(R(P(a_1 a_2 \dots a_i), k))$$

où :

- $i = 0, 1, \dots, n-1$,
- L la fonction qui retourne le bit le moins significatif,
- R est une fonction pseudo-aléatoire (comme par exemple Rijndael [8]),
- P est une fonction de bourrage qui étend $(a_1 \dots a_i)$,

– k la clef cryptographique utilisée par la fonction R.

La fonction f_i est uniquement déterminée par k. En d'autres termes, une adresse IP sera anonymisée de la même façon par deux sites différents s'ils partagent la même clef. Comme la taille de la clef est de l'ordre de la centaine de bits, elle est aisée à transmettre à travers un canal sécurisé.

Nous avons écrit une implémentation en c++ de cette fonction d'anonymisation, à partir des sources disponibles sur le web.

7.3.3 Mise en oeuvre du mécanisme d'anonymisation

Il existe un grand nombre de protocoles au dessus d'IP comme TCP, UDP, ICMP[6], EGP[7], IGMP[8], etc . . . Mais en pratique, nous n'observons quasiment aucun paquet autre que TCP, UDP ou ICMP sur les réseaux que nous mesurons. Nous avons donc décidé de ne traiter que les paquets TCP, UDP et ICMP. La figure 7.3 résume la façon dont nous traitons les paquets lors de l'anonymisation.

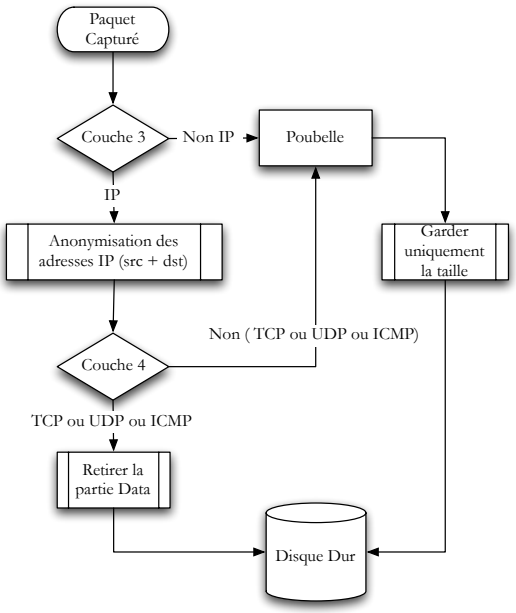


FIG. 7.3 – Détail du processus d'annonymisation

7.3.4 Quelques attaques possibles

Cependant, toute solution sécurisée peut faire l'objet d'attaques. Nous avons donc évalué le potentiel des attaques possibles sur la solution d'anonymisation cryptographique.

[6] Internet Control Message Protocol
[7] Exterior Gatway Protocol
[8] Internet Group Management Protocol

Attaque cryptographique. Aidé par la connaissance de couples d'adresses IP de la trace originale et de leur version anonymisée, un attaquant peut essayer de retrouver la clef cryptographique utilisée lors de l'anonymisation. Dans [32], il a été prouvé que cette attaque n'est pas utilisable dans la pratique.

Attaque sémantique. Toujours à l'aide de couples d'adresses IP, l'attaquant peut essayer de retrouver une partie du préfixe, ou la totalité d'une adresse. Pour cela, il peut utiliser des techniques cryptographiques standard comme l'analyse en fréquence. La sécurité de l'implémentation à ce type d'attaque dépend uniquement de la fonction pseudo-aléatoire utilisée. En utilisant Rijndael avec une clef de 128 bits, nous rejoignons les standards de sécurité utilisés à l'heure actuelle.

Chapitre 8

Caractérisation simple du trafic

8.1 Caractéristiques retenues

La plateforme de capture Métropolis nous permet de réaliser et de stocker un nombre important d'échantillons de trafic. Ces derniers, de part leur durée ou l'état du réseau au moment de leur réalisation, peuvent posséder des caractéristiques et illustrer des comportements très différents. Il est donc nécessaire de disposer d'un ensemble de paramètres de base permettant de les comparer. Pour cela, nous avons sélectionné les caractéristiques suivantes :

- Les premières sont statiques :
  - nombre de paquets et d'octets dans la trace (tous protocoles confondus, uniquement TCP ou UDP)
  - nombre de flux (tous protocoles confondus, uniquement TCP ou UDP)
  - répartition du nombre de paquets, d'octets et de flux par type d'application
  - nombre d'acquittements TCP
  - nombre de pertes TCP
- D'autres dynamiques :
  - évolution du débit global au cours du temps,
  - évolution du débit TCP ou UDP ou par application au cours du temps,
  - évolution du RTT des flux TCP au cours du temps,
- Ainsi que certaines statistiques :
  - distribution des inter-arrivées des flux TCP
  - distribution de la taille des flux TCP
  - distribution de la durée des flux TCP
  - distribution des inter-arrivées des paquets TCP
  - distribution des tailles des paquets TCP

Nous sommes conscients que cette liste de paramètres n'est pas exhaustive. En effet, un certain nombre d'autres caractéristiques pourraient être ajoutées et ainsi améliorer la précision de la caractérisation. Néanmoins, avec l'ensemble des paramètres sélectionnés nous disposons des différents niveaux d'analyse (octet, paquet ou flux) nécessaires pour pouvoir porter différents regards sur l'état du réseau, illustrer ses comportements et ainsi isoler certaines de ses propriétés. Ainsi, les paramètres statistiques permettent de réaliser une évaluation des performances du réseau considéré en

estimant, par exemple, le niveau moyen d'utilisation de ses différents liens ou encore une estimation des comportements utilisateurs en permettant de déterminer le profil général des flux (TCP ou UDP) qui y sont échangés (par exemple en terme de taille, de durée ou de loi d'arrivée).

D'autre part, bien que ces premiers résultats mettent en évidence les possibilités d'analyse du trafic offertes par les paramètres retenus précédemment, ils vont aussi mettre en évidence leur limitation. En effet, on montrera que lorsqu'on souhaite dépasser le cadre de la supervision réseau et s'orienter vers des phases de modélisation ou d'optimisation du réseau, ces informations ne sont plus suffisantes. Il apparaît alors nécessaire de se focaliser sur des paramètres plus complexes (par exemple des grandeurs statistiques d'ordre 2 comme nous le verrons dans la section 10) ou encore de proposer des méthodes d'analyse du trafic qui vont au delà de la simple décomposition en octet, paquet ou flux (ce sera l'objectif de la section 9).

Mais avant de rentrer plus avant dans les méthodes visant à améliorer la phase de caractérisation du réseau qui feront l'objet des chapitres suivants, focalisons nous sur les paramètres qui sont majoritairement utilisés à l'heure actuelle. L'un des objectifs du sous-projet 3 est de fournir une batterie d'outils permettant de caractériser le trafic collecté. Dès lors, dans cette partie, nous nous attarderons donc sur les caractéristiques simples que l'on peut isoler dans le trafic et sur les informations qu'elles apportent sur l'état ou le comportement du réseau.

Ainsi, dans la suite nous allons illustrer les différentes caractéristiques que nous avons pu observer sur deux types de trafic (cf. figure 8.1 pour la topologie des réseaux analysés dans le projet Métropolis) :

- Un trafic collecté sur le lien d'accès du LAAS-CNRS qui est un lien Fast-Ethernet,
- Un trafic collecté sur le lien d'accès du réseau de Jussieu à Renater qui est un lien Giga-Ethernet.

Nous tenterons, dans un premier temps, de montrer en quoi elles diffèrent et dans un second, d'expliquer quels types de comportements réseaux ces différences peuvent mettre en évidence.

8.2 Résultats de caractérisation pour le lien d'accès du LAAS

- Caractéristiques statiques
  Le tableau 8.1 résume l'ensemble des valeurs pour les différents paramètres mesurés au LAAS-CNRS.
  On observe dans ce tableau que la très grande majorité du trafic collecté au LAAS-CNRS est du trafic TCP. Cette remarque va dans le sens de ce qui est traditionnellement observé dans l'Internet.
  Si l'on s'intéresse maintenant à la répartition du trafic par application au sein du LAAS-CNRS (cf. figure 8.2, classification obtenue par l'intermédiaire de la sonde QoSMOS) on observe que le trafic P2P (Kazaa et E-donkey) est en proportion importante (à peu près équivalent au trafic HTTP). Cette remarque illustre l'intérêt d'une caractérisation simple du réseau. En effet, les ingénieurs systèmes du LAAS-CNRS, à la découverte de cette importante quantité de trafic P2P ont fermé l'ensemble des ports jusque là ouverts sur les firewalls permettant ainsi de réduire très fortement la proportion de ce trafic. D'autre part, ce résultat illustre les limitations d'une classification du trafic par numéro de port qui se révèle très incomplète par rapport à celle obtenue par l'intermédiaire de la sonde QoSMOS.
- Caractéristiques dynamiques

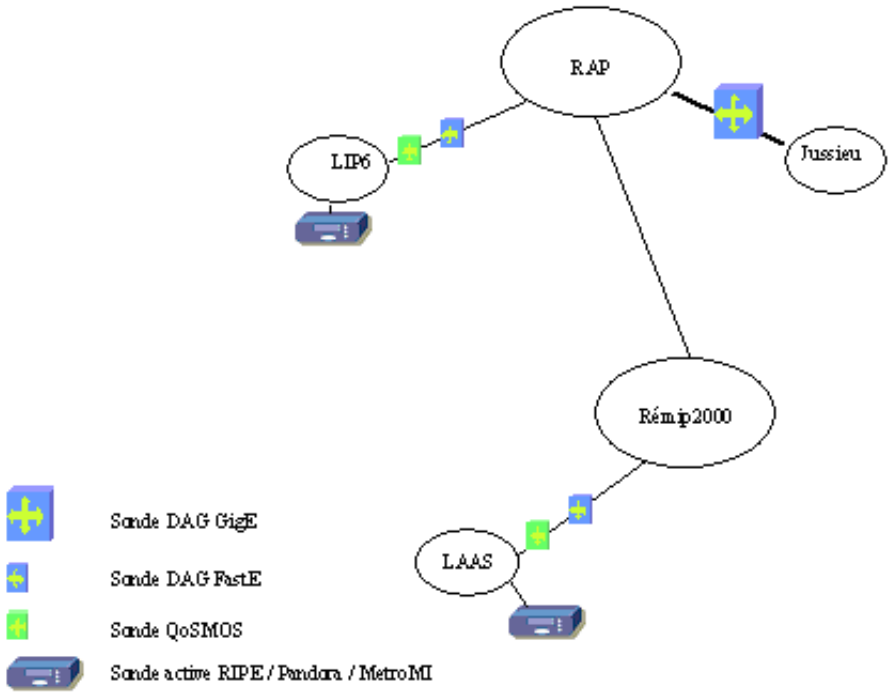


FIG. 8.1 – Répartition des sondes de métrologique passives pour projet Métropolis

TAB. 8.1 – Résultats de caractéristation obtenus au LAAS-CNRS

Paramètre	Valeur
Jour de la capture	24/04/2003
Heure de la capture	10 h 30 min 10 secondes
Durée de la trace (s)	5 774
Nombre de paquets (IP)	5 000 000
Nombre d'octets (IP)	1 134 711 699
Nombre de paquets (TCP)	3 993 288 (79,87 %)
Nombre d'octets (TCP)	1 088 131 640 (95,89 %)
Nombre de paquets (UDP)	233 987 (14,68 %)
Nombre d'octets (UDP)	46 580 059 octets (4,11 %)
Nombre d'acquittements (TCP)	965 165
Nombre de flux (tous protocoles)	420 132
Nombre de flux (TCP)	208 435 (49,61 %)
Nombre de flux (UDP)	211 697 (50,39 %)

Les figures 8.3 et 8.4 représentent respectivement, l'évolution du débit global (tous protocoles confondus) et l'évolution du débit pour le trafic TCP uniquement analysées sur le lien d'accès à Remip et Renater du LAAS-CNRS.
On peut noter la sous-utilisation générale du lien du LAAS-CNRS au moment de la capture. En effet, le plus fort pic atteint 1 Moctet, ce qui fait un taux d'utilisation d'environ 10 % au maximum de la charge.

– Caractéristiques statistiques
Les figures 8.5, 8.6, 8.7, 8.8 et 8.9 représentent les paramètres statistiques observés pour la trace collectée sur le POP du LAAS-CNRS.
On observe sur les distributions des lois d'arrivées des flux et des paquets TCP des comportements asymptotiquement exponentiel ou exponentiel par morceaux (cf. apparition de segment approximable par une droite dans les distributions des figures 8.5 et 8.8).
La durée moyenne des flux TCP (cf. figure 8.7) est de 100 secondes alors que leur taille moyenne est de 450 octets (cf. figure 8.6).
On observe sur la distribution de la taille des paquets TCP deux pics (aux alentours de 40 et 1500 octets), ces deux pics sont l'illustration de la forte proportion dans le trafic analysé, respectivement, de paquets d'acquittement TCP et de paquets qui ont une taille égale à la MTU Ethernet.

8.3 Résultats de caractérisation pour l'accès de Jussieu

– Caractéristiques statiques
Le tableau 8.2 résume l'ensemble des valeurs pour les différents paramètres mesurés sur le POP de Jussieu. On observe comme dans le cas du réseau du LAAS-CNRS une très forte proportion du trafic TCP. Nous n'avons pas été en mesure de réaliser une étude de la répartition du trafic par application aussi détaillée que celle présentée pour le réseau du LAAS-CNRS (car la sonde QoSMOS ne supporte pas des débits aussi élevés et le logiciel de rejeu TDPlayer n'est pas encore suffisament au point). Néanmoins, nous présentons dans le tableau 8.2 des

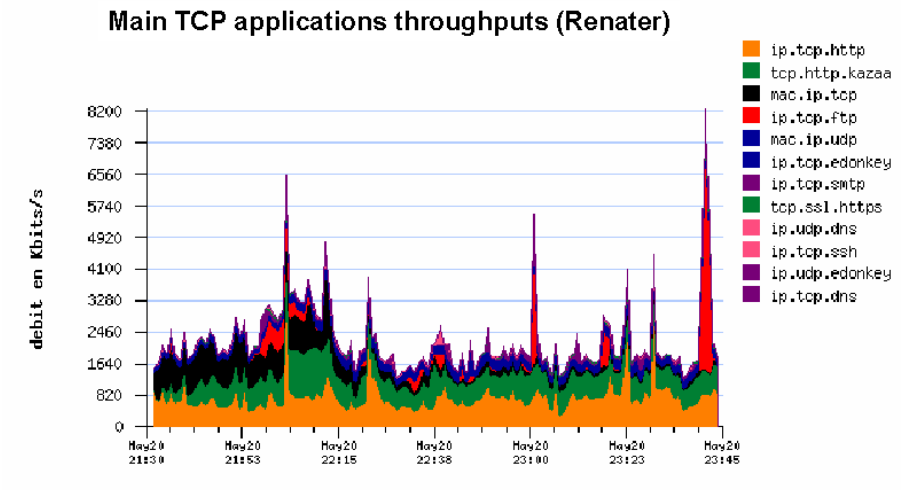


FIG. 8.2 – Répartition des différentes applications au cours du temps par ordre décroissant (LAAS-CNRS)

résultats qui permettent d'avoir une première idée de la proportion de trafic classique (i.e. non P2P) circulant sur le réseau de Jussieu[1] à partir d'une classification par les numéros de port. On observe en particulier une forte proportion de trafic FTP. Nous reviendrons sur cette observation par la suite.

– Caractéristiques dynamiques
  Les figures 8.10 et 8.11 représentent respectivement, l'évolution du débit global (tous protocoles confondus) et l'évolution du débit pour le trafic TCP uniquement analysées sur le POP de Jussieu.
  On peut à nouveau noter la sous-utilisation générale du lien de Jussieu au moment de la capture. En effet, le plus fort pic atteint 13 Moctet, ce qui fait un taux d'utilisation d'environ 10 % au maximum de la charge.
– Caractéristiques statistiques
  Les figures 8.12, 8.13, 8.14, 8.15 et 8.16 représentent les paramètres statistiques observés pour la trace collectée sur le POP de Jussieu.
  La durée moyenne des flux TCP est de 100 secondes alors que la taille moyenne est de 550 octets.

[1] Nous n'avons pas pris en compte dans cette analyse les nouveaux protocoles de P2P comme Kazaa ou E-Donkey aux numéros de port très dynamiques.

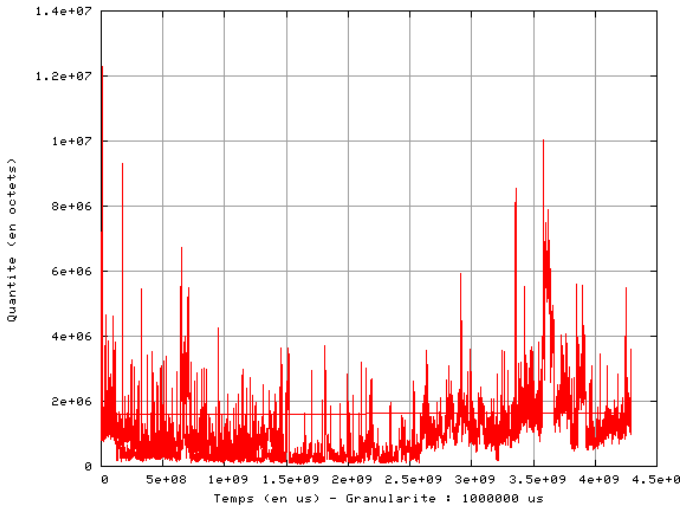


FIG. 8.3 – Evolution du débit global au cours du temps (LAAS-CNRS)

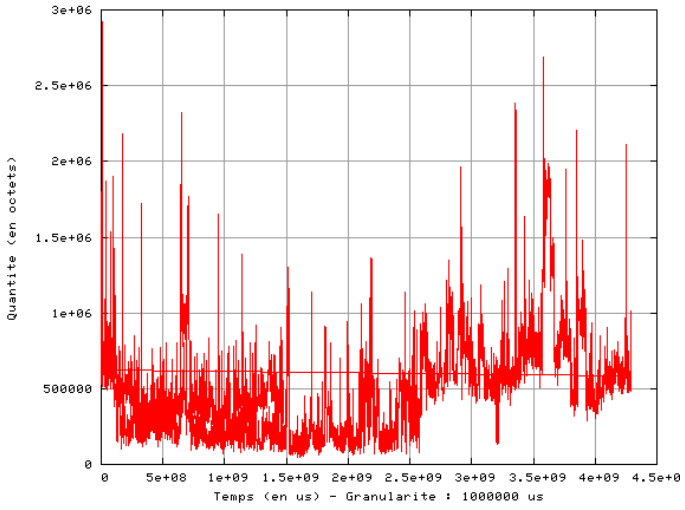


FIG. 8.4 – Evolution du débit TCP au cours du temps (LAAS-CNRS)

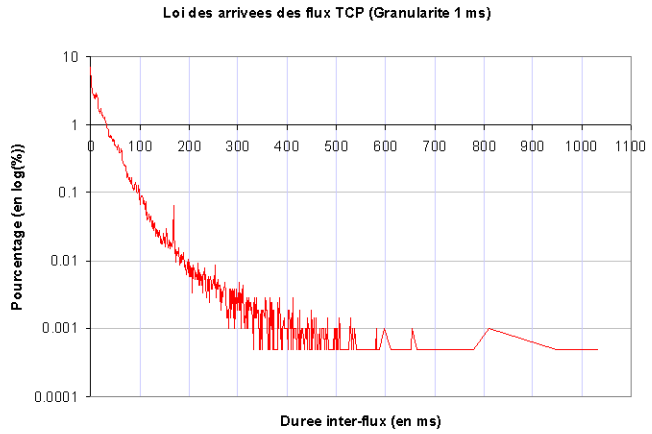


FIG. 8.5 – Distribution des inter-arrivées des flux TCP (LAAS-CNRS)

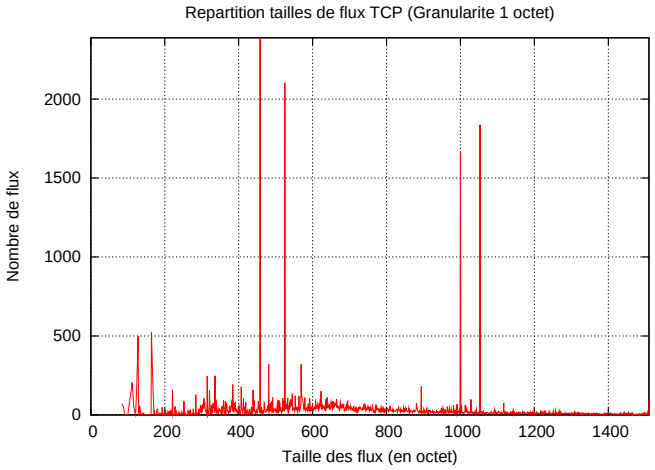


FIG. 8.6 – Distribution des tailles de flux TCP (LAAS-CNRS)

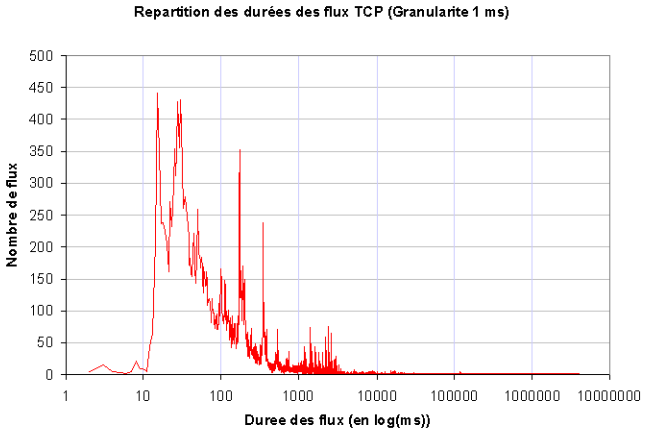


FIG. 8.7 – Distribution des durees de flux TCP (LAAS-CNRS)

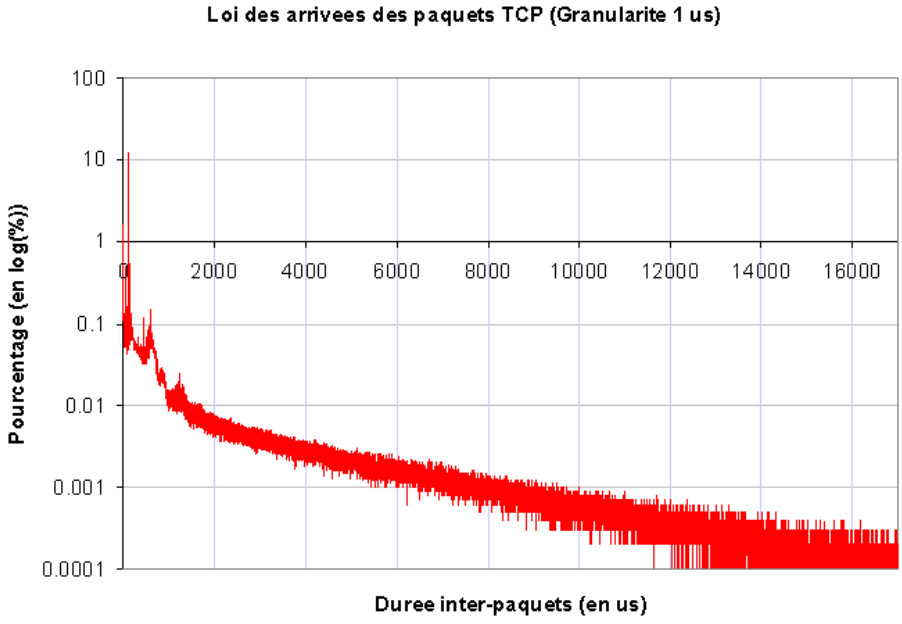


FIG. 8.8 – Distribution des inter-arrivées des paquets TCP (LAAS-CNRS)

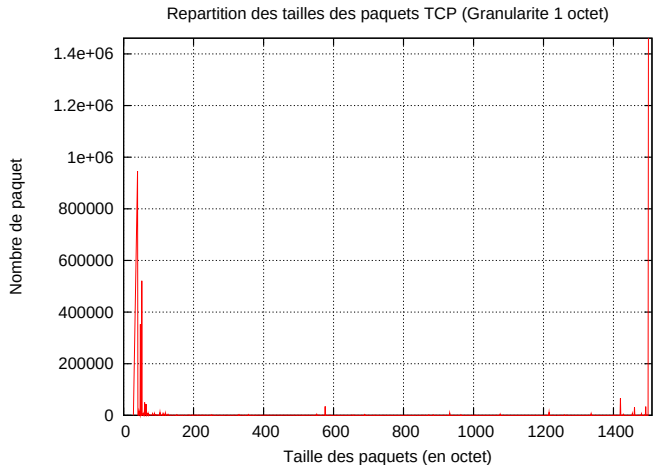


FIG. 8.9 – Distribution des tailles de paquet TCP (LAAS-CNRS)

TAB. 8.2 – Résultats de caractéristation obtenus à Jussieu (1/2)

Paramètre	Valeur
Jour de la capture	19/09/2003
Heure de la capture	15 h 20 min 05 secondes
Durée de la trace (s)	409
Nombre de paquets (IP)	5 000 000
Nombre d'octets (IP)	1 554 009 425
Nombre de paquets (TCP)	4 942 518 (98,85 %)
Nombre d'octets (TCP)	1 508 606 629 octets (97,08 %)
Nombre de paquets (UDP)	56 725 (1,13 %)
Nombre d'octets (UDP)	45 368 727 octets (2,92 %)
Nombre d'acquittements (TCP)	608 081
Nombre de flux (tous protocoles)	167 659
Nombre de flux (TCP)	127 258 (75,90 %)
Nombre de flux (UDP)	40 401 (24,10 %)

TAB. 8.3 – Résultats de caractéristation obtenus à Jussieu (2/2)

Type d'application	Volume	Pourcentage (%)
Web (octets)	1 198 423 016	77,12
Web (paquets)	1 868 012	37,36
Web (flux)	173 129	65,26
Secure Web (octets)	16 104 997	1,04
Secure Web (paquets)	43 187	0,86
Secure Web (flux)	4 443 flux	2,65
Ftp (octets)	467 670 407	30,09
Ftp (paquets)	526 425	10,53
Ftp (flux)	51 255	30,57
Smtp (octets)	91 174 332	5,87
Smtp (paquets)	150 020	3,00
Smtp (flux)	7 920	4,72
Telnet (octets)	2 332 404	0,15
Telnet (paquets)	14 178	0,28
Telnet (flux)	1 442	0,86
RealaudioTcp (octets)	7 209 746	0,46
RealaudioTcp (paquets)	16748	0,33
RealaudioTcp (flux)	3 171	0,80
RealaudioUdp (octets)	2 398 928	0,15
RealaudioUdp (paquets)	3 028	0,06
RealaudioUdp (flux)	1 680	1,00
MediaplayerTcp (octets)	46 076 778	2,97
MediaplayerTcp (paquets)	63 344	1,27
MediaplayerTcp (flux)	6 435	3,84
MediaplayerUdp (octets)	691 300	0,04
MediaplayerUdp (paquets)	872	0,02
MediaplayerUdp (flux)	518	0,31
Napster (octets)	2 740 390	0,18
Napster (paquets)	4 982	0,10
Napster (flux)	949	0,57
Nntp (octets)	160 595 690	10,33
Nntp (paquets)	337 901	6,76
Nntp (flux)	27 325	16,30

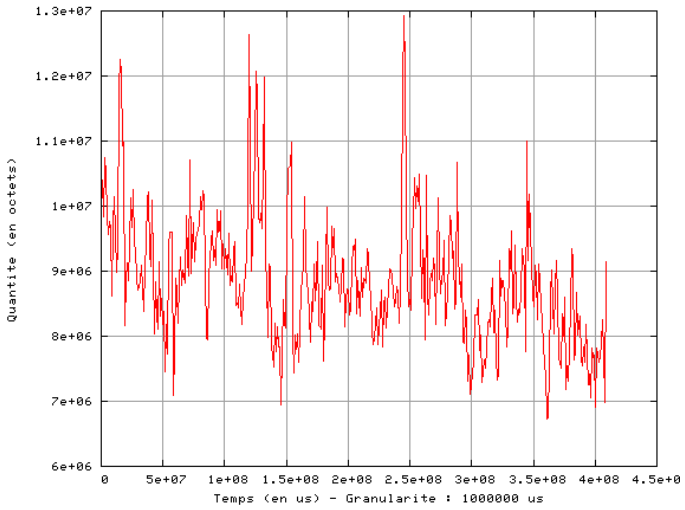


FIG. 8.10 – Evolution du débit global au cours du temps (Jussieu)

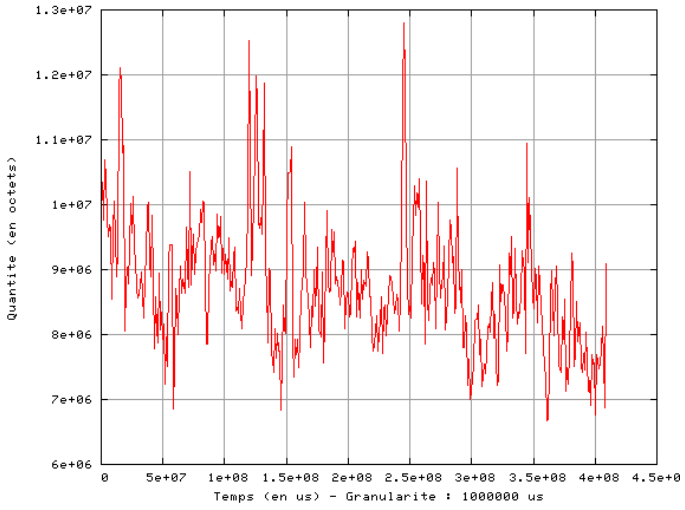


FIG. 8.11 – Evolution du débit TCP au cours du temps (Jussieu)

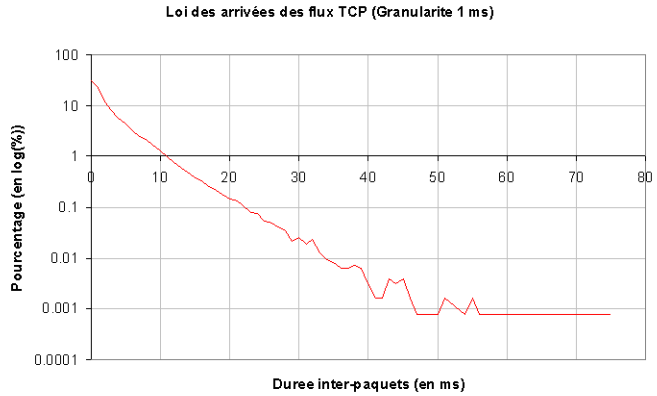


FIG. 8.12 – Distribution des inter-arrivées des flux TCP (Jussieu)

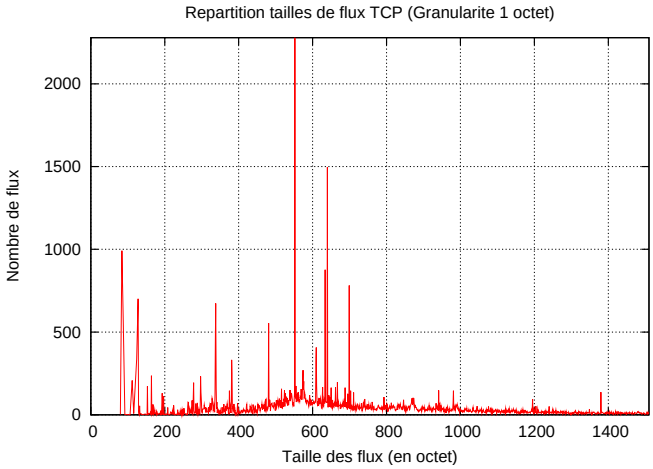


FIG. 8.13 – Distribution des tailles de flux TCP (Jussieu)

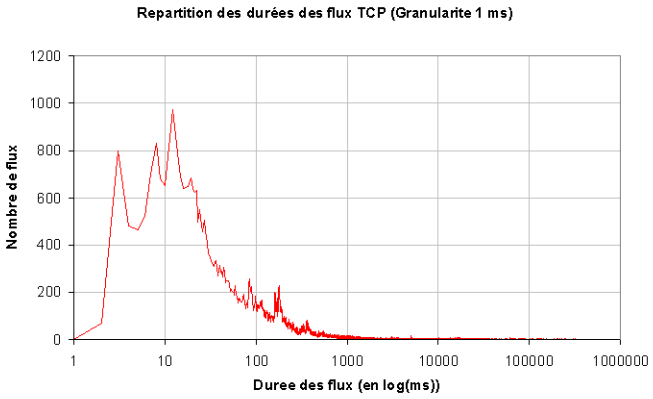


FIG. 8.14 – Distribution des durees de flux TCP (Jussieu)

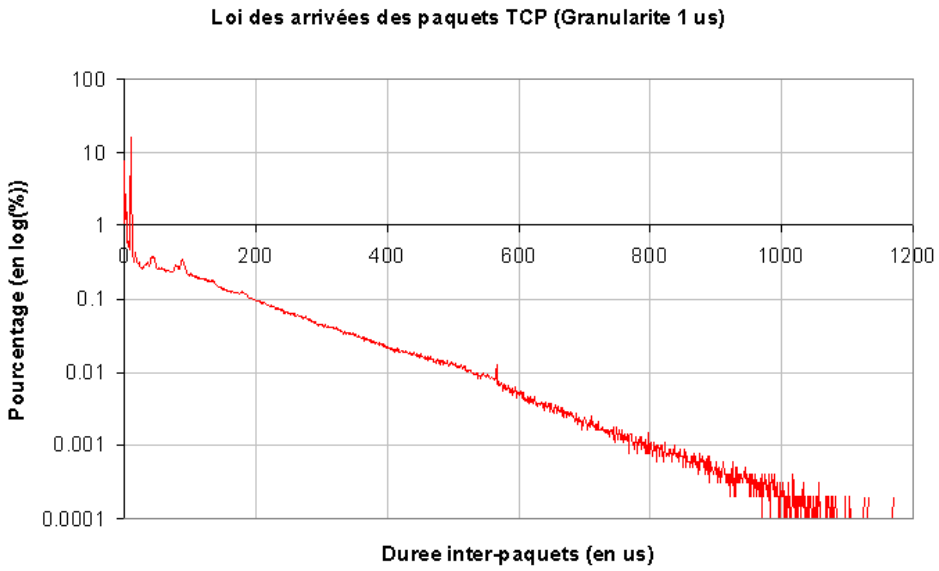


FIG. 8.15 – Distribution des inter-arrivées des paquets TCP (Jussieu)

La distribution de la taille des paquets est la même que pour le réseau du LAAS, on observe deux pics principaux aux alentours de 40 octets et 1500 octest.
Pour ce qui est des lois d'arrivées des paquets et des flux, le comportement exponentiel est beaucoup plus marqué sur ce type de lien à haut débit que dans le cadre du LAAS-CNRS. En effet, la queue des distributions s'approxime facilement par une droite dans les figures 8.12 et 8.15.

8.4 Conclusion

Si l'on met en exergue les résultats obtenus sur les deux types de réseau, un certain nombre de différences apparaissent. En effet, si l'on analyse avec attention les données de la trace, on met en évidence que le type de trafic véhiculé est différent.

Tout d'abord, en menant une étude uniquement quantitative des différentes caractéristiques nous avons montré que le trafic du LAAS-CNRS en terme de flux s'équilibre entre les protocoles TCP et UDP alors que cette proportion n'est pas respectée à Jussieu où plus de 75 % des flux sont des flux TCP. D'autre part, la part de trafic Web au LAAS-CNRS est plus grande que sur le réseau de Jussieu (84 % contre 77 %) alors que pour le trafic FTP on observe le phénomène inverse (9 % contre 30 %). Cette dernière remarque peut s'expliquer par le fait que le site de Jussieu abrite un grand nombre de données en miroir qui représentent un volume très important d'information téléchargée par l'intermédiaire du protocole FTP.

De plus, on observe que le réseau de Jussieu connait un niveau de fonctionnement beaucoup plus important que celui du LAAS : la durée moyenne entre deux arrivées consécutives de flux sur le

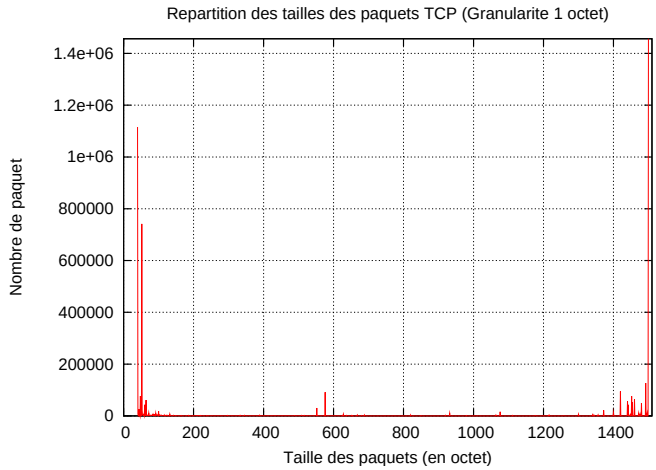


FIG. 8.16 – Distribution des tailles de paquet TCP (Jussieu)

réseau de Jussieu est de 5 ms contre 50 ms pour le réseau du LAAS. Ce résultat est corroboré par les courbes de débit qui mettent en évidence un volume d'information beaucoup plus important pour le réseau de Jussieu. L'analyse de la distribution des interpaquets pour les deux réseaux corrobore ce résultat étant donné que la durée moyenne séparant deux arrivées de paquets est beaucoup plus courte pour le trafic de Jussieu que pour celui du LAAS.

Ainsi, cette première analyse quantitative, certes rudimentaire, permet déjà de mettre en évidence certains comportements (du réseau ou des utilisateurs) différents entre les deux points de collecte.

On peut signaler que l'on observe aussi des invariants entre les deux types de réseau comme par exemple la répartition des tailles de paquets ou la distribution des durées de flux qui sont sensiblement identiques entre les deux réseaux. Cette invariance est présente à deux niveaux de granularité : paquet et flux.

A l'inverse, si l'on reprend maintenant les données statistiques présentées telles que les distributions des inter-arrivées des paquets ou des flux pour les différents réseaux, on observe des lois d'arrivée très différentes pour les deux types de réseaux et ce aux niveaux paquet et flux. En effet, le trafic du LAAS-CNRS est beaucoup plus éloigné d'un processus poissonnien que le trafic de Jussieu et rejoint en cela les résultats décrits dans [7]. Ce résultat illustre la difficulté actuelle à laquelle les chercheurs se heurtent lorsqu'ils tentent de caractériser le trafic Internet dans son ensemble. Cette remarque est très importante pour la suite car elle motive notre besoin de décomposition du trafic en classe pour permettre une caractérisation plus aisée.

La comparaison de ces caractéristiques pourrait se poursuivre mais la conclusion qui a motivé la suite de notre travail est qu'un tel niveau de caractérisation bien que très intéressant et indispensable pour réaliser de la supervision réseau s'avère assez limité lorsque l'on souhaite mettre en évidence des comportements plus fins au niveau du réseau ou des utilisateurs. Le projet Métropolis souhaite proposer une méthodologie de caractérisation qui mette en évidence des résultats qui soient plus forts que ceux obtenus dans la cadre de la supervision réseau traditionnelle.

En effet, nous allons illustrer dans la suite de ce document que la complexité du trafic Internet nécessite une décomposition en sous classe de trafic pour permettre une caractérisation plus aisée de ses différentes parties et par la suite simplifier les tâches de modélisation, d'évaluation de performances ou encore de dimensionnement du réseau.

Chapitre 9

Caractérisation zoologique

9.1 Motivations

Modéliser le trafic n'est pas une tâche aisée. Ce problème est soulevé depuis plusieurs années par une grande partie de la communauté réseau mais à l'heure actuelle aucun modèle proposé n'a été accepté par cette dernière. En effet, les premiers résultats montrent que les caractéristiques du trafic ne sont pas simples et difficilement intégrables dans un modèle. Plus précisément, les premiers résultats ont montrés que le trafic Internet est dépendant à long terme (LRD[1]) [4], auto-similaire [19] ou encore multi-fractal [9]. Dans tous les cas, même si des modèles pour le trafic Internet peuvent être proposés, ils ne sont pas du tout simples et ne décrivent pas tous les aspects du trafic. Cette complexité des modèles entraîne que, même de nos jours, chercheurs et ingénieurs réseaux continuent d'utiliser des sources de trafic Markoviennes (de type Poisson la plupart du temps), étant donné qu'il reste difficile de développer des générateurs de trafic intégrant les spécificités des modèles auto-similaires ou fractaux. Il en va de même pour les modèles utilisés pour adresser les problèmes d'évaluation de performance.

De plus, les modèles de trafic doivent être aussi simples que possible ; par exemple, Markoviens ou encore mieux Poissonniens. De nombreux outils implémentant ces modèles existent et permettent aux utilisateurs de facilement analyser et valider leurs propositions. Malheureusement, le trafic Internet n'est pas Poissonnien ! Ainsi, le travail détaillé dans cette section porte donc sur la détermination d'une méthode pour classifier les flux d'un trafic donné en différentes classes dans le but d'associer à chaque composante du trafic identifiée une classe particulière qui suivrait un modèle simple et bien connu (Poisson, Markov, Gaussien, etc.). Cette idée, qui va de pair avec le besoin d'appliquer des actions différentes en fonction des différentes classes de trafic, provient d'une première analyse réalisée sur le trafic Internet en s'intéressant à la taille des flux (i.e. savoir si le flux considéré est une "souris" ou un "elephant"). Les résultats de cette étude, détaillés dans la suite de cette partie, ont montré que le trafic des souris (du moins au niveau paquet) peut être assimilé à un processus Poissonnien. D'autre part, les arrivées de flux éléphants peuvent aussi suivre un processus Poissonnien. Ainsi, nous avons souhaité allez plus loin dans ce schéma de classification et définir d'autres classes de flux : les tortues, les libellules, les buffles, etc. Pour réaliser cet objectif, nous avons développé

[1] LRD : Long Range Dependent

un outil générique pour analyser les propriétés de base du trafic pour ces différentes classes, et ainsi pouvoir valider si elles suivent ou non un modèle de processus simple. Cet outil s'appelle "ZOO". Son nom s'est imposé à nous étant donné qu'il permet de mettre en évidence, comme nous allons le voir dans la suite, toute la "ménagerie" des flux qui sont présents dans l'Internet.

Ainsi, cette partie décrit le travail de classification réalisé dans l'outil Zoo et présente les différentes familles d'animaux qui ont été créées pour désigner les différentes sortes de flux. Le but final de ce travail sera de trouver une classe pour chaque flux, et un trafic pour chaque classe qui suit un modèle basé sur une loi simple. Ainsi, la modélisation du trafic Internet apparaitra aussi simple qu'une agrégation de processus stochastiques de base. Dans tous les cas, même si toutes les classes de flux ne peuvent être créées pour suivre des processus simples, ce travail sera d'une grande importance si un grand nombre de classes de flux suivent un modèle simple, ceci entrainant une forte simplification de la tâche de modélisation du trafic, ainsi que des autres tâches associées : dimensionnement du réseau, gestion de la QoS, ingénierie du trafic, planification du réseau, etc.

9.2 Description du logiciel ZOO

9.2.1 Fonctionnalités du logiciel

Les fonctionnalités du logiciel Zoo sont décomposables en deux grandes classes. Une première qui permet d'extraire d'une trace de trafic toutes les caractéristiques basiques qui ont été présentées dans la partie précédente et une seconde, qui est en relation avec le besoin, expliqué dans le paragraphe précédent, de décomposer le trafic en différentes classes.

Les caractéristiques de base ont déjà fait l'objet d'une description détaillée dans la partie précédente, nous allons donc uniquement détailler les fonctionnalités associées à la deuxième classe d'entre elles.

Ainsi, le logiciel Zoo, à partir d'une trace de trafic, permet une analyse aux niveaux paquets et flux et offre les décompositions suivantes :

- souris / éléphant : il s'agit de distinguer les flux selon leur volume en nombre de paquets (moins de 10 = flux souris, plus de 100 = flux éléphants)[2].
- libellules / tortues[3] : il s'agit de distinguer les flux selon leur durée de vie dans la trace (moins de 2 secondes = flux libellule, plus de 15 minutes = flux tortue).

L'un des objectifs principaux de cet outil étant de trouver un processus simple à chaque classe de flux identifiée, Zoo fournit donc à l'utilisateur un certain nombre de paramètres lui permettant d'infirmer ou de confirmer si les hypothèses stochastiques faites pour une classe particulière du trafic sont correctes, ces paramètres (valables pour chaque classe analysée) sont :

- la distribution des arrivées de flux,
- la distribution des arrivées de paquets,
- le niveau de corrélation des arrivées de flux,
- le niveau de corrélation des arrivées de paquets,
- le niveau de LRD des arrivées de flux,

[2]Ces bornes ont été choisies en s'appuyant sur la définition des flux souris et éléphants présentée dans [16] mais elles restent modifiables par l'utilisateur du logiciel

[3]Cette décomposition a été présentée dans [22], nous nous sommes inspirés de ce travail pour déterminer les bornes temporelles qui permettent de différencier un flux libellule d'un flux tortue.

– le niveau de LRD des arrivées de paquets.

Ainsi, l'utilisateur à la possibilité de déterminer si une caractéristique stochastique précise (par exemple le caractère Poissonnien des lois d'arrivées des paquets pour une classe de flux donnée) est recevable. Pour cela, l'utilisateur peut appliquer, pour chaque classe considérée, une démarche systématique qui se décompose selon les différentes étapes ci-dessous :

1. Déterminer les distributions des inter-arrivées,
2. Chercher si cette distribution suit une loi exponentielle par l'intermédiaire de la décroissance de la distribution et de son niveau d'auto-corrélation,
3. Evaluer le niveau de dépendance de la série de valeurs en se référant :
   – au diagramme LDestimate représentant l'évolution de la variance en fonction de la largeur d'analyse temporelle (cf. analyse statistique au second ordre par la méthode de la transformée en ondelettes [4]).
   – à l'évaluation quantitative du niveau de dépendance à long terme donné par le facteur de Hurst.

9.2.2 Fonctionnement du logiciel

Pour simplifier l'explication du fonctionnnement du logiciel Zoo, nous allons d'abord présenter l'algorithme de traitement et de classement des flux en classe dans sa globalité avant de présenter plus en détails les traitements spécifiques appliqués aux différentes classes.

– Description globale de l'analyse
  Pour analyser l'ensemble de la trace, l'algorithme extrait les paquets un par un puis analyse l'en-tête de ces paquets. Les informations de cet en-tête permettent de déduire l'origine, la destination et le protocole de chaque paquet.
  L'analyse se poursuit ensuite en fonction du protocole, sachant que l'analyse d'un paquet TCP est plus complexe que celle d'un paquet UDP (car les paquets TCP sont classés en flux et on ne fait pas cette opération pour les paquets UDP). Les statistiques pour chaque protocole sont ensuite mises à jour à chaque nouvelle occurence d'un paquet de ce protocole.
  Pour pouvoir caractériser le trafic selon deux niveaux (paquet et flux), il est nécessaire d'associer chaque paquet à un flux. Ainsi, chaque paquet est attribué à un ancien flux et si le flux n'existe pas (paquet SYN rencontré) ce dernier est créé. Il est ensuite mémorisé puis actualisé à chaque paquet jusqu'à ce que le flux s'achève (paquet FIN rencontré). Une fois le flux terminé, il est alors possible de savoir s'il s'agit d'un flux Eléphant, Souris, Tortue ou Libellule (cette répartition ne se faisant que dans le cas des flux TCP) pour pouvoir ensuite caractériser séparément chacun de ces types de flux.
  Pour informer l'utilisateur de l'avancement de l'analyse, la fenêtre de progression est actualisée tous les 5000 paquets. Cette analyse précise le nombre de flux, paquets et octets analysés ainsi que la durée de l'analyse.
  Enfin, dès que l'analyse paquet par paquet est achevée, le fichier trace analysé et les fichiers résultats (débit, inter-paquets, inter-flux...) sont fermés. Les étapes suivantes sont l'enregistrement des derniers résultats (calcul du niveau de corrélation ou de LRD) puis la génération des graphes.
  Le détail de la structure logique retenue pour le squelette du logiciel Zoo ainsi que les différentes bibliothèques développées sont présentés dans la figure 9.1.

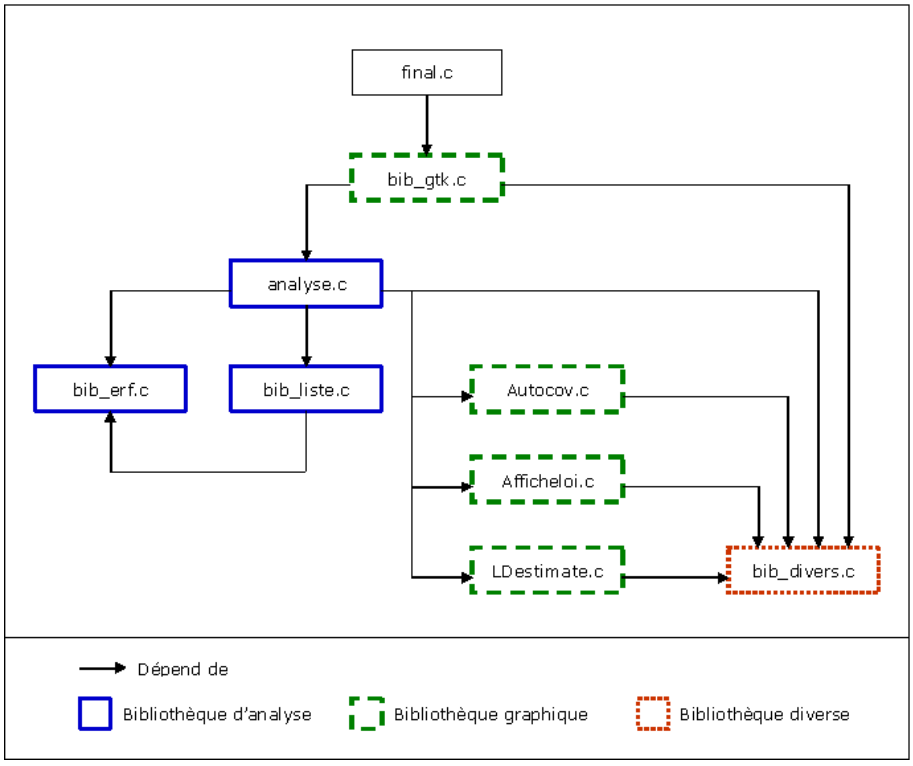


FIG. 9.1 – Structure du logiciel Zoo

Nous allons maintenant détailler le traitement opéré en fonction des différents protocoles rencontrés (TCP ou UDP) et les différentes classes de flux rencontrées (souris, éléphants, tortues ou libellules).

– Caractérisation des flux TCP
Cette caractérisation se fait lorsque le numéro de protocole trouvé dans l'en-tête du paquet (champ protocol) est égal à 6, c'est à dire lorsqu'il s'agit d'un paquet TCP encapsulé dans un paquet IP. Ensuite, l'analyse se déroule en plusieurs étapes détaillées ci-après :

1. Enregistrement de la taille du paquet pour le calcul et l'affichage futur de la distribution des tailles de paquet.
2. Sauvegarde de la durée inter-paquets (il s'agit de la durée séparant l'arrivée de deux paquets successifs du même type) dans la distribution des durées inter-paquets, puis mémorisation de la valeur des inter-paquets dans un fichier annexe (données nécessaire pour le calcul de l'auto-corrélation ou de la LRD) ainsi que de la date des arrivées des paquets.
3. Recherche du flux : pour cela nous vérifions si le paquet appartient à un ancien flux déjà enregistré ou s'il s'agit du premier paquet d'un flux.

   S'il s'agit d'un nouveau flux, nous mettons à jour le compteur de flux, nous sauvegardons la valeur temporelle de l'inter-flux de la même façon que cela a été fait avec l'inter-paquet puis nous sauvegardons les informations sur ce nouveau flux. Cela fait nous ajoutons le nouveau flux à la liste des flux en ayant au préalable ajouté le paquet à la liste des paquets de ce flux. Il est nécessaire de sauvegarder tous les paquets des flux TCP car l'analyse des flux particuliers (Eléphants, Souris, Libellules, Tortues...) n'est possible qu'une fois le flux terminé.

   S'il ne s'agit pas d'un nouveau flux, nous mettons à jour les caractéristiques du flux puis nous ajoutons le paquet à la liste des paquets de ce flux.
4. Evaluation du paquet de fin, pour cela si le paquet analysé est le dernier paquet du flux (bit FIN à 1), on exécute les étapes 5, 6 et 7. Sinon, on passe directement à l'étape 8.
5. Enregistrement de la durée et de la taille du flux dans les distributions correspondantes.
6. Traitement suivant que le flux appartient à la classe des flux Eléphant, Souris, Tortue ou Libellule (cf. point ci-après de la section 9.2.2).
7. Suppression du flux. Pour cela nous supprimons le flux de la liste chaînée des flux puis nous libérons l'espace mémoire (cette étape est très importante pour garantir une vitesse d'exécution du programme raisonable).
8. Mise à jour du compteur du nombre de paquets.
9. Enregistrement du débit, pour cela nous enregistrons le débit dans le fichier résultat à chaque fois que la durée atteint le niveau de granularité souhaitée par l'utilisateur.

– Caractérisation des flux UDP
L'analyse des paquets UDP est à peu de chose près similaire à l'analyse des paquets TCP. Cette caractérisation se fait lorsque le numéro de protocole trouvé dans l'en-tête du paquet (champ protocol) est égal à 17, c'est à dire lorsqu'il s'agit d'un paquet UDP encapsulé dans un paquet IP.
Le processus de caractérisation du paragraphe précédent reste valable pour les étapes 3 à 9 en évitant les étapes 5, 6 et 7 décrites ci-dessus. Néanmoins, certaines particularités différencient l'analyse TCP de l'analyse UDP. Tout d'abord, il n'est pas possible, avec le protocole UDP, de

connaître la fin d'un flux en consultant un champ de l'en-tête. C'est pour cela que nous avons recours à la fonction qui vérifie dans la liste complète des flux UDP s'il existe un flux inactif depuis au moins 900 secondes (soit 15 minutes).
Ensuite, seuls le nombre de paquets, le nombre de flux et la quantité d'octets nous intéressent dans la caractérisation des flux UDP. De ce fait, aucune distribution, aucun calcul de débit et aucune valeur d'inter-paquets ou d'inter-flux ne sont sauvegardées. Ces informations bien qu'intéressantes à analyser ne sont pas à l'heure actuelle extraites des traces. Ces fonctionnalités seront ajoutées dans la version suivante du logiciel.
Enfin, il est inutile pour le protocole UDP de conserver les paquets dans le flux puisque nous n'analysons aucun flux spéciaux de ce protocole comme nous le faisons avec TCP pour les flux Eléphants, Souris... De la même façon que pour le calcul des distribution la décomposition en classe de flux pour le trafic UDP n'est pas implémentée dans cette version du logiciel. Ces fonctionnalités feront partie de la prochaine version de Zoo.

– Caractérisation des flux Eléphants, Souris, Tortue et Libellule
Dans la version actuelle de Zoo, il est possible d'analyser 4 types de flux différents selon leur taille ou leur durée.
On définit ainsi qu'un flux Eléphant comporte au moins 100 paquets et un flux Souris comporte au plus 10 paquets. On définit également qu'un flux tortue dure au moins 900 secondes tandis qu'un flux Libellule dure au plus 2 secondes. Toutes ces données sont les valeurs par défaut du logiciel, elles sont issues de l'article [22] qui définit la méthodologie pour distinguer les différentes classes de flux au sein du trafic Internet. Néanmoins, l'utilisateur a la possibilité de les modifier s'il le souhaite.
La distinction Eléphant/Souris se fait donc en fonction du nombre de paquets. La distinction Tortue/Libellule, quand à elle, se fait en fonction de la durée du flux.
L'analyse de ces flux se fait pour le protocole TCP uniquement et lorsqu'un flux est achevé. En effet, il n'est évidemment pas possible de connaître la durée ou la taille d'un flux tant que ce dernier n'est pas terminé.
Pour tous les types cités ci-dessus la procédure d'analyse est la suivante :

1. Mise à jour des compteurs de paquets et de flux,
2. Ajout des paquets du nouveau flux dans la liste des paquets existants par ordre chronologique.
3. Mise à jour des distributions des tailles de paquets, d'inter-paquets, des tailles de flux et des durées de flux et enregistrements des valeurs dans le fichier résultat,
4. Enregistrement ordonné de la date de fin du flux,
5. Enregistrement des débits dans le fichier résultat.

– Caractérisation des autres flux
Cette caractérisation se fait lorsque le protocole trouvé dans l'en-tête du paquet n'est ni TCP ni UDP. Dans ce cas, l'analyse est très simple puisque nous ne nous intéressons ici qu'au nombre de paquets et au volume d'octets échangés.
Le nombre de flux n'est dans ce cas pas comptabilisé car il est très difficile de définir la fin d'un flux lorsqu'on s'intéresse à plusieurs protocoles simultanément. En effet, il faut dans ce cas, s'intéresser à la sémantique de chaque flux pour déterminer quel type de protocole de niveau transport ou application l'a généré. Cette méthode n'est à l'heure actuelle pas implémentée dans le logiciel Zoo.

FIG. 9.2 – Fenêtre d'introduction

9.2.3 Utilisation du logiciel

Nous allons présenter dans cette section les différentes fonctionnalités de l'interface graphique du logiciel Zoo (générée par l'intermédiaire de la bibliothèque Gtk 2.0 [27]).

- Fenêtre d'introduction
  Au démarrage du logiciel Zoo, une fenêtre d'introduction (cf. figure 9.2) s'ouvre en avant de la fenêtre principale. Cette fenêtre d'introduction présente le logiciel et les auteurs de ce projet et propose de continuer ou de quitter. Si le bouton Non est sélectionné, le programme est fermé tandis qu'un clic sur le bouton Oui ferme la fenêtre d'introduction et rend active la fenêtre principale.
- Fenêtre principale
  La fenêtre principale (cf. figure 9.3) est composée de :
  - Une barre de menu : cette barre permet d'accéder à toutes les fonctionnalités du logiciel. Elle inclut les menus Fichier, Analyse et Aide détaillés dans les sections qui suivent.
  - Une barre d'outils : cette barre contient les fonctionnalités les plus utiles du logiciel à savoir Nouveau, Ouvrir, Enregistrer, Analyser, Afficher, Options et Quitter.
  - Un "notebook" : cette zone permet d'afficher les graphes et les résultats statistiques obtenus après analyse.
  - Une barre d'état : cette barre permet d'afficher des messages pour l'utilisateur (aide, message d'erreur, etc.).
- Le Menu Fichier
  Le menu fichier (cf. figure 9.4(a)) permet de gérer les projets et de quitter le programme. Il est composé des sous menus Nouveau, Ouvrir, Enregistrer, Fermer et Quitter.

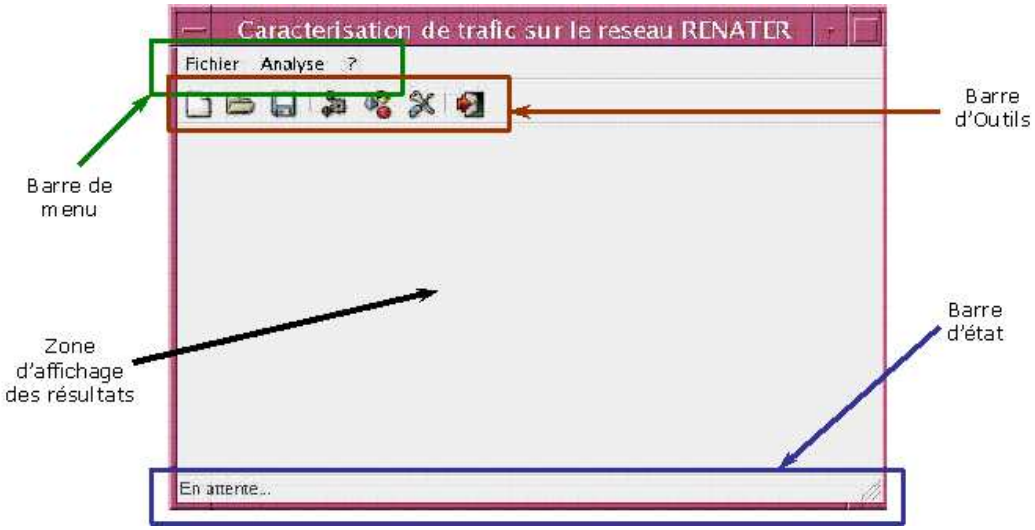
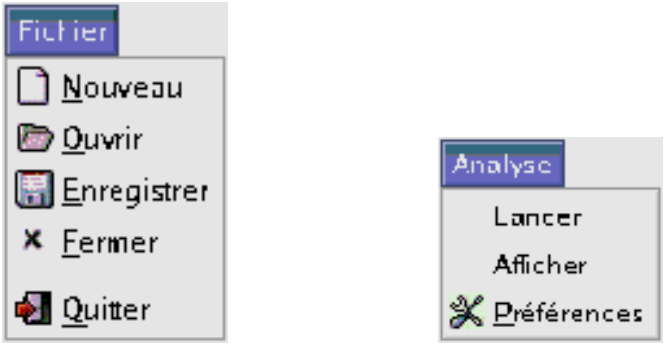


FIG. 9.3 – Fenêtre principale



(a) Menu Fichier

(b) Menu Analyse

FIG. 9.4 – Détails de la barre de menu principale

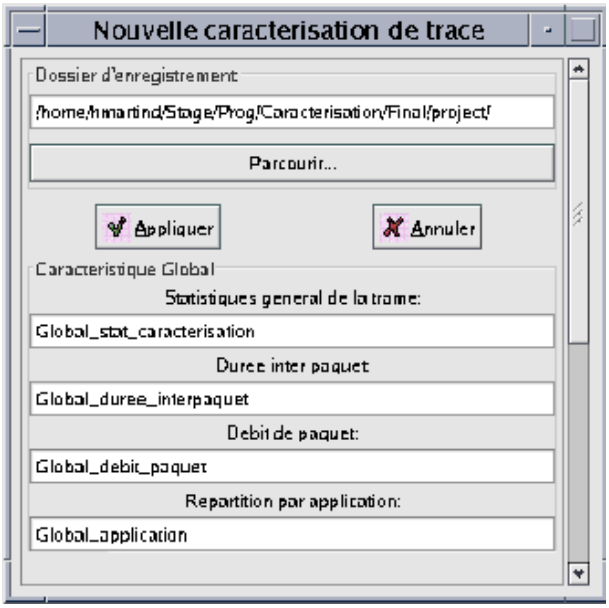


FIG. 9.5 – Fenêtre nouveau

Nouveau

Le sous menu Nouveau permet de créer un nouveau projet en spécifiant le répertoire et les noms de fichier dans lesquels seront enregistrés les résultats. En outre, la création d'un nouveau projet réinitialise toutes les options.

En cliquant sur Nouveau, et si aucun autre projet n'est déjà ouvert, la fenêtre de la figure 9.5 s'ouvre alors.

En haut de cette fenêtre, un champ permet d'entrer le nom du répertoire (dans lequel les résultats seront sauvegardés) soit directement, soit à l'aide du bouton Parcourir qui permet de sélectionner un répertoire dans l'arborescence. Ensuite, l'utilisateur peut entrer les noms des fichiers dans lesquels seront enregistrés les résultats. Par commodité, toutes les entrées sont déjà remplies mais l'utilisateur a la possibilité de changer les noms de fichiers si les noms par défaut ne lui conviennent pas.

Enfin, l'utilisateur a la possibilité de cliquer sur Appliquer pour fermer la fenêtre et mémoriser les fichiers ou sur Annuler pour revenir à la fenêtre principale sans sauvegarder les noms de fichier.

Ouvrir

Le sous menu Ouvrir permet d'ouvrir un projet déjà sauvegardé. Les données sauvegardées sont le nom du répertoire de sortie, les noms des fichiers et toutes les options modifiables par l'utilisateur. Une fois le projet ouvert, toutes les valeurs sont initialisées avec celles contenues dans le fichier, il est alors possible de lancer une nouvelle analyse ou d'afficher des anciens résultats.

En sélectionnant le sous menu Ouvrir, et si aucun autre projet n'est ouvert, une fenêtre s'ouvre. L'utilisateur a alors la possibilité de choisir le nom du projet.

Enregistrer

Le sous menu Enregistrer permet d'enregistrer un projet ouvert. Les données sauvegardées sont le nom du répertoire de sortie, les noms des fichiers résultats et toutes les valeurs des options modifiables par l'utilisateur.

Fermer

Le sous menu Fermer permet de fermer un projet ouvert et de réinitialiser l'affichage de la fenêtre principale. La fermeture d'un fichier permet à l'utilisateur de pouvoir ouvrir un autre projet car l'ouverture simultanée de deux projets est impossible.

Quitter

Le sous menu Quitter permet de quitter le programme. Un message de confirmation apparaît alors pour s'assurer du choix de l'utilisateur.

– Le Menu Analyse

Le menu Analyse (cf. figure 9.4(b)) permet de gérer l'analyse d'un projet. Il est composé des sous menus Lancer, Afficher et Préférences.

Lancer

Le sous menu Lancer (cf. figure 9.6) permet de lancer une nouvelle analyse à condition qu'un projet soit ouvert au préalable par l'utilisateur. Un clic sur ce menu provoque l'affichage d'une fenêtre permettant de choisir les paramètres principaux de l'analyse qui sont :

– Le nom de la trace à analyser : il peut être choisi directement en entrant le nom dans le champ de saisie ou bien en passant par un sélectionneur de fichier pour choisir une trace dans l'arborescence.
– Les graphes à afficher : en cliquant sur le "checkbox" associé, l'utilisateur peut désactiver l'analyse d'un type de flux. La désactivation permet de sauter l'analyse, la caractérisation et l'affichage d'un type de flux (cette possibilité est utile pour accélérer l'analyse globale par exemple).
– Les paramètres principaux de l'analyse.

Une fois les paramètres d'analyse sélectionnés, l'utilisateur peut choisir de lancer l'analyse en cliquant sur Appliquer ou de fermer simplement la fenêtre avec le bouton Annuler.

Afficher

Le sous menu Afficher permet d'afficher les graphes et les statistiques d'une ancienne analyse à condition qu'un projet soit ouvert (l'ouverture d'un projet permet en effet de connaître les noms des fichiers et les paramètres de l'analyse) et que tous les graphes au format *.PNG[4] existent dans le répertoire courant.

Préférences

Le sous menu Préférences permet d'afficher et de modifier les paramètres secondaires d'analyse. Les options sont divisées en deux catégories :

– Les options d'analyse (cf. figure 9.7) : l'utilisateur peut définir les limites caractéristiques des flux qu'il souhaite analyser, les applications supplémentaires qu'il souhaite caractériser et la largeur d'analyse pour les calculs d'auto corrélation.
– Les options diverses (cf. figure 9.8) : l'utilisateur peut définir le coefficient de grossissement appliqué dans l'affichage des graphes et les noms des fichiers à conserver lors de la sortie du logiciel.

– Le menu Aide (' ?')

Ce menu permet d'accéder à l'aide utilisateur ainsi qu'aux informations diverses du logiciel.

– Après l'analyse

[4]Le format PNG a été choisi pour sa gratuité et son caractère ouvert.

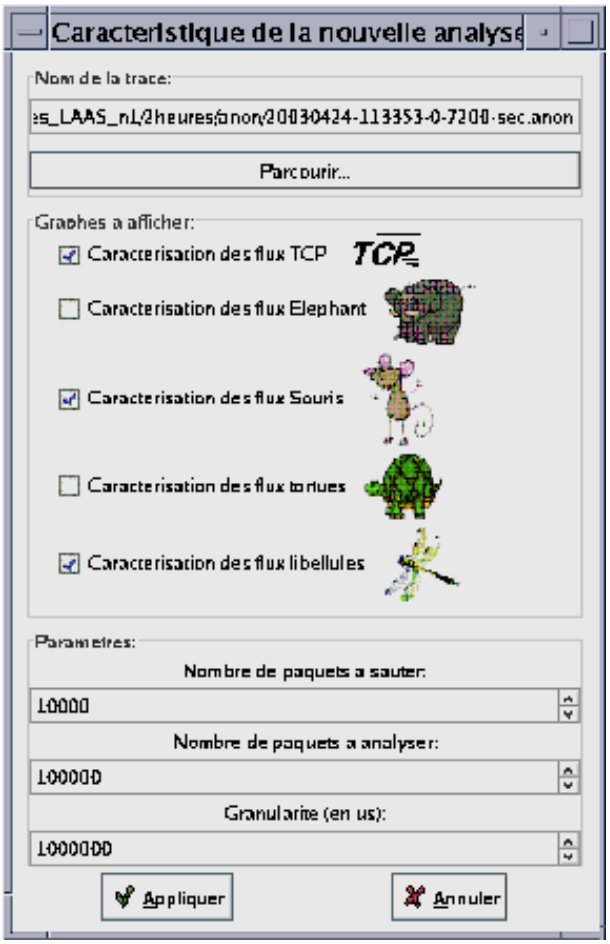


FIG. 9.6 – La fenêtre de lancement d'une analyse

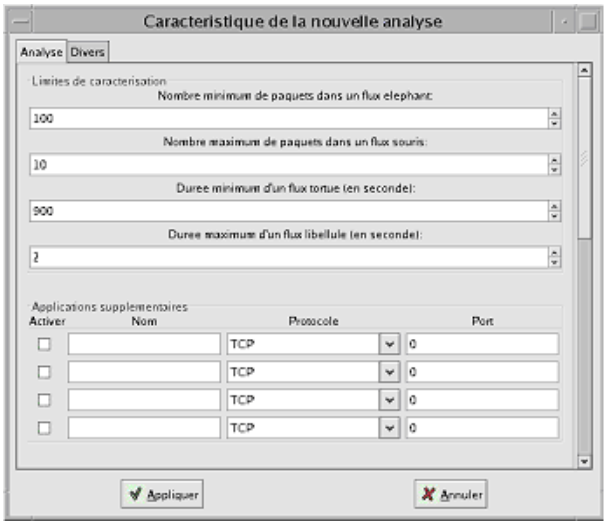


FIG. 9.7 – Les options d'analyse

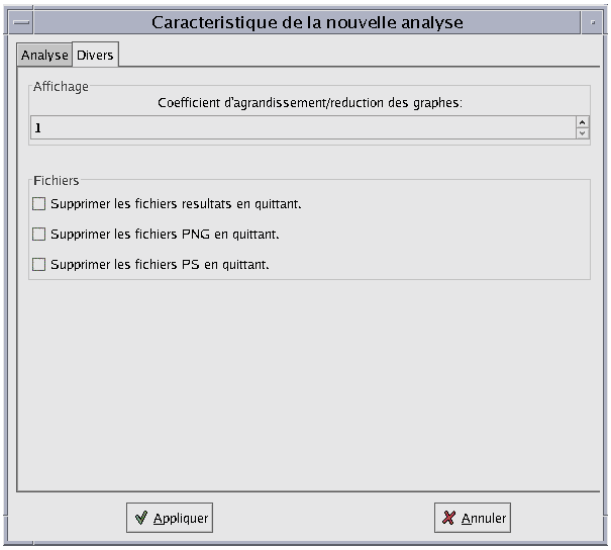


FIG. 9.8 – Les options diverses

Une fois l'analyse terminée, le notebook est mis à jour et les graphes sélectionnés dans les paramètres principaux d'analyse s'affichent. Les figures 9.9 et 9.10 ci-dessous montrent un exemple typique d'affichage après analyse.
Les premières pages du notebook présentent les statistiques globales de l'analyse puis présentent les résultats obtenus par type d'application. Les applications utilisateurs sont ajoutées à la suite des applications de base. Les pages suivantes affichent les graphiques générés après analyse en les regroupant par type (TCP, Eléphant, Souris, Tortue et Libellule) et par niveau (flux et paquet). Si l'analyse révèle un nombre insuffisant de paquets ou de flux d'un type particulier, les graphes de ce type ne peuvent pas être générés. Dans ce cas, un message d'erreur avertit l'utilisateur et les pages du type en question ne sont pas ajoutées au notebook.

9.3 Exemple de décompositon : caractérisation du trafic Souris vs. Elephant

Nous allons illustrer les possibilités du logiciel Zoo sur un exemple simple. Nous avons analysé une trace collectée sur le lien du LAAS-CNRS en divisant le trafic en deux classes de flux relativement bien connues : les flux souris et les flux éléphants. Nous allons nous intéresser aux différences qui peuvent être mises en évidence pour chacune de ces classes en considérant les paramètres suivants :

- Distribution des arrivées de flux,
- Distribution des arrivées de paquets,
- Niveau de corrélation des arrivées de flux,
- Niveau de corrélation des arrivées de paquets,
- Niveau de LRD des arrivées de flux,
- Niveau de LRD des arrivées de paquets.

Dans cette partie, nous allons commencer par caractériser les flux souris. Ensuite, nous nous intéresserons aux flux éléphants. Pour la caractérisation de chaque classe de flux, nous avons considéré deux niveaux d'analyses différents (flux puis paquets). Cette approche a l'avantage d'aboutir à des modélisations différentes selon le besoin. En effet, si on se place au niveau des couches hautes (le niveau applicatif par exemple) il est plus intéressant d'étudier les comportements des flux. Par contre, si l'effort de modélisation vise le dimensionnement des ressources, l'amélioration des mécanismes protocolaires de la couche transport (mécanismes de contrôle de congestion, de reprise des pertes, etc.), l'évaluation des performances et l'amélioration de la QoS, il peut sembler plus judicieux d'étudier le niveau paquet.

Caractérisation du trafic Souris

- Caractérisation des flux souris
  Les flux souris ont été définis comme des flux contenant un petit nombre de paquets [16]. Un exemple de ces flux est donné par les connexions ouvertes pour télécharger différents objets appartenant à une page web. Il est intéressant de modéliser les arrivées de ces flux car elles sont très liées au niveau applicatif et donnent une idée plus précise sur les comportements des utilisateurs et des applications.
  Loi d'arrivée des flux souris

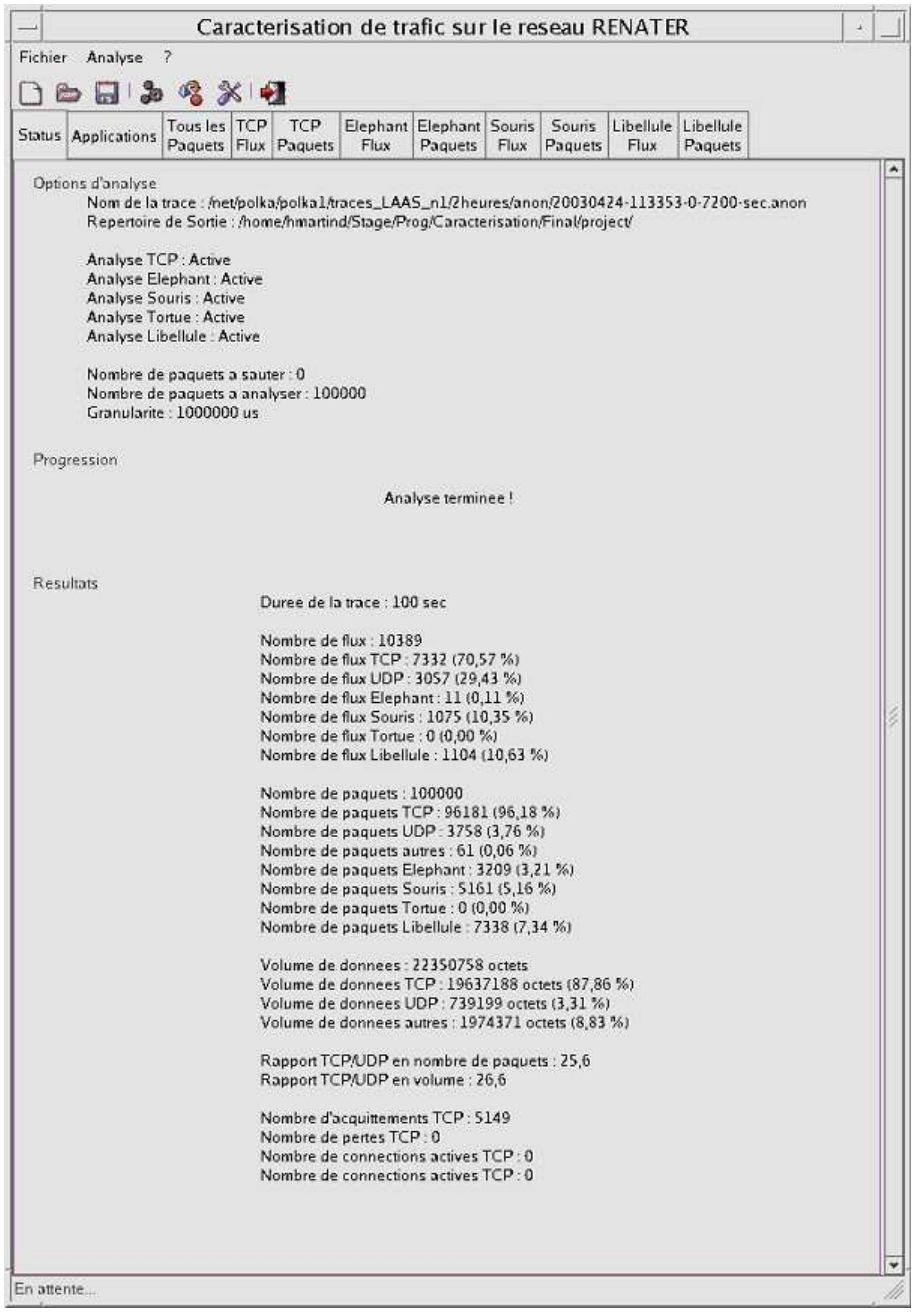


FIG. 9.9 – Affichage des statistiques de l'analyse

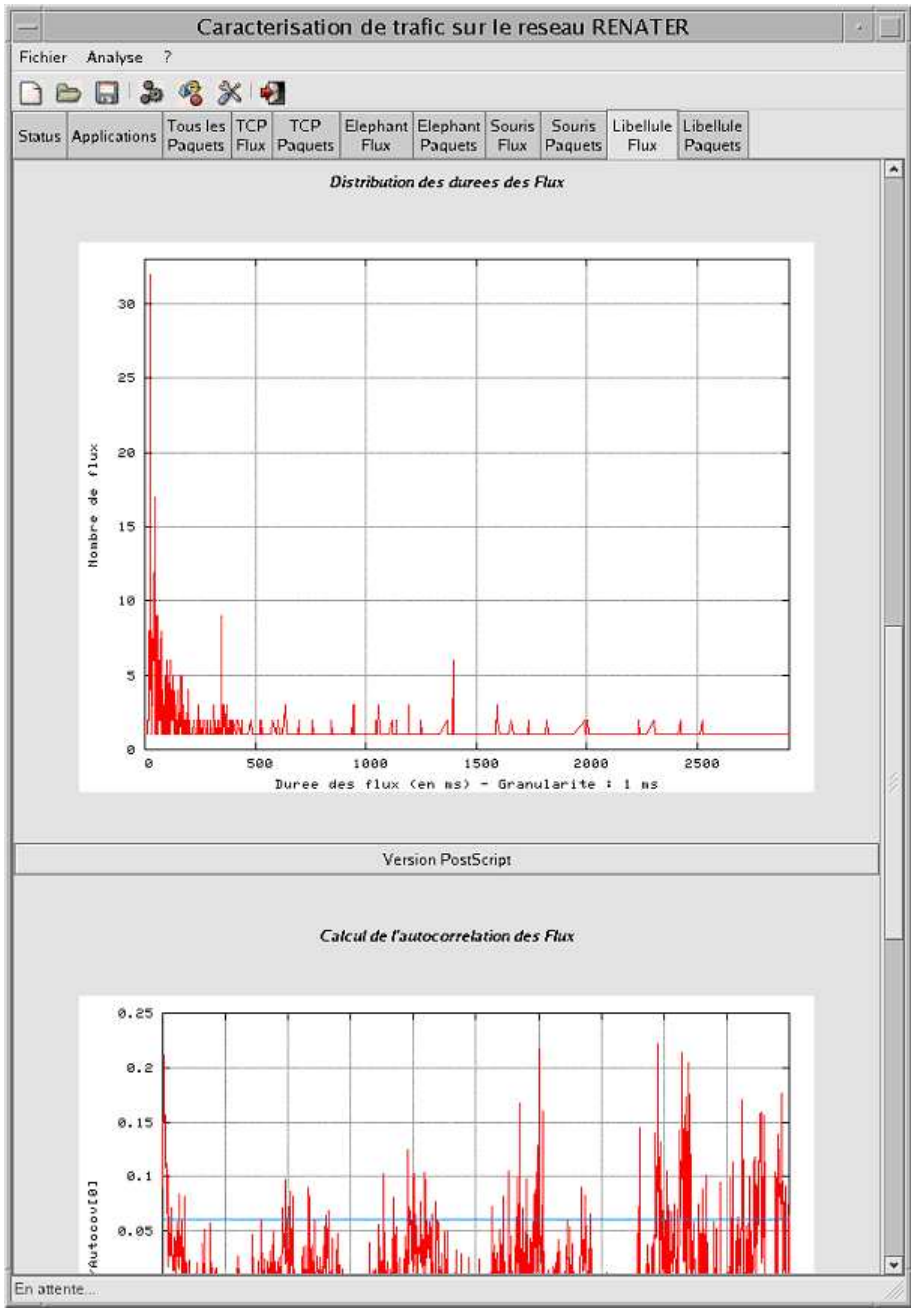


FIG. 9.10 – Affichage des graphiques générés après analyse

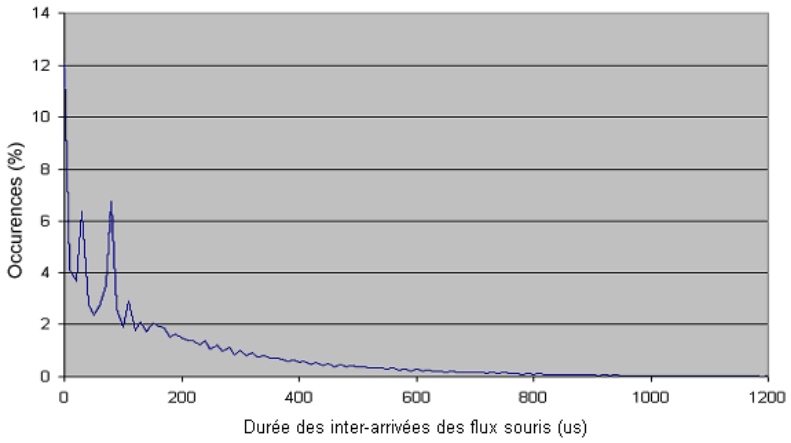


FIG. 9.11 – Distribution des arrivées des flux souris (Granularité 50 us)

La première étape de notre caractérisation consiste en la détermination de la distribution des arrivées des flux souris (cf. figure 9.11)
Nous remarquons sur cette figure que le plus grand pourcentage des flux souris ont des inter-arrivées petites ($\leq$ 100 us). Par contre, très peu de flux arrivent avec des inter-arrivées trop grandes ($\geq$ 1 ms). Cependant, même si cette distribution a une allure similaire à la loi exponentielle, nous ne pouvons pas affirmer au stade actuel de notre étude que la loi d'arrivée des flux souris est assimilable à une loi exponentielle. Bien au contraire, une analyse plus détaillée, présentée dans la suite, va montrer que la distribution des arrivées des flux suit une loi à décroissance lente.
La présence de queue lourde dans les distributions des tailles de fichiers, des temps de transferts ou des arrivées des flux, est considérée dans de nombreuses publications comme une des causes présumées de la LRD (voir [29] [28] [17] [11] pour détails). Il est donc intéressant de vérifier si la distribution des arrivées des flux souris présente une queue lourde. La représentation QQ-Plot est une technique simple pour faire ce type d'analyse. En effet, la figure 9.12 fait apparaître un éloignement visible de la fonction quantile par rapport à la ligne de référence. Ceci prouve que les deux séries de données (les inter-flux souris, et les inter-arrivées exponentielles) sont régies par des distributions différentes.

Structure d'auto-corrélation des paquets des flux souris

La fonction d'auto-corrélation (cf. figure 9.13) de la loi d'arrivée des flux souris permet une mesure qualitative de la dépendance à long terme. En effet, cette figure fait apparaitre une structure complexe de la fonction d'auto-corrélation (fortes oscillations et décroissances lentes). De plus, toutes les valeurs n'appartiennent pas aux bornes de confiances gaussiennes ($\pm$ 0,0038). Ces constatations nous permettent donc de mettre en évidence de la corrélation et donc de la dépendance dans les arrivées des flux souris.
Cette dépendance n'est pas surprenante car le trafic souris est très lié au trafic web et les mécanismes du protocole HTTP 1.0 en sont la cause principale. Le trafic web est majoritairement composé de flux souris correspondants aux transferts d'objets graphiques, d'images, de texte, d'Applets Java, etc. Concrètement, lorsqu'un utilisateur navigue sur le Web, le protocole HTTP 1.0 ouvre autant de connexions que d'objets dans la page visitée. Il crée ainsi de

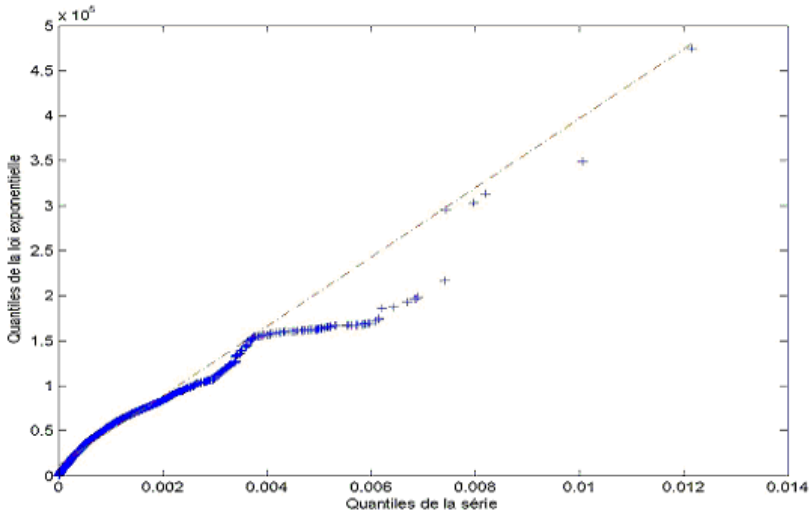


FIG. 9.12 – Représentation QQ-Plot de la loi d'arrivée des flux souris

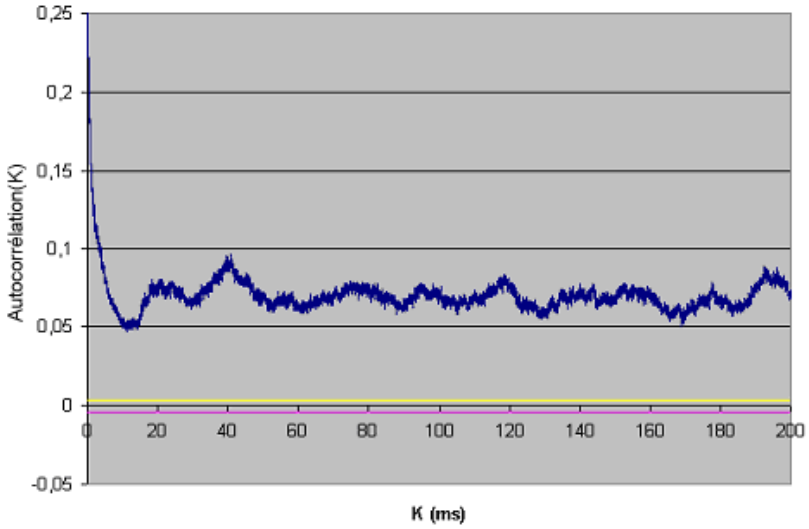


FIG. 9.13 – Fonction d'auto-corrélation de la loi d'arrivée des flux souris

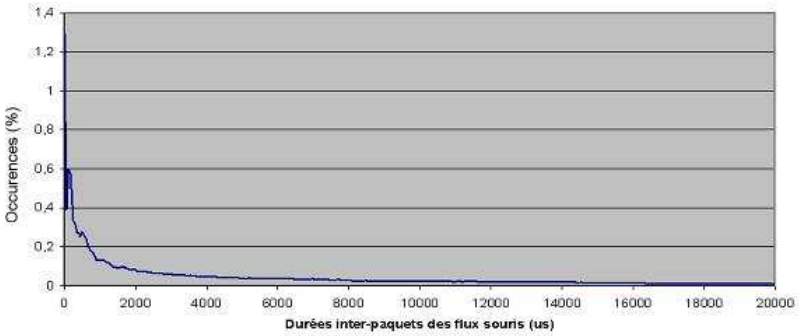


FIG. 9.14 – Distribution des arrivées des paquets souris (Granularité 50 us)

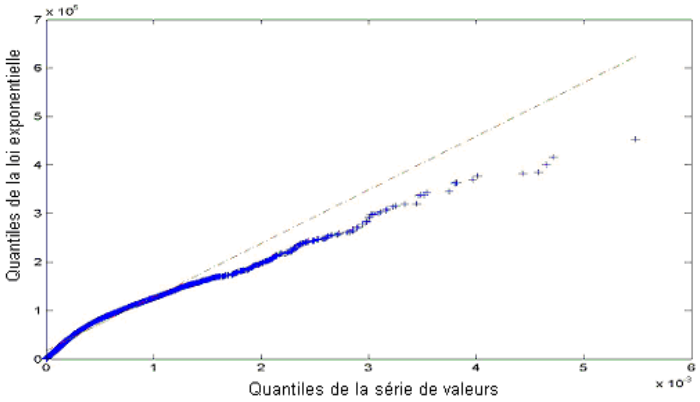


FIG. 9.15 – Représentation QQ-Plot de la loi d'arrivée des paquets appartenant à la classe des flux souris

la dépendance entre chaque connexion ouverte pour télécharger chaque élément d'une même page, qui sont en majorité des flux souris.

– Caractérisation des paquets souris

Loi d'arrivée des paquets souris

Après avoir isolé les flux souris et étudié leur loi d'arrivée, nous nous sommes intéressés aux paquets composant chacun de ces flux. Pour cela, la figure 9.14 illustre la distribution des temps d'inter-arrivées des paquets en microsecondes pour cette catégorie de flux.

Il est évident que la distribution des inter-paquets ne nous délivre pas les informations nécessaires pour caractériser ce type de trafic. Il est indispensable alors d'affiner l'analyse afin de mieux capturer la complexité du trafic.

La technique basée sur le calcul des quantiles de la série est sollicitée pour vérifier le comportement de cette série de données par rapport à une série théorique qui provient d'une distribution exponentielle.

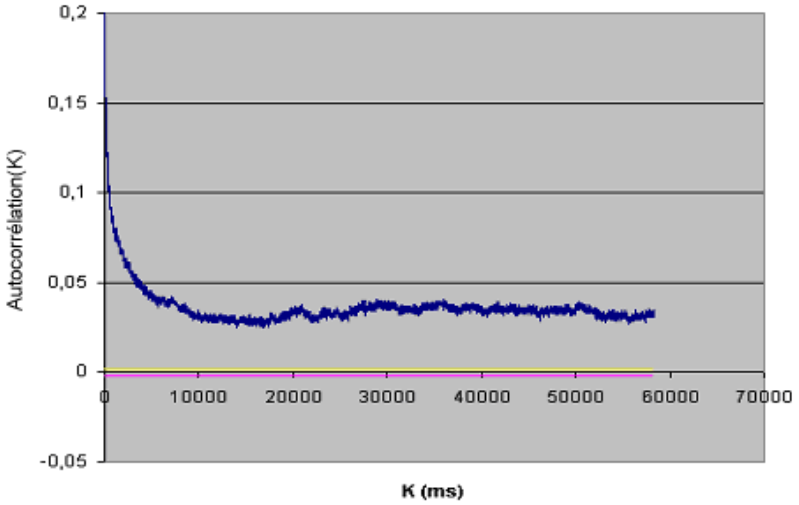


FIG. 9.16 – Fonction d'auto-corrélation de la loi d'arrivée des paquets souris

La figure 9.15 montre un éloignement progressif par rapport à la ligne de référence (la première bissectrice). Ce décalage est moins important que dans le premier cas (cf. figure 9.12) mais il est suffisant pour signaler une décroissance plus lente que la distribution exponentielle.

Structure d'auto-corrélation des paquets des flux souris

La fonction d'auto-corrélation illustrée par la figure 9.16 se caractérise par un comportement complexe (décroissance lente) et toutes les valeurs sont en dehors des bornes gaussiennes ($\pm$ 0,0020). Cette corrélation entre les paquets souris prouve la présence de dépendance entre les paquets.

Paramètre de Hurst et méthode de la transformée en ondelettes

La mesure du facteur de Hurst par la méthode des ondelettes (cf. détails dans [4]) donne le résultat suivant : H=0,622. Ce facteur supérieur à 0,5 indique la présence de la LRD dans le trafic des flux souris. De plus, cette valeur nous permettra ultérieurement de comparer quantitativement le degré de dépendance longue introduite par chaque type de flux. Le diagramme permettant l'étude des lois d'échelles est représenté sur la figure 9.17.

Nous soulignons ici que la pente de la courbe est plus forte pour les octaves entres 8 et 12 que pour les échelles les plus longues (octave 11 à 16). Cela montre que la dépendance est surtout présente à court terme pour les arrivées des paquets de la classe des flux souris. Nous pouvons conclure ici, que les arrivées des flux souris sont extrêmement corrélées et que la dépendance au niveau des paquets souris se manifeste plus sur du court terme. Nous pouvons également dire que la LRD est présente mais de façon très réduite au niveau paquet. Ainsi, ce sont les résultats de caractérisation obtenus pour les arrivées de flux qui mettent en évidence le plus de corrélation et de dépendance pour l'analyse de la classe des flux souris.

Caractérisation du trafic Elephant

– Caractérisation des flux Elephants

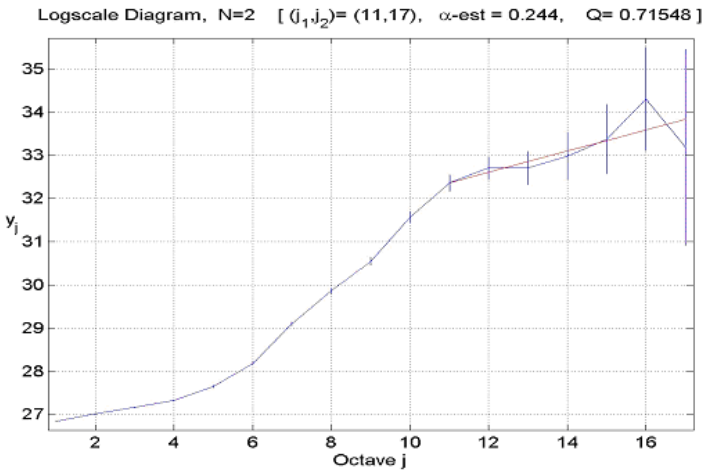


FIG. 9.17 – Diagramme logscale obtenu par la méthode des ondelettes appliquée à la série d'inter-paquets des flux souris

Ces dernières années, nous avons assisté à une mutation profonde du trafic Internet. L'émergence d'applications "pair à pair", par exemple, associée à l'augmentation des capacités des liens de l'Internet a eu un impact sur la taille des fichiers transférés. Ainsi, le trafic comporte de plus en plus de flux longs (les éléphants formés de fichiers audio, vidéo...). Il est donc indispensable de chercher à mieux comprendre le comportement de ces flux qui représentent aujourd'hui la plus grande partie du volume du trafic total.

Loi d'arrivée des flux éléphants

La figure 9.18 illustre la distribution des durées inter-flux éléphants. Pour vérifier si la loi d'arrivée des flux éléphants s'approche d'une loi exponentielle nous procédons au test QQ-Plot sur cette série.

La représentation QQ-Plot de la figure 9.19 montre que le nuage de points générés par les fonctions quantiles ne s'éloigne pas trop de la droite de référence ; au contraire, la première partie de la courbe est quasiment linéaire. Ceci veut dire que la décroissance de cette distribution peut être assimilée à une décroissance exponentielle.

Structure d'auto-corrélation des flux éléphants

La fonction d'auto-corrélation appliquée sur le trafic des flux éléphants montre que notre série de valeurs est très faiblement corrélée. En effet, la majorité des points de la courbe sont situés entre les deux bornes gaussiennes ($\pm$ 0,048).

Notre analyse sur les flux éléphants montre que la loi d'arrivée des flux éléphants semble donc suivre une loi poissonnienne (avec une distribution exponentielle et manifeste peu de dépendance).

Ce résultat n'est en fait pas surprenant. En effet, nous avons déjà dit que le trafic éléphant peut correspondre à des transferts de gros fichiers. Il s'agit donc de téléchargements par des utilisateurs de fichiers par exemple audio ou vidéo. Ces flux dépendent essentiellement du comportement des utilisateurs et sont généralement exécutés en tâche de fond. Ils sont donc très peu dépendants entre eux.

– Caractérisation des paquets des flux éléphants

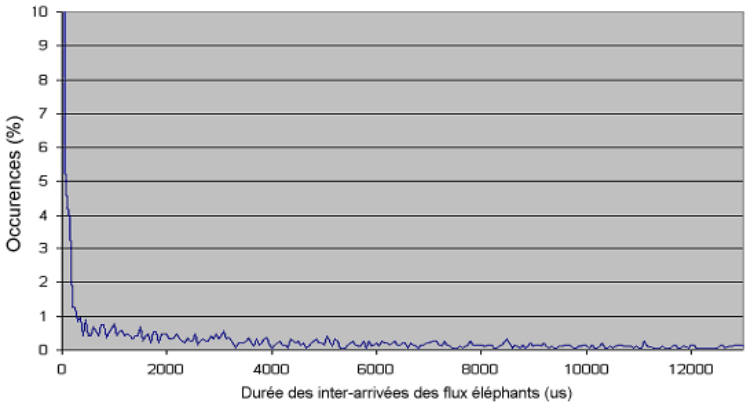


FIG. 9.18 – Distribution des flux elephants (Granularité 50 us)

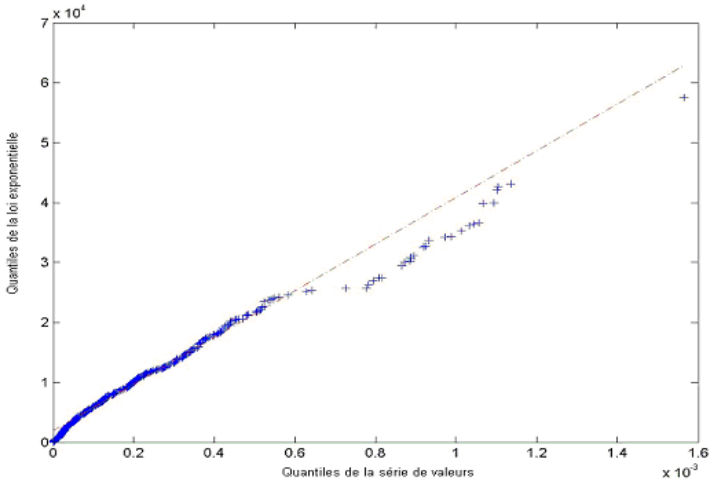


FIG. 9.19 – Représentation QQ-Plot de la loi d'arrivée des flux éléphants

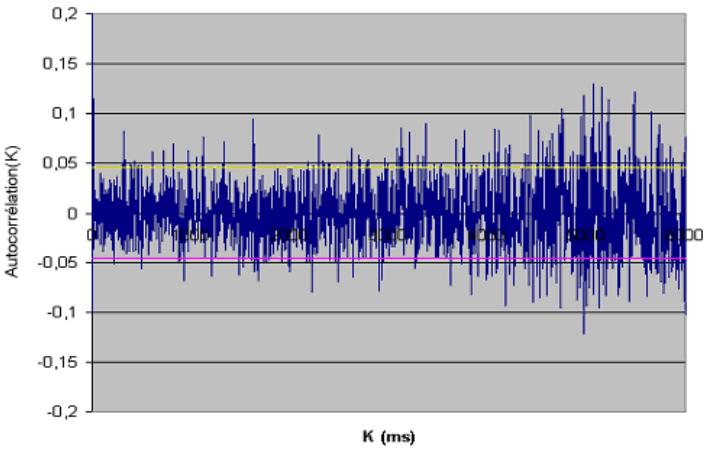


FIG. 9.20 – Fonction d'auto-corrélation de la loi d'arrivée des flux éléphants

Loi d'arrivée des paquets éléphants

Le test QQ-Plot (cf. figure 9.22) montre que les points s'éloignent visiblement de la droite de référence ce qui prouve que la loi d'arrivée (cf. figure 9.21) ne peut être approchée par une loi exponentielle mais plutôt par des lois de puissance.

Structure d'auto-corrélation des paquets des flux éléphants

La fonction d'auto-corrélation (cf. figure 9.23) prouve l'existence d'une corrélation très forte (tous les points sont en dehors des bornes gaussiennes et le comportement de la fonction manifeste une décroissance très lente).

Paramètre de Hurst et évaluation du niveau de LRD

Le diagramme (cf. figure 9.24) généré par la méthode de la transformée en ondelettes permet d'analyser les lois d'échelles régissant le trafic.

Dans la figure 9.24, on distingue deux régimes distincts : le premier concerne les petites échelles de l'octave 1 à l'octave 8 . Le second régime concerne les octaves les plus élevées (octave 8 à 18). Ces deux régimes peuvent être assimilés à deux droites de pentes visiblement différentes. Lorsqu'on s'intéresse à la LRD dans le trafic il faut se focaliser sur les octaves les plus élevées (supérieures à 5). Nous observons sur ce diagramme un alignement des points de la courbe pour les octaves entre 8 et 18. Cet alignement indique la présence d'une forte LRD représentée par la droite et exhibe un comportement en loi de puissance. Nous tenons aussi à signaler que la présence de deux régimes (appelé phénomène de biscaling) doit être prise en compte lors de la modélisation. En effet, un point intéressant serait de pouvoir isoler les deux régimes et appliquer un modèle pour chacun d'eux.

La valeur du facteur de Hurst estimée par cette méthode est H=0,840. Cette valeur est particulièrement significative car elle nous permet de constater que les paquets des flux éléphants sont responsables d'une grande partie de la LRD dans le trafic. Tout d'abord, on observe un facteur de Hurst largement supérieur à 0,5. Ensuite, cette valeur est très proche du paramètre du trafic global observé dans le trafic global (H=0,844). Rappelons que le même facteur pour les paquets souris est largement moins important H=0,622. Cela nous pousse à nous interroger sur la contribution des paquets des éléphants à la présence de la LRD dans le trafic.

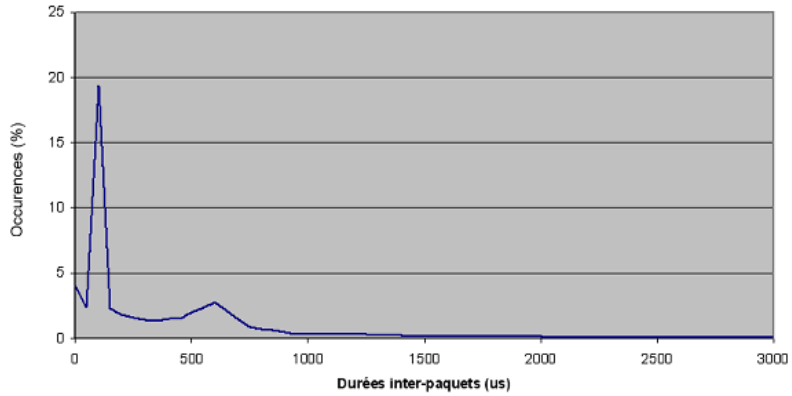


FIG. 9.21 – Distribution des arrivées des paquets éléphants (Granularité 50 us)

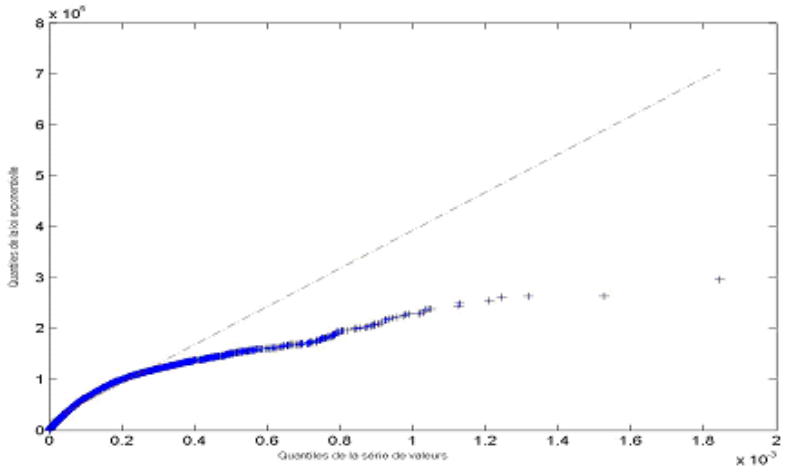


FIG. 9.22 – Représentation QQ-Plot de la loi d'arrivées des paquets appartenant à la classe des flux éléphants

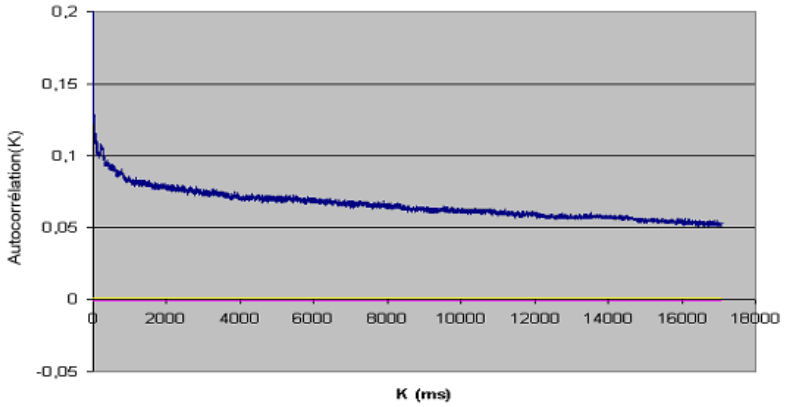


FIG. 9.23 – Fonction d'auto-corrélation de la loi d'arrivées des paquets éléphants

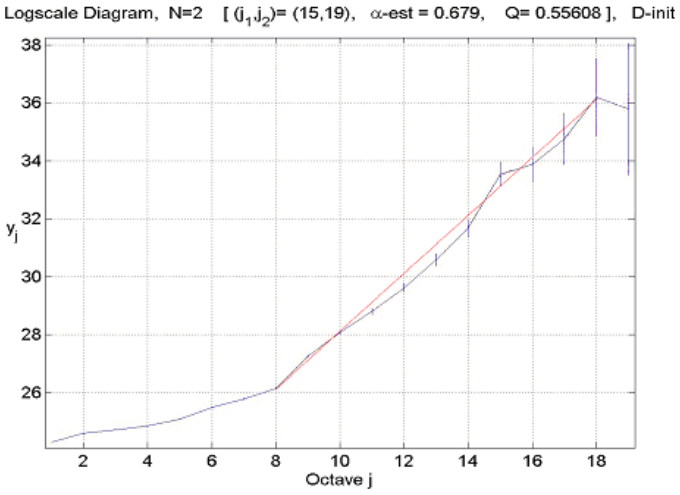


FIG. 9.24 – Diagramme logscale obtenu par la méthode des ondelettes appliquée à la série d'inter-paquets des flux éléphants

Une des explications de la dépendance à long terme au niveau des paquets éléphants est la suivante. Nous avons déjà mentionné que les flux éléphants correspondent généralement à des transferts de gros fichiers audio ou vidéo. Les paquets composant ces flux ont plus de chance de suivre les mêmes chemins et de traverser les mêmes nœuds. En plus, les congestions et les pertes survenues lors des transferts de ces paquets peuvent introduire de la dépendance longue mémoire. En effet, les mécanismes de contrôle de congestion de TCP tels que le “Slow Start” et le “Congestion Avoidance”) peuvent créer de la dépendance entre les paquets et les pertes de paquets d’un même flux [3] [15] [6].
Les résultats de la caractérisation des paquets des flux éléphants soulignent l’importance de la LRD dans ce type de flux. Nous pouvons même penser que la LRD dans le trafic Internet est due en majorité aux paquets des flux éléphants. Cette réflexion est très importante car elle va nous permettre de simplifier l’étape suivante de modélisation du trafic en nous permettant de concentrer l’effort de modélisation sur cette classe de flux.

9.3.1 Conclusion du travail de caractérisation Souris vs. Elephants

Nous avons illustré l’utilisation de la méthode de caractérisation du trafic Internet au travers d’une décomposition simple des flux en deux classes (souris et éléphants) ce qui nous a permis d’étudier chacun de ces flux à part. Afin d’éviter la complexité d’une analyse globale, nous avons procédé à une analyse du trafic à deux niveaux (le niveau flux et le niveau paquet). A l’issue de ce travail de caractérisation, nous avons observé les phénomènes suivants :

1. la dépendance est bien marquée au niveau flux pour la classe des souris,
2. la dépendance au niveau paquet pour les flux souris est surtout marquée sur du court terme (ce résultat mis en évidence par l’analyse semblait recevable a priori étant donné que les flux souris sont des flux très courts pour lesquels les phénomènes de corrélation et de dépendance à long terme ne devaient pas être très prononcés),
3. les arrivées des flux éléphants semblent être assimilables à un processus de Poisson et la LRD au niveau des arrivées des flux éléphants n’est pas importante par rapport à celle des flux souris. Ainsi le comportement du niveau flux pour la classe des éléphants s’apparente assez à un processus de Poisson,
4. l’analyse au niveau paquet des flux éléphants met en évidence plus de dépendance à long terme pour cette classe par rapport à la LRD présente dans le trafic global. Ceci est lié au volume des paquets éléphants qui sont majoritaires dans le trafic Internet et qui apportent une très forte contribution à la LRD globale du trafic.

Ces résultats mettent en évidence la diversité des caractéristiques des classes de flux étudiées. En effet, nous avons remarqué que chaque type de flux possède des propriétés et des caractéristiques différentes : ainsi, les arrivées des flux souris sont dépendantes à long terme alors que celles des flux éléphants sont faiblement dépendantes et inversement au niveau paquet. Cette constatation est très importante pour la modélisation. En effet, elle nous permet d’imaginer une approche visant à trouver un modèle pour chaque type de flux en tenant compte des caractéristiques spécifiques à chaque classe.

9.4 Conclusion du travail de caractérisation Zoologique

La partie qui se termine a successivement mis en évidence une méthode de caractérisation du trafic basée sur sa décompostion en différentes classes de trafic et illustré au travers d'un exemple l'intérêt et les possibilités offertes par cette méthode.

En effet, le trafic Internet est par nature difficile à caractériser dans sa globalité, en particulier à cause de la complexité des caractéristiques traditionnement mises en évidence lors des phases de d'analyse simple (cf. section 8). La méthode proposée dans ce rapport permet de casser cette complexité en se focalisant sur des sous-ensembles du trafic qui possèdent des caractéristiques plus intéressantes car plus simples. Un exemple de ces caractéristiques a été développé au travers de la caractérisation du trafic en deux classes, souris et éléphants, qui a permis de mettre en évidence par exemple un comportement des flux éléphants assimilable à un processus poissonnien et à l'inverse de faire ressortir les problèmes de corrélation et de dépendance à long terme au niveau des flux souris ou des paquets des flux éléphants.

Bien sur, les bénéfices apportés par cette analyse sont pour le moment insuffisants pour proposer un modèle complet du trafic Internet mais la démarche qui a été initiée reste très prometteuse pour l'avenir de la communauté réseau pour ce qui concerne les problèmes de modélisation, d'optimisation ou encore d'évaluation de performance.

Chapitre 10

Analyse du lien entre LRD et oscillations

10.1 Oscillations et éléphants

Les mesures de métrologie sur les liens de l'Internet, présentées dans ce rapport ou mises en évidence dans d'autres projets, révèlent la présence d'oscillations dans le trafic Internet [**?**]. Un exemple de trafic observé sur un lien Internet est donné dans la figure 28.1. Celle-ci compare le trafic Internet actuel avec un modèle simple de trafic : le modèle de Poisson qui était il y a quelques années l'un des modèles supposés de l'Internet. En fait, les courbes de trafic doivent se lisser lorsque la granularité de l'observation augmente. C'est ce qui est représenté dans la figure 28.1 où pour chaque trafic (Internet actuel et Poissonien) l'amplitude des oscillations décroît lorsque la granularité d'observation est plus importante. Sur cette figure, on note aussi la différence entre les deux types de trafic : pour une granularité d'observation importante (1 seconde par exemple), l'amplitude des oscillations du trafic Internet est plus importante et se lisse moins vite que pour celles du trafic Poissonien.

Certaines analyses du trafic Internet réalisées dans le cadre de récents projets de métrologie ont montré que ces oscillations étaient le résultat de la présence de LRD[1] et / ou d'auto-similarité dans le trafic [19]. Ces phénomènes ont plusieurs causes, notamment les mécanismes de contrôle de congestion, et tout particulièrement ceux de TCP qui est le protocole dominant dans l'Internet [17]. Parmi tous les mécanismes de TCP, il est clair que son système de contrôle en boucle fermée introduit de la dépendance dans le trafic étant donné que les acquittements dépendent de la réception d'un paquet, et que tous les autres paquets de ce flux dépendent de cet acquittement. De la même façon, les deux mécanismes de TCP (slow-start et congestion avoidance), introduisent de la dépendance entre les paquets de différentes fenêtres de contrôle de congestion. Et naturellement, la notion d'émission en rafale des sources TCP ajoutée à la LRD permettent d'expliquer la présence d'oscillations qui se répètent à toutes les échelles de temps dans le trafic global. En généralisant ces observations, nous pouvons affirmer que tous les paquets d'un flux sont dépendants les uns des autres. De plus, avec l'augmentation des capacités de l'Internet permettant aux utilisateurs d'échanger des fichiers de plus

[1] LRD : Long Range Dependence

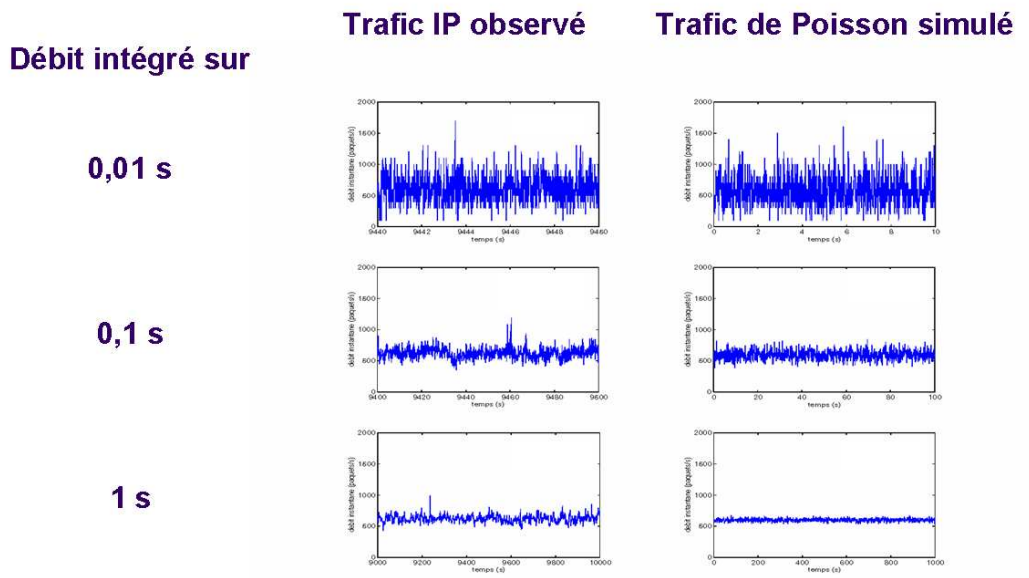


FIG. 10.1 – Comparaison entre les oscillations d'un trafic Internet et celles d'un trafic de type Poissonnien

en plus volumineux, comme des données musicales ou vidéo par exemple, il est clair que l'échelle du phénomène de LRD augmente, ce qui explique pourquoi l'amplitude des oscillations mesurées dans l'Internet, même avec une granularité d'observation élevée, est si importante. En effet, les phénomènes de dépendance de TCP se propageant dans le trafic par l'intermédiaire des flux, l'allongement des flux observable avec l'arrivée des applications P2P, augmente aussi la portée des dépendances qui se retrouvent à très long terme. Une oscillation à la date t provoque ainsi d'autres oscillations à des dates pouvant être éloignées de t. Une congestion (sporadique) induite par une forte oscillation sur un flux peut ainsi ne pas être complètement résorbée plusieurs heures après (dans le cas du téléchargement d'un film par exemple), c'est à dire que ce flux qui continuera à proposer au réseau des pics de trafic dépendants de cette première oscillation, engendrera de nouvelles congestions sporadiques[2]. De plus, il est clair que les longs flux, à cause de leur longue existence dans le réseau, et par l'importance des capacités des réseaux la plupart du temps très surdimensionnés, ont le temps d'atteindre des valeurs élevées de la fenêtre de contrôle de congestion (CWND). Aussi, une perte va entraîner une forte baisse, suivie d'une forte hausse du débit du flux. Les nouveaux usages de l'Internet (P2P notamment) qui entraînent la transmission de fichiers de plus en plus gros favorisent l'émergence d'oscillations de très fortes amplitudes et dépendantes sur de très longues périodes[3].

[2]Naturellement, avec les nouvelles capacités hauts débits des réseaux et le choix des opérateurs de sur-dimensionner les capacités du réseau pour améliorer la qualité de service, les congestions sur de longues durées n'existent plus. Par contre, des congestions sporadiques continuent à exister dans le réseau, notamment dans le réseau d'accès, à cause des fortes oscillations du trafic.

[3]En effet, les résultats métrologiques sur la caractérisation du trafic depuis l'an 2000 ont montré que l'Internet qui était alors presque exclusivement utilisé pour de la navigation web, est aujourd'hui de plus en plus utilisé par des applications P2P pour des échanges de fichiers, souvent de tailles importantes (fichier audio, films, etc.). Aussi, le trafic Internet en 2000 se caractérisait par la transmission de flux courts, avec seulement un très faible pourcentage de flux de tailles plus

Bien sûr, les oscillations sont très dangereuses pour l'utilisation globale des ressources du réseau étant donné que la capacité consommée par un flux après une perte, par exemple, ne peut pas être immédiatement utilisée par les autres flux : ceci correspond à un gaspillage de ressources, et évidemment entraîne une diminution de la QdS globale du trafic et du réseau. En effet, plus l'amplitude des oscillations est importante, plus les performances globales dans le réseau sont faibles [18].

10.2 Estimation de la LRD du trafic

Il est apparu clairement dans la partie précédente que les éléphants et les oscillations importantes qu'ils induisent ont un fort impact sur les profils de trafic et sur les performances globales du réseau. La Figure 10.2 montre la mesure de la LRD du trafic représenté sur la Figure 28.1. Cette figure a été produite en utilisant l'outil LDestimate conçu par Abry et Veitch [4] [5] qui estime la LRD d'un trafic Internet à toutes les échelles. Le principe de cet outil repose sur une décomposition en ondelettes des séries temporelles du trafic, ce qui permet d'obtenir une représentation graphique des lois de dépendance à toutes les échelles temporelles. Les petites valeurs d'octaves représentent la dépendance à court terme, alors que les grandes valeurs d'octaves représentent la LRD. Sur la figure 10.2, un phénomène de "bi-scaling" apparaît (cf. le coude sur la Figure 10.2 autour de l'octave 8) qui montre une différence significative du niveau de dépendance entre les petites et grandes échelles du trafic. Pour les petites échelles temporelles, (octave < 8), qui montrent la dépendance entre des packets proches dans le temps (c'est-à-dire des paquets dont les dates d'émission sont voisines), la dépendance est très limitée. Une telle dépendance est celle qui existe entre des paquets d'une même fenêtre de congestion et qui sont donc transmis très près les uns des autres. En revanche, pour les grandes échelles de temps (octave > 8), la LRD peut être très élevée. Pour les octaves entre 8 et 12 qui correspondent par exemple à la dépendance qui existe entre des paquets transmis au cours de deux fenêtres de congestion adjacentes, la dépendance est plus forte que pour les octaves inférieures à 8. Ceci peut s'expliquer à cause de la structure de contrôle à boucle fermée des mécanismes de contrôle de congestion de TCP pour lesquels l'émission d'un paquet d'une fenêtre de contrôle de congestion dépend de la réception d'un acquittement associé à la réception d'un paquet de la fenêtre de congestion précédente Naturellement, ce phénomène existe pour des fenêtres de congestion consécutives, mais aussi pour toutes les fenêtre de congestion d'un même flux. Cela signifie donc que la présence dans le trafic de très longs flux introduit des phénomènes de dépendance à très long terme, comme cela a été observé sur le trafic (Figure 10.2 Pour les grandes octaves). La conséquence d'une telle LRD est un des problèmes majeurs pour le trafic car chaque oscillation au temps t peut se répéter à n'importe quel autre moment t', et les comportements du trafic à ces deux dates seront dépendantes (à cause de la dépendance entre paquets créée par un protocole comme TCP sur de longs flux).

---

importantes (environ 2 % des flux faisaient plus 100 Ko). Avec l'arrivée des applications P2P, la proportion de gros flux a considérablement augmenté, de même que la taille moyenne des flux transmis. De fait, le trafic Internet présente aujourd'hui une double caractéristique issue des applications dominantes générant ce trafic, avec de très nombreux petits flux (souris) et un nombre grandissant de gros flux (éléphants) [26].

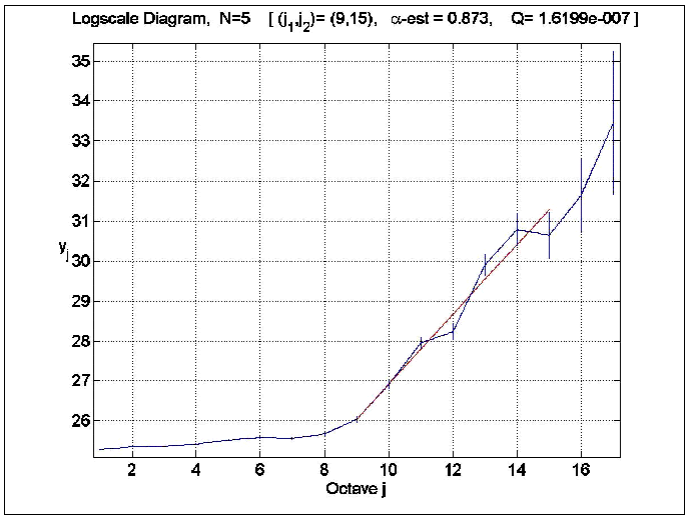


FIG. 10.2 – Evaluation de la LRD du trafic d'un lien d'accès

10.2.1 Une étude de cas montrant le lien entre oscillation et LRD dans le trafic Internet

Les deux parties précédentes viennent de mettre en évidence les phénomènes oscillatoires et de LRD du trafic Internet (au niveau d'un réseau d'accès à Renater, plus généralement appelés réseaux de collecte dans la terminologie Renater). Ces observations et analyses, associées à la littérature dans le domaine amènent à penser que la LRD est un bon moyen de caractériser les phénomènes oscillatoires du trafic, en particulier dans leur composante "portée". Toutefois, ce problème n'a, à notre connaissance, jamais été vraiment abordé dans la littérature existante. Aussi, cette expérience qui va être décrite dans la suite a pour objectif, sur un exemple, de montrer l'existence de ce lien entre les deux aspects oscillations et LRD. Pour ce faire, l'expérience menée s'est proposée de comparer au travers de simulations NS le trafic réel avec le même trafic re-simulé, mais pour lequel le protocole de transmission TCP a été remplacé par TFRC [4] [21] [14]. Comme nous allons le voir plus loin, l'objectif de TFRC par rapport à TCP est de fournir des sources de trafic beaucoup plus lisses et régulières, c'est-à-dire des sources qui ne présentent pas, ou peu d'oscillations. L'objectif est donc de montrer que lorsque l'on emploie TFRC, et donc quand on génère un trafic régulier et lisse, la LRD qui apparaît est très réduite par rapport au cas où TCP est utilisé.

10.2.2 Principes de TFRC

TFRC a pour objectif d'offrir aux applications des débits en émission lisses et réguliers avec des variations très douces ; dans tous les cas plus douces que celles de TCP. En utilisant un tel mécanisme de contrôle de congestion à la transmission des flux éléphants, c'est-à-dire à la part la plus importante du trafic, nous espérons pouvoir réduire sensiblement les oscillations du trafic. Le débit en émission de chaque source TFRC est calculé sur la base d'informations provenant du récepteur, notamment

[4] TFRC : TCP Friendly Rate Control

le taux de perte observé lors du dernier RTT.Ce débit est calculé une fois par RTT[5] et dépend du taux de perte mesuré par récepteur et du RTT mesuré par l'émetteur [13] [14] en utilisant l'équation 10.1 :

$$X = \frac{s}{R * \sqrt{2 * b * \frac{p}{3}} + (t_{RTO} * (3 * \sqrt{3 * b * \frac{p}{8}}) * p * (1 + 32 * p2)))} \quad (10.1)$$

where :

- X est le debit d'émission en octets/seconde,
- s est la taille des paquets en octets,
- R est le RTT en secondes,
- p est le taux de perte (entre 0 et 1.0), c'est à dire le nombre d'événements de pertes divisé par le nombre de paquets transmis,
- t_{RTO} est le timeout de retransmission de TCP en secondes,
- b est le nombre de paquets acquittés par un seul acquittement TCP.

Dans TFRC, un événement de perte est considéré si au moins une perte apparaît au cours de la période d'un RTT. Cela signifie que plusieurs pertes intervenant dans le même RTT font partie du même événement de perte. Ainsi, le modèle de dépendance des pertes de l'Internet est cassé car la plupart des pertes dépendantes apparaissent dans le même RTT (liées à une congestion sporadique). La récupération des pertes va ainsi être facilitée et beaucoup plus efficace qu'avec TCP qui n'est pas très efficace pour récupérer plusieurs pertes en séquence. Cette approche utilise en fait les résultats de [33] qui présente une analyse et un modèle pour le processus de perte des liens de l'Internet.

10.2.3 Description de l'expérience

Cette expérience a pour objectif de fournir une étude comparative des caractéristiques globales du trafic suivant que les éléphants sont transmis en utilisant TCP ou TFRC. Cette expérience vise aussi à fournir des résultats dans un environnement réaliste. Pour cela, elle se base sur des traces de trafic capturées par les équipements de métrologie passive DAG [23] [10]. Ainsi, les flux identifiés dans les traces de trafic originales sont rejouées dans NS-2 avec les mêmes dates de démarrage relatives et en respectant les autres caractéristiques des flux (taille des paquets, taille des flux, etc.). D'autre part, l'environnement de simulation a été construit de façon à ce que la mise en forme des paquets soit faite de façon cohérente avec ce qui s'est passé dans la réalité. Ainsi, les files d'attente et capacités des liens de l'environnement de simulation sont dimensionnées de façon à respecter les taux et modèles de pertes observés sur la trace réelle. De même, les délais de chacun des liens est fixé de façon à respecter les RTT mesurés pour les flux. Enfin, les sources de trafic sont positionnées de façon à respecter entre les flux re-simulés les contentions qui sont apparues dans la réalité. Pour plus d'informations quand à la méthode de re-simulation, le lecteur pourra se reporter à [20] [25]. Dans cet environnement de simulation et pour le compte de notre expérimentation, les flux éléphants sont donc transmis dans le simulateur en utilisant TFRC alors que les autres flux utilisent TCP New Reno[6]. Dans la suite, l'étude comparative va donc porter sur la trace originale d'une part et sur la trace simulée d'autre part dans laquelle les flux éléphants sont transmis en utilisant TFRC.

[5] RTT : Round Trip Time

[6] TCP New Reno a été choisi car c'est encore aujourd'hui la version du protocole TCP la plus utilisée dans l'Internet. Pour augmenter encore le réalisme des simulations, il serait intéressant de rejouer les flux courts avec la version de TCP qui a été utilisée dans la trace originale. Mais déterminer cette information à partir d'une trace passive est impossible pour la plupart des flux courts : seuls ceux qui subissent un grand nombre de pertes peuvent donner suffisamment d'informations pour déterminer la version de TCP qui a été utilisée. De toute manière, l'erreur reste minime car sur des souris les mécanismes

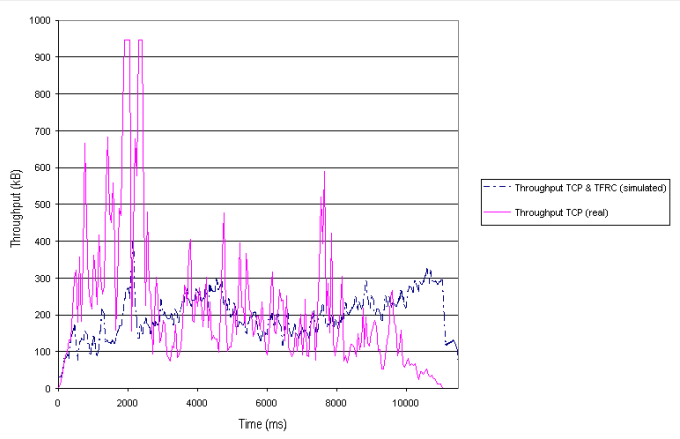


FIG. 10.3 – Evolution du debit au cours du temps

Par rapport au thème de cette étude comparative qui vise à étudier les effets de TFRC sur le caractère oscillant du trafic, les paramètres qui vont être évalués sont les paramètres traditionnels de débit, mais aussi des paramètres statistiques du trafic comme la LRD (comme cela a été justifié plus haut), et quelques paramètres mesurant le niveau de variabilité du trafic. Pour cela, nous utilisons un coefficient de stabilité (SC) qui est défini par le quotient :

$$\text{Coeeficient de Stabilité (SC)} = \frac{\text{trafic moyen échangé}}{\text{écart type du trafic échangé } (\sigma)} \tag{10.2}$$

10.2.4 Impact de TFRC sur les oscillations

La figure 10.3 présente le trafic dans les deux cas d'étude soit le cas reel et le cas simulé (avec TFRC). Visuellement, il apparaît clairement qu'en utilisant TFRC pour transmettre les éléphants à la place de TCP, le trafic global est bien plus lisse et régulier, et que tous les grands pics de trafic que l'on peut voir sur le trafic réel ont disparu du trafic simulé avec TFRC.

Les résultats quantitatifs sont présentés dans la table 10.1. Ils confirment que la variabilité du trafic dans le cas du trafic réel (utilisant TCP pour transmettre les éléphants) est bien plus importante par rapport au cas simulé dans lequel les éléphants sont générés avec le protocole TFRC (pour l'écart type σ nous avons calculé que σ(trafic réel) = 157.959 ko $\gg$ σ(trafic simulé) = 102.176 ko). De la même façon, le coefficient de stabilité est plus faible dans le cas réel (SC = 0.521) par rapport au cas simulé (SC = 0.761).

En ce qui concerne le débit global, nous avons mesuré des débits assez proches dans les deux cas (Débit(trafic réel) = 82.335 ko $\approxeq$ Débit(trafic simulé) = 77.707 ko). Ce résultat est excellent pour TFRC qui n'a pas les même capacités que TCP pour consommer rapidement une grande quantité de ressources [24], et même si TFRC est donc moins agressif que TCP, il est capable d'atteindre

modifiés d'une version à l'autre de TCP n'interviennent que très rarement. Les modifications successives de TCP ont surtout été faite pour augmenter la performance de TCP pour la transmission de flux longs.

Protocole	Débit moyen (kB)	σ du débit(kB)	SC
TCP New Reno (NR) : cas réel	82.335	157.959	0.521
TCP NR & TFRC : cas simulé	77.707	102.176	0.761

TAB. 10.1 – Caractérisation du débit pour les protocoles TCP et TFRC

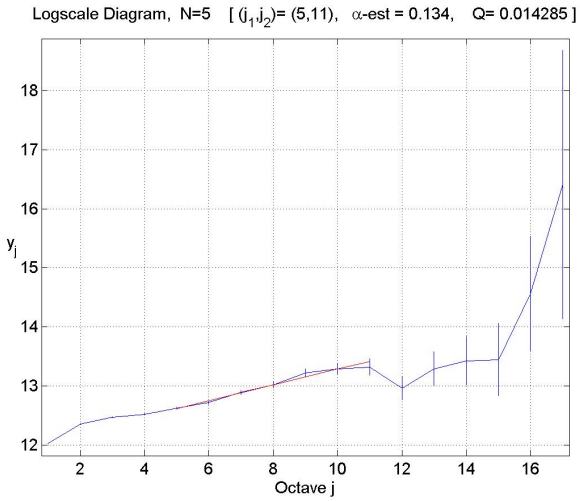


FIG. 10.4 – Evaluation de la LRD pour le trafic simulé incluant des éléphants TFRC

le même niveau de performance que TCP. Ceci confirme l'importance de la stabilité du trafic pour obtenir des performances de haut niveau et optimisées pour les réseaux de communication [18].

En ce qui concerne la LRD maintenant, la figure 10.4 montre que dans le cas simulé la propriété de bi-scaling est sensiblement réduite et la courbe, même pour les grandes octaves a une pente peu marquée. Cela signifie que toutes les formes de dépendance, et en particulier celles à long terme ont été réduites de façon drastique. Les valeurs pour la LRD qui s'exprime à l'aide du facteur de Hurst sont : H(trafic réel) = 0.641 et H(trafic simulé) = 0.194 (ce qui dans ce dernier cas est non significatif, mais marque bien l'absence de LRD). Un tel résultat confirme deux aspects de notre proposition :

- TFRC permet de lisser le trafic lié à chaque flux individuellement, ainsi que le trafic global sur un lien ;
- La LRD est le bon paramètre qui permet de qualifier et quantifier les lois d'échelles et de dépendance des oscillations.

10.3 Conclusion

Sans en donner une preuve formelle irréfutable, cette expérience a permis de mettre en évidence le lien étroit qui existe entre la caractéristique oscillante du trafic et la LRD. En effet, à partir du moment où on utilise pour transmettre l'essentiel de la charge du trafic (les éléphants) un protocole

qui ne crée pas d'oscillations (TFRC) et qui brise le modèle de dépendance des pertes à récupérer [7], la LRD disparaît quasiment du trafic.

Ce résultat d'analyse est important car il donne un outil pour caractériser qualitativement et quantitativement un des phénomènes caractéristiques du trafic Internet, qui est de plus un élément dégradant de la performance du réseau. Surtout il permet de donner des directions de recherche pour trouver des parades à ce phénomène, en particulier concernant les protocoles de transport et leurs mécanismes de contrôle de congestion.

[7] TFRC brise le modèle de dépendance entre pertes en regroupant dans le même événement celles qui sont dues au même événement de congestion. Or ces pertes dépendantes sont à la source des réponses oscillantes de TCP conditionnées par ses mécanismes de contrôle de congestion. Ainsi, les réponses à des changements de la charge du réseau (voire des congestions) de TFRC ne sont pas interdépendantes, faisant ainsi disparaître la LRD

Chapitre 11

Conclusion

Ce document a présenté les travaux qui ont été menés dans le sous-projet 3 de METROPOLIS intitulé “Analyse du réseau”. Nous avons successivement abordé les points suivants relatifs à l’analyse des traces passives DAG :

- mise en forme des traces,
- caractérisation simple du trafic,
- caractérisation avancée du trafic.

Dans chacune des parties précédentes nous avons détaillé les difficultés que nous avons rencontrées et présenté les solutions que nous envisagions. En particulier, concernant la problématique de la caractérisation avancée du trafic, nous avons présenté le logiciel ZOO qui implémente une méthode de décomposition du trafic qui permettra de simplifier la phase de modélisation qui sera abordée dans le sous-projet 5 intitulé “Modélisation”.

De plus, les premiers résultats de décomposition du trafic (obtenus grâce à ZOO) ont été détaillés dans ce document (cf. décomposition souris / éléphants) et seront approfondis dans les futurs travaux de modélisation. En particulier, le lien qui existe entre la LRD et les oscillations du trafic présenté dans la dernière section de ce document sera repris pour servir de base aux propositions de modèles que nous envisageons pour les différentes classes (souris, éléphants...) que nous avons pu isoler dans le trafic.

Bibliographie

[1] http://www.endace.com.

[2] Tcpdpriv. http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.

[3] Veres A. and Boda M. The chaotic nature of tcp congestion control. In *INFOCOM*, 2000.

[4] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44(1) :2–15, 1998.

[5] Patrice Abry, Darryl Veitch, and Patrick Flandrin. Long range dependence : Revisiting aggregation with wavelets. *Journal of Time Series Analysis*, 19(3) :253–266, 1998.

[6] Sikdar B. and Vastola K. On the contribution of tcp to the selfsimilarity of network traffic. In *International Workshop on Digital Communications : Evolutionary Trends of the Internet*, 2001.

[7] Jin Cao, William S. Cleveland, Dong Lin, and Don X. Sun. Internet traffic tends to poisson and independent as the load increases. Technical report, Bell Labs, 2001.

[8] J. Daemenand and V. Rijmen. Aesproposal : Rijndael. Technical report, Computer Security Resource Center, National Instituteof Standards and Technology.

[9] Trang Dinh Dang, Sandor Molnar, and Istvan Maricza. Capturing the complete multifractal characteristics of network traffic. *GLOBECOM*, 2002.

[10] S. Donnelly. Passive calibration of an active measurement system. to be presented at PAM2001.

[11] A. Downey. Evidence for long-tailed distributions in the internet, 2001.

[12] R. Casellas et al. Sous-projet 1 : 4 : méthodologie pour la mesure et l'échantillonage. Technical report, Projet Metropolis - Contrat RNRT METROPOLIS.

[13] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4) :458–472, 1999.

[14] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.

[15] Liang Guo, Mark Crovella, and Ibrahim Matta. How does TCP generate pseudo-self-similarity ? Technical Report 2001-014, 12 2001.

[16] Papagiannaki K., Taft N., Bhattacharyya S., Thiran P., Salamatian K., and Diot C. A pragmatic definition of elephant in internet backbone traffic. 2001.

[17] Park K., Kim G., and Crovella M. On the relationship between file sizes, transport protocols, and self similar network traffic. In *IEEE ICNP*, 1996.

[18] Park K., Kim G., and Crovella M. On the effect of traffic self-similarity on network performance. In *SPIE International Conference on Performance and Control of Network Systems*, 1997.

[19] Park K. and Willinger W. Self-similar network traffic : an overview. In *Self-similar network traffic and performance evaluation*, 2000.

[20] N. Larrieu and P. Owezarski. De la métrologie pour l'ingénierie des réseaux de l'internet. *Techniques et Sciences Informatiques numéro thématique Réseaux et Protocoles*, 2003.

[21] N. Larrieu and P. Owezarski. Tfrc contribution to internet qos improvement. In *QoFIS*, 2003.

[22] K. Claffy N. Brownlee. Understanding internet traffic streams : Dragonflies and tortoises. In *IEEE Communications*, 2002.

[23] P. Owezarski and N. Larrieu. Sous-projet 7 : infrastructure de mesure. Technical report, Projet Metropolis - Contrat RNRT METROPOLIS.

[24] P. Owezarski and N. Larrieu. Coherent charging of differentiated services in the internet depending on congestion control aggressiveness. *Computer Communications, special issue on "Internet Pricing and Charging : Algorithms, Technology and Applications"*, 2003.

[25] P. Owezarski and N. Larrieu. Trace based methodology for realistic simulations. In *ICC'04*, 2004.

[26] Owezarski P., Andreu D., Fricker C., Salamatian K., Chekroun C., Benameur N., Olivier P., Roberts J., Robert P., and Guillemin F. Sous-projet 1 : Rapport état de l'art. Technical report, Projet Metropolis - Contrat RNRT METROPOLIS.

[27] GTK Web Site. `http://www.gtk.org`.

[28] A. Veres and M. Boda. On the impact of short files and random losses on chaotic tcp systems. In *IFIP ATM & IP 2000 Workshop*, 2000.

[29] Willinger W., Paxson V., and Taqqu M. Self-similarity and heavy tails : Structural modeling of network traffic. In *Practical Guide To Heavy Tails : Statistical Techniques and Applications*, 1998.

[30] Site web du logiciel TCPDUMP. `http://www.tcpdump.org`.

[31] Site web QoSMOS. `http://www.qosmos.net`.

[32] Jun Xu, Jinliang Fan, Mostafa Ammar, and Sue B. Moon. On the design and performance of prefix-preserving ip traffic trace anonymization. In *Proc. of 10th IEEE International Conference on Network Protocols (ICNP 2002)*, 2002.

[33] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *ACM SIGCOMM Internet Measurement Workshop (IMW'2001)*, 2001.

Chapitre 12

Analyse de trace par le logiciel T-RAT

12.1 introduction

The cause that limits the rate of a TCP connection can be located at the sender side, the receiver side or within the network itself. The main objective of this work is to find these causes. We perform the analysis as follows : a connection is split up into flows, in order to receive a consistent unit referring to the network traffic. Our TCP rate analysis tool (TRAT) then attempts to determine the factor that limits each flow ?s transmission rate respectively and classify the cause. Although network measurements are an active research area currently, there has still not yet been done much of work on the actual behavior of TCP in the Internet. One of the first known attempts was made by a group at the AT&T Labs in 2002 [7], whose results are an important base for the development of the tool. The remaining of this part is structured as follows : the next section provides the terminology used when describing the tool. We next describe the design of the TRAT. We then present validation results of the tool, followed by results obtained from the analysis of traces collected at Eurecom and the LAAS. In the last section, we will conclude with future directions.

12.2 Terminology

The following definitions ensure a consistent terminology throughout the rest of this part : For the measurements there is no traffic injected, that means traces are obtained from passive measurements. A trace is a snapshot of the Internet traffic, which is taken at a specific point. It consists of all packets passing this point. A packet is an atomic unit of data, whose size depends on the protocol to which the packet belongs. A trace contains apart from TCP packets all usual packets, which belong to the common protocols used in the Internet, e.g. UDP, ARP, ICMP. Investigations of the network traffic show, that most traffic is caused by TCP packets (about 90%), followed by UDP (about 10%) and a negligible amount shared by other protocols [6]. This justifies our approach where we concentrate on the TCP traffic. Since within a trace lots of different connections are mixed together, the first

step for the analysis is to separate transferred packets from each other in order to bundle them to its original connections. A connection covers all packets of a trace with the identical sender and receiver, whereas both sender (source of data stream) and receiver (destination) are defined by an IP address and a port number. The combination of the IP address and a port number is called a socket. The Round trip time (RTT) is the time interval between the point of time, when a packet is sent and the receipt of the according acknowledgement. The RTT varies with time and therefore the average RTT is used to describe a given connection. Typical values of the RTT range from a few milliseconds up to 1200ms or even more.

RTT in ms Description / Examples <100 In general, an RTT below 100ms is very good, indicating that 1) the corresponding sender and receiver are electronically "close" to each other, 2) that sender and receiver are well connected to the net and 3) the receiver is responding well. 100-200 RTT values of 100-200ms are typical of ISDN and DSL connections. 200-300 RTT values of 200-300ms are typical of dialup connections. >400 RTT values well above 400ms on a connection may indicate a very distant site, a very heavily loaded site, or a problem. 800-1200 RTT values in the 800-1200ms range are typical of a satellite connection. >1200 Wireless lines

RTT in ms	Description / Examples
<100	In general, an RTT below 100ms is very good, indicating that 1) the corresponding sender and receiver are electronically "close" to each other, 2) that sender and receiver are well connected to the net and 3) the receiver is responding well.
100-200	RTT values of 100-200ms are typical of ISDN and DSL connections.
200-300	RTT values of 200-300ms are typical of dialup connections.
>400	RTT values well above 400ms on a connection may indicate a very distant site, a very heavily loaded site, or a problem.
800-1200	RTT values in the 800-1200ms range are typical of a satellite connection.
>1200	Wireless lines

FIG. 12.1 – Typical RTT values [4]

Since the conditions of the state of the network during a long connection may change and consequently the cause of the rate limitation may also change, each connection is split up into flows for the rate limitation decision itself. A flow is defined by a maximum of 256 data packets or when no packets are seen for 15 seconds. A flow always terminates at the end of a flight (see definition below). Therefore the number of 256 data packets is not strict and depends on the number of packets within the flight, in which the amount of 256 data packets is reached. For each flow within a connection, a rate limitation inference is made. Due to the characteristics of TCP, packets are sent in bursts. The resulting bundle of packets is called a flight. A flight contains the packets, which are sent during the interval of a round trip time. A flight starts always with a data packet. Its flight size is defined by the difference of the largest sequence number within this flight and the largest sequence number seen before this flight, divided by the maximum segment size of the corresponding connection. The flight size is therefore a value, which correlates with the amount of data within the flight.

Figure 12.2 visualizes some definitions made so far. Each packet represents a TCP packet, whereas the different colors stand for different connections. Time increases from left to right. The dotted

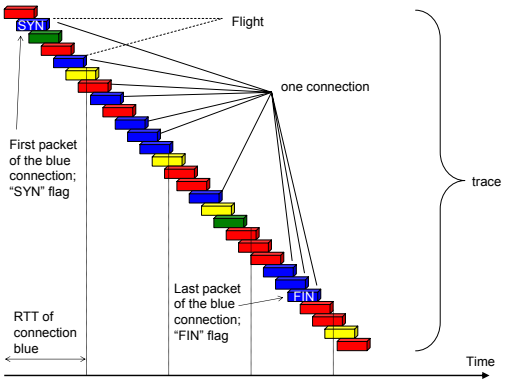


FIG. 12.2 – Schematic illustration of basic definitions

vertical lines indicate the interval of the round trip time of the blue connection. All blue packets within such an interval compose a flight. In the example above there is only one packet within the third interval. This might indicates a loss or the application on top of TCP not sending data any more. It is more difficult to visualize a flow, as its definition indicates a set of about 256 packets, which obviously cannot be represented in the scheme above. In this example the whole blue connection corresponds to one flow, since the connection consists of only ten packets. Note that this is not unusual in reality because most TCP connections are very short. A loss is a packet, which does not arrive at the receiver side. The most common reason is congestion. Losses due to damage are negligible low, as their amount is usually much less than one per cent [3]. The two main states of a TCP connection are the slow start (SLS) phase and the congestion avoidance (CA) phase. The rate analysis tool described in this paper categorizes the TCP flights to these two states. This categorization is necessary as an intermediate result in order to find the correct grouping of flights. Apart from these two states, fast retransmission and fast recovery phases can be entered as a response to a loss. The realization of the latter two states depends on the implementation of the TCP version. A description of all possible states can be found in [2].

12.3 Characterization of rate limiting factors

The rate limitation is categorized to eight different cases :

1. Bandwidth limited : The sender is limited by the bandwidth of the bottleneck link. It fully utilizes the bandwidth without competing with any other flows on the bottleneck link
2. Congestion limited : The sender ?s window is adjusted according to TCP ?s congestion control algorithm in response to losses.
3. Receiver window limited : The sending rate is limited by the receiver ?s maximum advertised window.
4. Sender window limited : The sender rate is constrained by the buffer space at the sender, which limits the amount of unacknowledged data that can be outstanding at any time.

5. Opportunity limited : The application has a limited amount of data to send and never leaves slow start.
6. Application limited : The application does not produce data fast enough to be limited by either the transport layer or by the network
7. Transport limited : The sender reaches the congestion avoidance state and then does not experience any more losses.
8. Unknown limited : None of the above seven causes can be determined.

Both bandwidth and congestion limitation are caused within the network itself. Due to that losses can be observed. The other limitations are due to characteristics of the sender, receiver or the application itself. Bandwidth limitation has a physical cause within the network, as the connection is not disturbed by other network traffic and the resources neither of the sender nor of the receiver are fully exhausted. The illustration below visualizes the problem : Packets of one connection arrive at a point within the network, from which on the bandwidth is too small in order to ship all the packets.

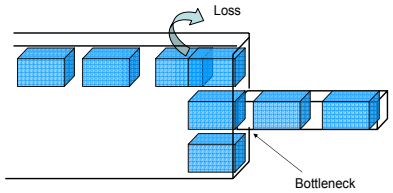


FIG. 12.3 – Illustration of bandwidth limitation

The consequence is a loss, when the buffer at the bottleneck runs out of space. A typical example is a modem line, but we could think also at an ADSL link where lots of users are simultaneously active. The ADSL concentrator then forms a bottleneck and the resulting TCP flows can be classified as bandwidth limited.

Compared to bandwidth limitation, congestion limitation is of a more dynamic nature. It depends on the current network traffic.

If there is lots of traffic, there are consequently many competitors for the network resources and congestion is more likely even if the physical bandwidth is quite large. Figures 12.4 and 12.5 show two snapshots of a network at a specific point. The location is near a point, where the physical conditions of the network change significantly, for example a router. In figure 12.4 there is no congestion, all the packets of the two competitive connections can be shipped. However, in figure 12.5 there is a change of the conditions. Four different connections compete with each other and overload the buffer at the critical point of the network. The consequence is a loss. Please note that the number of connections of the example above are not realistic and are chosen only to make it possible to visualize the problem. Receiver and sender window limitations are due to too small buffer space located at

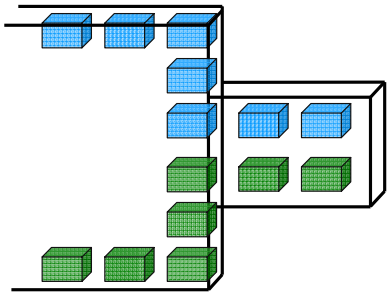


FIG. 12.4 – Illustration of congestion limitation (1) - Network situation without congestion

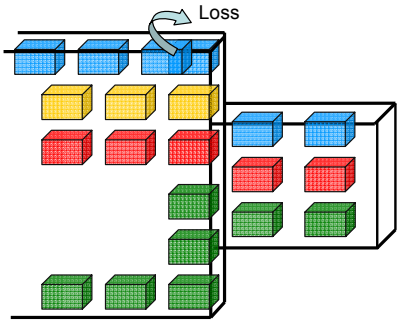


FIG. 12.5 – Illustration of congestion limitation (2) - Network situation with competitive connections causing congestion and resulting in a loss

the receiver and then sender side respectively. These buffers are the buffers located between the TCP and the application layers. Both opportunity and application limited connections are caused by the application, which runs over a TCP connection. Opportunity limited connections have a very limited amount of data to send. They correspond to the so-called mice in the network measurements literature. Therefore the resources of the network cannot be exploited sufficiently. An example scenario is web browsing : A user calls a web page containing a small amount of data from a server and does not cause any further action, because he reads the transferred information. In opposition to that, an application limited connection is caused by a data transfer, which is too slow. This is the case for instance for an interactive application like manually typed commands during a telnet or ssh session. The last rate limitation cause is transport limitation. Transport limitation is defined by a connection, which reaches the congestion avoidance phase and then does not experience losses any more. This category consists of sessions that leave the slow-start state since they experience a loss but are still quite small in terms and duration (this is why they don’t experience losses any more). This cate-

gory has been introduced because it has been observed on traces that most of the small connections experiences 0 or 1 loss at most. The opportunity and transport categories have been introduced to distinguish between these two categories.

12.4 TCP rate analysis tool

12.4.1 Outline of the algorithm

The input for the tool is a tcpdump file in text format representing a trace of the network traffic. Only TCP packets of a trace are taken into consideration. The trace is split up into connections, whereas for each connection a dynamic array is used. The array consists of TcpPacket objects, which are defined in the appropriate TcpPacket class. Each array (respectively connection) is identified by a key and is pushed on a heap. The end of a connection is reached if a packet with a FIN or RST flag is found. Once a connection is deemed to be finished, the round trip time is estimated. For this purpose 27 candidates between 3ms and 3s are tested, in order to estimate the best RTT. Since typical RTT are expected to be less than a second, 23 candidates cover this range and only four other candidates have values of more than a second, in order to handle outliers as well. As the burst of packets, which are governed by the TCP protocol, depends on the RTT, a connection is divided into flights according to one of the RTT candidates. The algorithm of grouping packets into flights will be described later in more detail. As a preparation for the RTT computation, an investigation follows to determine, whether a flight contains a loss. If a loss is detected within the interval of a flight, that means a packet of the flight has been lost during the transmission, the flight is marked accordingly. We use two techniques to find out losses : detection of three consecutive duplicate acknowledgments or detection of a packet in the wrong flight (with respect to its sequence number). The next step is the categorization of flights into the three different states ?slow start ?, ?congestion avoidance ? and ?unknown ? (describes flights, which cannot be assigned to the first two categories). The candidate with the best fitting categorization, i.e. the most data packets detected in either slow start or congestion avoidance state, delivers the first intermediate result for the best estimated round trip time. Figure 12.6 summarizes the description, which has been made so far.

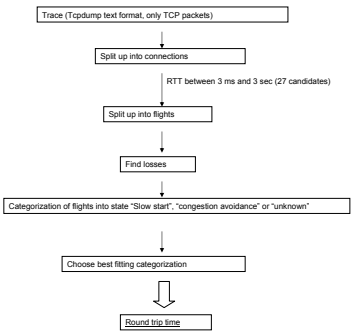


FIG. 12.6 – Outline of the algorithm (1)

Hereafter is a description of the possible flight states. Groups of four flights are analyzed in order to make a positive decision either for slow start or congestion avoidance state. As mentioned above three categories for flight states exist : slow start, congestion avoidance and unknown. The first flight of a group of four flights is supposed to be in slow start state, if four successive flights are consistent with multiplicative increase, i.e. the flight sizes grow by a factor of about two. If a group of four successive flights show additive increase behavior, then the first flight is supposed to be in congestion avoidance state. In the latter case all following flights until the next group of four flights, which contain a loss, are also assumed to be in congestion avoidance state. If there is a loss within a group of four successive flights, the flights are in the state unknown. Flights, which are neither in slow start nor congestion avoidance state, are marked unknown. Having found the best estimated round trip time for the connection, the latter is split up into flows. For each flow a rate limitation test is made, which leads to the final results.

12.4.2 Design and implementation

The implementation is based on the recent version of JavaTM 2 Platform, Standard Edition, v 1.4.1. Although Java may be less time efficient as alternative languages like C++, it has the advantage of easy portability to different computing systems. Since the application is not time critical, the latter advantage was deciding. The italicized words in the following paragraphs refer to specific parts or expressions within the source code of the TCP rate analysis tool.

Class diagram The package fr.eurecom.tcpAnalyser contains 8 classes. The TcpAnalyse class includes the main method and calls the parser, which is implemented in the class Parser, and the analysis of the trace itself. The class TcpPacket defines a TCP packet as an object. Each line of a Tcpdump file represents one TCP packet, for which an object of the TcpPacket class is created and stored in the hash table allPackets, which contains for each connection of the trace a dynamic array. The further components are defined in the classes Connection, Flight and the class RateLimitAnalysis. For the graphical output of the results the two classes Output and TcpAnalyseChart are created. In the class Output the data set is defined, which is needed by the Chart2d library. The class TcpAnalyseChart is the interface to the package net.sourceforge.chart2d.

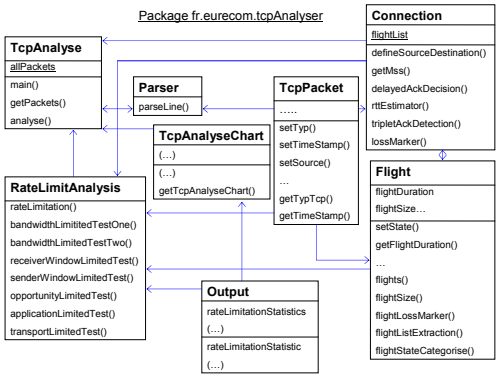


FIG. 12.7 – class diagram of the package fr.eurecom.tcpAnalyser

Figure 12.7 provides an overview of the design of the tool. For the sake of straightforwardness attributes and methods of the classes are not complete. The following subparagraphs describe each class of the package fr.eurecom.tcpAnalyser in more detail, containing a class specific diagram with all attributes and methods. For the sake of simplicity, we won't describe in the next sections, all the classes (please refer to [5] for a complete description) but only the most important ones from the algorithmic point of view : the flight class and the ratelimitanalysis class

Flight class The class Flight consists of 11 member variables and 18 methods. Each member variable can be requested by a ?get ? method. The method flightSize creates objects by the constructor Flight, which is described by the following 10 variables : The variable flightDuration describes the duration of the flight, defined by the difference of the timestamp of the last packet and the first packet of the flight. The timestamps of the latter two packets are stored in the variables startTT and endTT respectively. The variable flightSize is defined by :

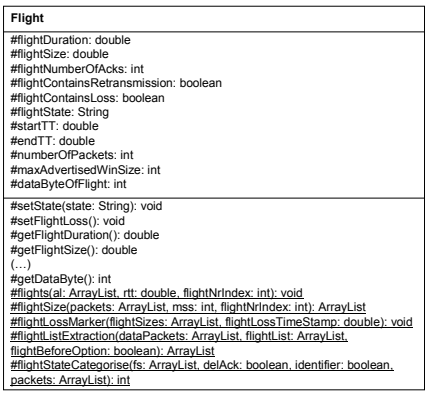


FIG. 12.8 – Diagram of the class Flight

The variable flightNumberOfAcks contains the number of acknowledgments of the flight. If a flight contains a packet, of which the upper sequence number is less or equal to the largest sequence number before the flight, the Boolean variable flightContainsRetransmission is set to true. The Boolean variable flightContainsLoss is set to true, if a packet within the flight is a loss indicator. The variable numberOfPackets contains the number of data packets, which are in the flight. The receiver ?s maximal window size, which is advertised during the flight, is stored in the variable maxAdvertisedWinSize. In the variable dataByteOfFlight is the number of data bytes stored, which have been transferred by this flight. The method flights groups packets into flights. The first packet of a flight has to be a data packet and defines the start T0 ; the end depends on the RTT and its variation. To cover the variation, the inter arrival intervals of packets between T0 + RTT and T0 + 1.7 RTT are investigated. Apart from these intervals, the inter arrival time of the first packet after T0 + 1.7 RTT is taken into consideration, however only with a half of the weight. The packet with the largest inter arrival time is the first packet of the next flight.

The method flightStateCategorise sets the value of the variable flightState. The method tests, whether a flight is in congestion avoidance or slow start state. Its design is based on the description of [7]. Both for the congestion avoidance and slow start states tests, we distinguish between connections with and without delayed acknowledgments. In order to make the decision more robust, a flight is

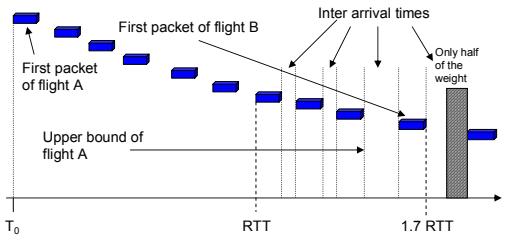


FIG. 12.9 – Illustration of a flight

supposed to be in SLS or CA state, if the flight itself and the three following flights, i.e. a group of four consecutive flights, pass the according test.

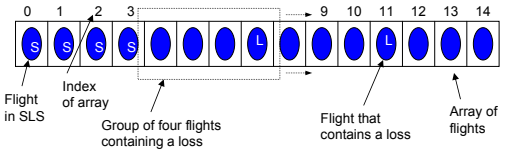


FIG. 12.10 – Execution of array of flights by the method flightStateCategoriser

Figure 12.10 visualizes the way of execution of the array of flights by the method flightState-Categoriser. The dotted rectangle marks the group of four flights, which is currently analyzed. The flights at indices 0 to 3 have already been analyzed and have been determined to be in slow start state (indicated with an S). When a flight is analyzed, the ?window ? of four flights slides one index further. In the example of figure 12.10 the flight at index 4 is currently analyzed. As the group of four flights contains a loss, it will be set to state unknown. If a flight fails both SLS and CA tests, the flight remains in state unknown. Figure 12.11 shows a visual representation of the flight state categorization. Input is an array of flights. During each cycle a group of four consecutive flights of the array is investigated. If there are less than four flights in the array, the automata transits from each state to the final state.

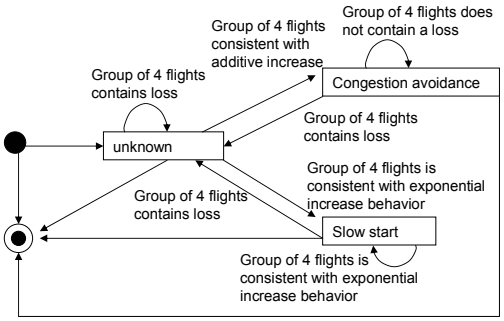


FIG. 12.11 – State chart diagram of flight states

The method flightLossMarker sets the Boolean variable FlightContainsLoss of a flight true by calling the method setFlightLoss. The method flightListExtraction is a help method in order to select a set of flights from an array of flights, whereas the bounds are defined by timestamps. This method is used in order to select the flights, which correspond to a specific flow.

RateLimitAnalysis In the class RateLimitAnalysis the rate limitation itself is done. The method rateLimitation splits a connection up into flows. After 256 data packets or a time out of 15 seconds the upper boundary of the current flight determines the end of a flow.

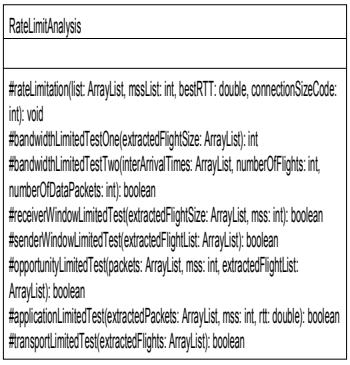


FIG. 12.12 – Diagram of the class RateLimitAnalysis

For each flow the ?LimitedTest ? methods are called, in order to make the rate limitation decision. The results are stored in the class Output, which is necessary for the graphical output. Apart from that the results are written to the two DataOutputStreams out and outMatlab, which create the textual output. The rate limitation tests are realized by seven methods. The method bandwidthLimitedTestOne determines a flow to be bandwidth limited, if at least three flights contain a retransmission

and the maximum and minimum flight sizes before the loss occurs differ by no more than 1, which means the amount of data does not differ by more than the MSS. If this test fails, the method tests for congestion limitation by looking for a flight containing a loss. The method bandwithLimited-TestTwo classifies a flow as bandwidth limited if the packets of the flow are nearly equally spaced. This test bases on the assumption that the inter arrival times of packets depends on the equally behavior of the bottleneck buffer rather than the causes of the protocol or the application. The method receiverWindowLimitedTest determines a flow to be receiver window limited if three consecutive flights contain more data bytes than the largest receiver advertised window size minus twice the MSS. If a flow is not receiver window limited, congestion limited or bandwidth limited, the method senderWindowLimitedTest determines the flow to be sender window limited, if the following two conditions are fulfilled : The percentile and the median are taken with regard to the flight sizes : 1. 80th percentile < median + 3 2. 80th percentile - 2 < four consecutive flight sizes < 80th percentile + 1 A flow is opportunity limited, if all flights of the flow are in state slow start and the total number of bytes transferred is less than 13 times the MSS, whereas 13 is a heuristic. This is implemented by the method opportunityLimitedTest. The method applicationLimitedTest determines a flow to be application limited, if a packet smaller than the MSS was transmitted followed by a lull greater than the RTT, followed by additional data. If a flow has entered congestion avoidance state, does not experience any more losses and at least every second flight shows an increasing flight size (to account for delayed ack), the flow is deemed to be transport limited by the method transportLimitedTest.

12.5 Validation

12.5.1 Simulation environment

For simulation purposes the sock program by Stevens [**?**] has been used in order to generate TCP data. It sends data continuously and allows specifying some parameters, such as sender and receiver buffers. Two machines have been connected through a third one, which acts as a router and run Nist Net [1] in a configuration visualized by figure 12.13.

FIG. 12.13 – Simulation environment

For all simulation files the router delays the packets the desired amount of milliseconds (half the delay in each direction), in order to simulate a specific RTT. For receiver and sender limited simulation the sock program sends a given amount of data from machine 1 to machine 2. At the same time Tcpdump collects the packets at the receiver side. For bandwidth limited simulation the same configuration is chosen. However, as an extension the bandwidth limitation at the router is also activated in order to choose the target bandwidth. In that case the bandwidth at the router is chosen lower than the maximum advertised window size divided by the RTT. In order to simulate the opportunity limited case, the sock program is specified to stop sending after n packets. The application limitation simulation has been made by switching the sock program to the mode ?Read standard input ?. Consequently the data transfer depends on the user, who generates data instead of the continuous data stream in the cases described above. For the transport limited simulation the number of transferred packets has been limited and a low loss probability is chosen. The trace has been collected several times until a single loss in the trace has been observed. For the congestion limited simulation two independent sock processes send at the same time data. The bandwidth limitation is lower than both connections ? sum of the quotient of maximum advertised window size divided by the RTT. In order to produce some losses, a finite queue size at the Nist Net router is specified. A simulation file has been created with a RTT of 20ms for each rate limiting cause as described above.

12.5.2 Validation results

The prototype of the tool decides correctly the rate limitations of bandwidth, congestion, receiver window, sender window and opportunity. The test for transport limitation behaves poorly.

			Causes for limitation →							
filename	estimated RTT (ms)	Number of flows	bandwidth	congestion	receiver window	sender window	opportunity	application	transport	unknown
l_sender	20.71	83	0	8	0	74	0	0	0	1
l_bw	6.16	85	84	0	0	1	0	0	0	0
l_receiver	20.47	85	0	13	85	0	0	0	0	0
l_oportunity	38.93	1	0	0	0	1	1	0	0	0
l_application	5629	1	0	0	0	0	0	1	0	0
l_congestion	25.80	83	5	40	0	24	0	0	0	15
	25.23	81	5	45	0	19	0	0	0	12
l_transport	16.62	5	0	1	0	2	0	0	0	2

FIG. 12.14 – Validation results for simulation files of the first prototype

The first column contains the filename, which suggests the rate limiting cause. The following two columns give information about the RTT and the number of flows, into which the connection has been divided. The example trace l_congestion is composed of two connections ; the other examples consist of a single connection. The last eight columns inform about the decision of the tool, how many flows have been identified and categorized to the appropriate limitation cause.

12.6 Results

12.6.1 Experimental environment

We present here results obtained from the analysis of traces collected at the Eurecom and LAAS networks. Since these two networks are campus networks, most of the traffic volume flows from outside to inside these networks. This setting represents a worst case scenario concerning the disturbance of the flow since almost all packets have passed through a long network channel before arriving at the measurement point.

12.6.2 Trace analysis

The Eurecom network has a 10Mb/s access to the Internet and a maximum observed load (for the incoming traffic) around 50%. We next present results for a trace of about one hour collected with an EtherReal probe. The LAAS network is about one order of magnitude larger than the Eurecom one (in terms of access capacity and number of hosts). The trace for which we present results has duration of about 4 hours. It has been collected with the DAG equipment on the LAAS network. Since the TRAT tool breaks connections into flows based on the inter-spacing of packets (a somewhat classical approach) but also on based on a maximum of around 256 packets per flow, we first present in tables 12.1 and i12.2 the distribution of the number of flows per connections. As expected, due to the highly varying nature of the Internet traffic, most of the connections consist of only a single flow. The tail is more pronounced for the LAAS trace than for the Eurecom trace.

Number of flows	1	2	>2
%	96	2	5

TAB. 12.1 – Number of flows per connections for the Eurecom trace

Number of flows	1	2	3	4	5	6	7	>8
%	81	7	3.5	1.6	0.8	0.6	0.4	5.3

TAB. 12.2 – Number of flows per connections for the LAAS trace

We next present the results in terms of rate limitation causes obtained for the two traces. We provide each time two views : a view by flows and view by bytes. These two views widely differ due to the high variability of the Internet traffic where a lot of small flows only represent a moderate fraction of the total load while a few number of very large flows generally carry most of the bytes. In figure 12.15 we present the distributions of causes in flows for the Eurecom trace while in figure 12.16 we plot the distributions of causes per bytes. Similar plots for the LAAS trace are provided in figures 12.17 and **??** Note that for all these figures, the percentages always sum to more than 100% since a flow (or a byte) night pass many tests since not all of them are orthogonal (for instance, except bandwidth limited flows, all flows might be deemed congestion limited). Overall, we observe that the opportunity limited flows always represent a large fraction of the flows and a small amount of the bytes. This is to be expected since opportunity limited flows broadly corresponds to mice. Also, the

main observed causes are : sender or receiver window limited, congestion limited and application limited. The latter always represent a significant fraction of the flows or bytes. This might be due to the test that simply seeks an interval between two packets (the first packet having a size less than the MSS) larger than RTT. Based upon this single event, the flow is deemed application limited. We intend, as future work, to further refine this definition.

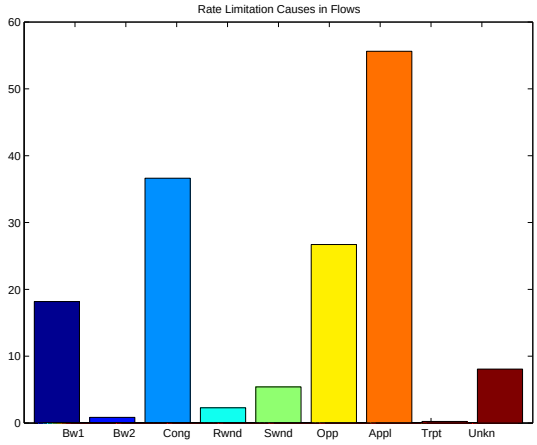


FIG. 12.15 – Causes in flow (Eurecom)

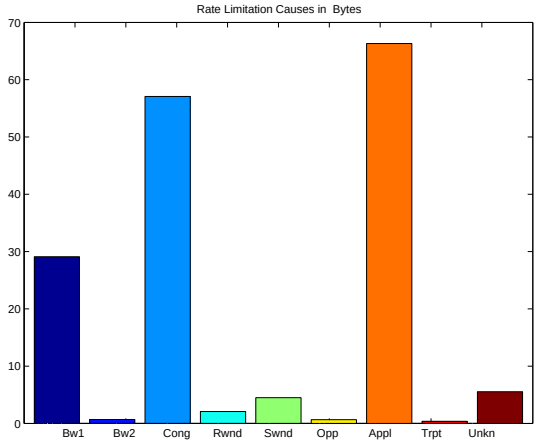


FIG. 12.16 – Causes in bytes (Eurecom)

Another interesting metrics provided by our tool is the average duration and throughputs achieved by the flows of the different categories. We plot the results for Eurecom in figure 12.19 and the results for LAAS in figure **??** The discrepancies between the two networks are more pronounced here. We can note for instance the important throughputs and durations in the Eurecom network of the bandwidth limited flows. Since bandwidth limitation should be caused, at least intuitively, by the access bandwidth of servers (remember most of the traffic comes from servers outside the networks),

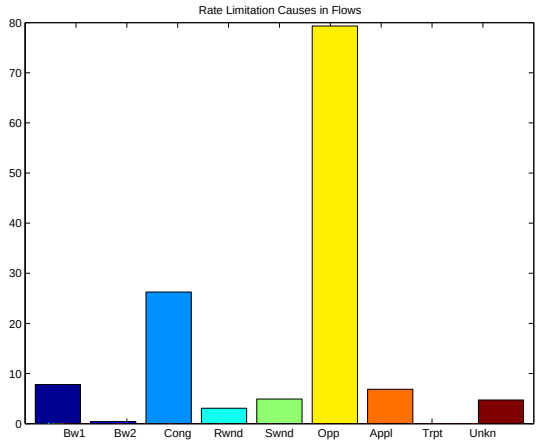


FIG. 12.17 – Causes in flows (LAAS)

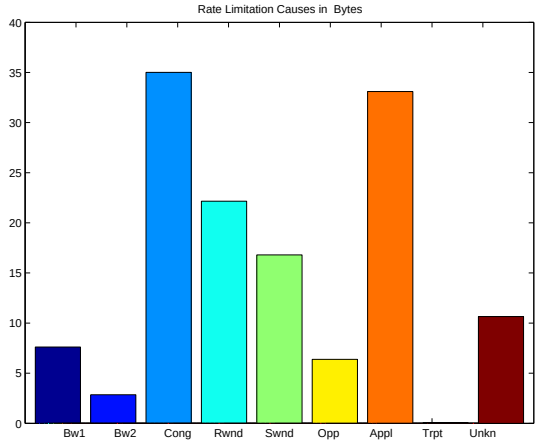


FIG. 12.18 – Causes in bytes (LAAS)

we should not observe two many of these flows. Our current hypothesis concerning this outlier is the use at Eurecom of a shaper that shapes the traffic to 10 Mb/s when we send traffic outside the regional area (otherwise we can burst at 100Mb/s). Additionally, the shaper aims at "optimizing" TCP by spacing ack packets. Such a behavior could cause the bandwidth limited test to be positive since it is partly based on the detection of even inter-spacing times of packets. We are currently investigating these issues.

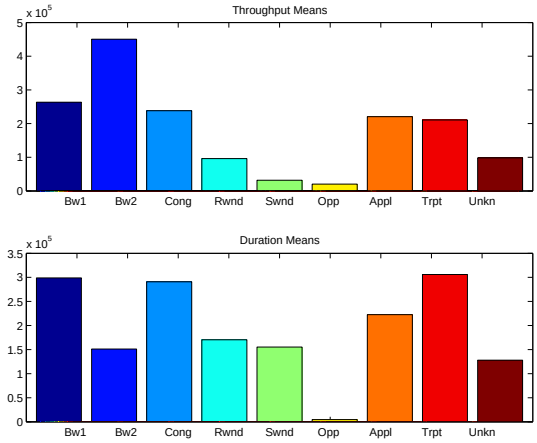


FIG. 12.19 – and Duration of flows for the Eurecom trace

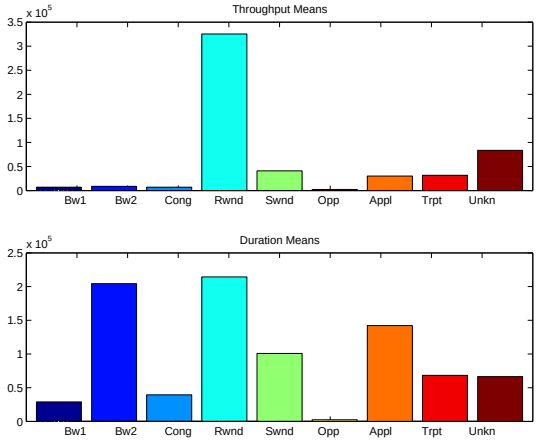


FIG. 12.20 – Throughput and Duration of flows for the LAAS trace

12.7 Outlook and future work

Both transport and application limitation tests have to be further investigated. In order to improve the application limited test, a different approach to calculate the correct RTT may lead to better results. This calculation has to be robust even for traces with a small amount of data transferred. Also, the rate limitation algorithms may be modified in order to make the rate limitation tests orthogonal to each other. It is also of interest to make a decision about the rate limitation of a connection. Currently, the tool determines the rate limitation for flows rather than connections. Although for long connections the rate limitation cause can change, which means it is impossible to make a distinct decision about the rate limitation, further investigation has to be done on this issue.

Bibliographie

[1] Michael J. Donahoo. Nist net. http://cs.ecs.baylor.edu/~donahoo/tools/nistnet/.

[2] IETF. *RFC 2001*. http://www.faqs.org/rfcs/rfc2001.html.

[3] Van Jacobson and Michael J. Karels. Congestion avoidance and control. In *SIGCOMM'98*, 1998.

[4] pcpitstop. *PC Pitstop*. http://www.pcpitstop.com/internet/pinger.asp.

[5] Christian Schleippmann. Desing and inplementation of a tcp rate analysis tool. Master's thesis, TU Muenchen/Eurecom, Aout 2003.

[6] Kevin Thompson, Gregory J. Miller, and Rick Wilder. *Wide-Area Internet Traffic Patterns and Characteristics*. MCI. http://www.vbns.net/presentations/papers/MCItraffic.pdf.

[7] Y. Zhang, L. Breslau, V. Paxson, , and S. Shenker. On the characteristics and origins of internet flow rates. In *ACM SIGCOMM 2002*, Aout 2002.

Chapitre 13

Analyse des flots pair a pair de type BitTorrent

13.1 Introduction

BitTorrent [3] is a file distribution protocol based on the peer-to-peer (P2P) paradigm. BitTorrent has quickly emerged as a viable and popular alternative to file mirroring for the distribution of large content, as testified by the numerous Web sites that host active "torrents" (e.g., http://f.scarywater.net/).

We have conducted a comprehensive analysis of BitTorrent to assess its performance. To that end, we have used two sources of information. First, we have obtained the server-side "tracker" log of arguably the most popular torrent so far—the 1.77GB Linux Redhat 9 distribution—for its first 5 months of activity. The log contains statistics for more than $180,000$ clients, and most interestingly, it clearly exhibits an initial flash-crowd period with more than $50,000$ clients initiating a download in the first five days. This first source of information allows us to estimate the global efficiency of BitTorrent, the macroscopic behavior of clients, and the scalability of a P2P application under heavy flash-crowd conditions. Our second source of information consists of data collected during one day with a modified client that participated to the same torrent (BitTorrent session). This second log allows us to study the direct interactions between the clients, and to assess the efficiency and fairness of the BitTorrent protocol. The remaining of the paper is organized as follows. In Section 2, we present the main features of BitTorrent. In Section III, we review the related work. In Section 4, we present the results obtained from the tacker log and in Section 5 the conclusions obtained from our client log. We conclude with future directions in Section 6.

13.2 BitTorrent

BitTorrent is a P2P application that capitalizes the *bandwidth* of peer nodes to efficiently distribute large contents. The clients involved in a torrent (the download of a single large file) cooperate to replicate the file among each other using *swarming* techniques (i.e., a server sends different parts

of a file to different clients, and the clients exchange chunks with one another). BitTorrent focuses on fair bandwidth allocation by keeping each peers download rate proportional to its upload rate.

Unlike other well-known P2P applications, such as Gnutella or Kazaa, which strive to quickly *locate* hosts that hold a given file, the sole objective of BitTorrent is to quickly *replicate* a large file on a set of clients. The challenge is thus to maximize the speed of replication.

A user can join an existing torrent by downloading a *torrent* file (usually from a Web server), which contains meta-information indicating a so-called "tracker". The tracker is not involved in the actual distribution of the file ; instead, it keeps track of the peers that are currently active and acts as a rendez-vous point for all the clients of the torrent. The tracker is the only centralized component of the system.

Active clients report their state to the tracker periodically (every 30 minutes in steady-state regime), or when disconnecting from the torrent, to update the tracker's global view of the system. A new client can bootstrap by obtaining from the tracker a list of active peers to connect to and cooperate with. Typically, the tracker provides 50 peers chosen at random among active peers while the clients seeks to maintain between 20 and 40 active connections. A torrent can thus be viewed as a collection of interconnected sets of peers. A peer that joins a torrent will obviously be the youngest peer in its initial set, but it may later be contacted by younger peers. The peer may also be contacted by older peers since peers try to keep contact with a minimum number of peers throughout their lifetimes in the system. This temporal diversity is a key component in the efficiency of the BitTorrent protocol, since it guarantees with high probability that a given peer will find other peers (younger or older) that hold some missing parts of the file.

To improve cooperation between clients, the initial file is broken into fixed-size blocks (typically 256kB each). Each time a client obtains a new block, it informs all the peers it is connected with. Interactions between clients are primarily guided by two principles. First, a "tit-for-tat" strategy is used to encourage cooperation and ban "free-riding" [1] : a peer preferentially sends data to peers that reciprocally send data to it.

The second principle is a "choke/unchoke" policy that limits the number of peers being served simultaneously (5 peers are typically served while the other peers are choked). Connections are periodically reevaluated (normally, every 10 seconds) in order to always select the best peers to send data to, based on fair exchange and on the perceived upload and download rates. An optimistic unchoke that give a select one peer in the list of choked peer is triggered every 30 seconds in order to evaluate new hosts. Also, to give a chance to newcomers (peers that join the system with no chunk at all) to obtain a first chunk, BitTorrent introduce some bias in the optimistic unchocke strategy to give them a higher chance of being serviced.

The last important feature of BitTorrent is the chunk selection algorithm. The main objective is to maximize the entropy of each chunk in the torrent at any time instant. The heuristics used to achieve this goal is that a peer always seeks to download the less duplicated chunks among the chunks it misses in its peer set (keep in mind that peers only have a local view of the torrent). This policy is called the rarest first policy. There exists an exception to this policy when a peer with no chunks joins a torrent. Since this peer needs to quickly obtain a first chunk (through optimistic unckoke), it should not try to ask for the rarest chunk because this is the chunk with the smallest number of servants. This is why a newcomer uses a random first policy for the first chunk and then turns to the rarest first policy for the next ones.

13.3 Previous Work

Approaches to replicate contents to a large set of clients can be classified as client side and server side approaches. The first client-side approach was to cache contents already downloaded by clients of a given network. Extension to hierarchies of caches [7] has also been proposed, though deployments are generally limited to single ISP domains.

A symmetric approach on the server side is to transparently redirect clients to a set of mirror sites run by a content provider (e.g. Akamai - `http://www.akamai.com`).

The peer-to-peer paradigm has been applied to obtain client side and server side solutions. On the server side solutions, one finds proposals like [4] where overlay nodes dynamically organizes themselves so as to obtain a tree with a maximum throughput. FastReplica [5] is designed to efficiently replicate large contents on a set of well-known and stable overlay nodes.

On the client side, one finds many proposals to build a multicast service on top of existing lookup layers [8, 6] or as ad-hoc solutions [2]. A solution similar to BitTorrent is Slurpie [9]. Slurpie aims at enforcing cooperation among clients and thus aleviating the load of a Web server that is the primary source of the content. The algorithms of Slurpie are much more complex than the ones of BitTorrent and require a rough estimation of the number of peers in the Slurpie network. The expected gains over BitTorrent is less load on the topology server (equivalent to the BitTorrent tracker) and on the primary source (original seed) through a back-off algorithm. Results are promising since Slurpie is able to outperform BitTorrent in a controlled environnement. Still, the actual performance of Slurpie for real large scale transfers are unkown.

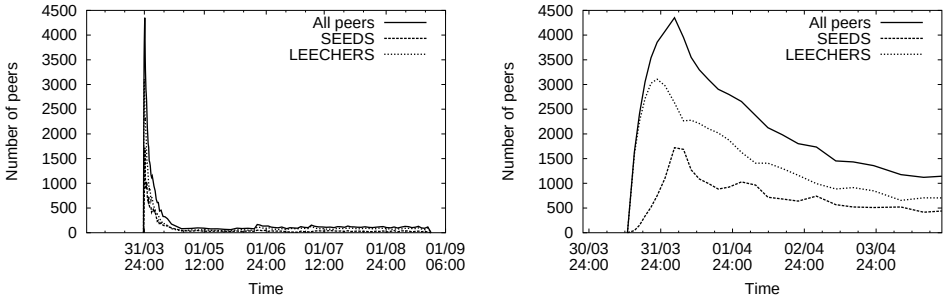
13.4 Tracker Log Analysis

The tracker log covers a period of 5 months. The corresponding torrent consists of the transfer of the 1.77 GB Linux Redhat 9 distribution. $180,000$ clients participated to this torrent with a peak of $51,000$ clients during the first five days (see figures 13.1(a) and 13.1(b)). These first five days thus clearly constitutes a flash-crowd. As clients periodically report to the tracker their current state, along with the amount of bytes they have uploaded and downloaded, the tracker log allows us to observe the global evolution of the file replication process among the peers.

13.4.1 Seeds and Leechers

We distinguish between two kinds of peers depending on their download completion status : clients that have a complete copy of the file and only serve other peers are called *seeds* ; clients that are still downloading the file are called *leechers*. Seeds apply the choke/unckoke policy (they cannot apply the tit-for-tat policy) to always try to service the fastest downloaders. This strategy makes sense in a system like BitTorrent where we do not know how long seeds will remain connected. It is then normal to replicate data as soon as possible, that is to the best downloaders. A consequence of this principle is that seeds favor clients with the best access links.

Observing the tracker log, our first finding is that BitTorrent clients are very altruistic in the sense that they actively send data to other clients, both as leechers and as seeds. Altruism is enforced



(a) Complete trace

(b) Zoom on the first five days

FIG. 13.1 – Number of peers with respect to time

during the download phase by the tit-for-tat policy, as a selfish client will be served with a very low priority. Once they become seed, the peers remain connected around six hours and a half on average. This good social behavior can be explained by two factors : first, the client must be explicitly terminated after download completion, which might well happen while the user is not at his computer, e.g., overnight ; second, as the content being replicated is perfectly legal, the user has no particular incentive to quickly disconnect from the torrent. In fact, the presence of seeds is a key feature of BitTorrent, since it greatly enhances the bandwidth capacity of the system and its ability to scale to large client populations. Over the 5 months period covered by the log file, we observed that the seeds have contributed more than twice the amount of data sent by leechers (see figure 13.2). We also observed that the proportion of seeds is consistently higher than 20%, with a peak at 40% during the first 5 days (see figure 13.3). This last figure clearly illustrates that BitTorrent can sustain a high flash-crowd since it can quickly create new seeds. To put it differently, in situations where new peers arrive at a high rate, the resources of the BitTorrent system are not divided evenly between clients, which would result, like in a processor sharing queue under overload, in no peers completing the download. On the contrary, older peers have a higher priority since they hold chunks, which gives them more chance to complete the download and become seeds for newcomers. Obviously, this strategy benefits from the cooperation of users that let their clients stay as seeds for long periods of time.

13.4.2 Individual Sessions Performance

A fundamental question concerning BitTorrent is its raw performance as opposed to a classical mirroring approach. We found that the average download throughput of the leechers was consistently above 500kb/s. This impressive figure indicates that most of the BitTorrent clients had good connectivity (at least ADSL), which is not surprising given the total size of the file. Moreover, BitTorrent exhibits good scalability : during the initial flash-crowd, the average download throughput was close to 600kb/s and the aggregate download throughput of all simultaneously active leechers was higher

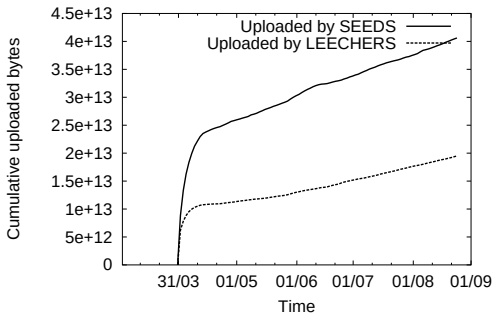


FIG. 13.2 – Volumes uploaded by seeds and leechers

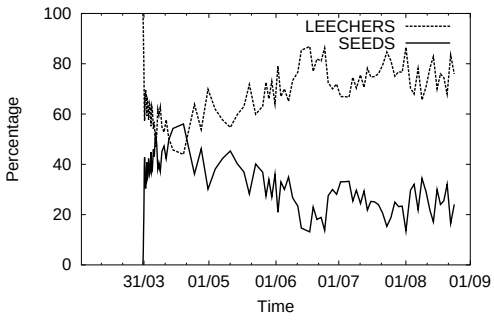


FIG. 13.3 – Proportions of seeds and leechers

than 800Mb/s. In comparison, we would need more than 80 mirror servers with 10Mb/s bandwidth capacity each to obtain a similar throughput.

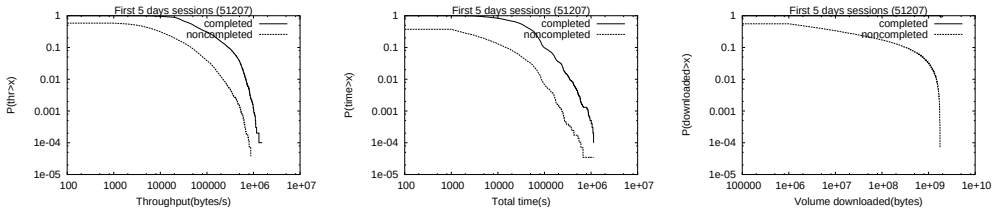
BitTorrent is thus able to capitalize the bandwidth of end system hosts to globally achieve a very high throughput. The next natural question concerns the efficiency of BitTorrent. Efficiency will be a function of the number of peers that complete the download and the fraction bytes that peers that do no complete the transfer downloaded. Since BitTorrent allows users to suspend and resume the download at any time, a peer might complete the download after one or several sessions. In the tracker log, a session id identifies the session (hash of the IP address of the host and its current time), along with the IP address of the host as seen by the tracker. Thus, identifying single session downloads is easy based on the session id. To reconstruct multi-sessions downloads, we assumed that the IP address of the host does not change from one session to another. NATs often used port numbers to disambiguate connexions and not IP addresses, so our assumption can hold even in this case. Of course, if two or more hosts behind a firewall simultaneously participate to the same torrent, it might be difficult to disambiguate them. In addition, since a peer provides the amount of bytes it has downloaded in each of its reports (sent every 30 minutes), we mated two sessions with the same IP if the amount of bytes of the last report for the first session is close to the amount of bytes of the first report of the second session. These two values are not necessarily the same if for instance, the user disconnected the BitTorrent client improperly which results in the latter not sending its disconnection report with its current amount of bytes downloaded. A set of sessions form a multi-sessions download if the download of the file is completed during the last session. Note that we observed some sessions that started with already 100% of the file downloaded. This is the case if the user is kind enough to rejoin the torrent after the completion of the download to serve leechers. We thus categorize the sessions into four categories : single session download, multi-session download, seed sessions and session that never complete (incomplete downloads). We provide statistics for these sessions in table 13.1.

Overall, table 13.1 indicates that only 33% of the sessions are part of a transfer that will eventually complete. However, the efficiency in terms of effective work is very good since the 62% of sessions that do not complete the transfer represent only 11.4% of the total amount of downloaded bytes. Note also that it is difficult to assess the reason behind the abortion of a transfer. It might be because of the user eventually decidies he is not interested in the file or because of the bad performance he experienced. To investigate this issue, we obtained statistics (download throughput,

Type	Number of sessions	Total downloaded (TB)	Total uploaded (TB)	Download per session (MB)	Upload per session (MB)
Single session downloads	20,584	38.1	31.3	1,851	1,521
Multi-sessions downloads	39,839	12.5	10.1	314	255
Incomplete downloads	112,869	65.6	46.5	58	41
Seed sessions	8,555	-	14.0	-	1,632
Total	181,847	57.2	60.0	330	330

TAB. 13.1 – Sessions types

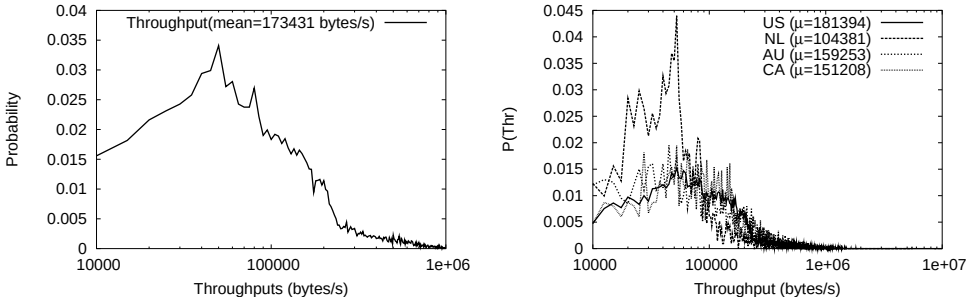
durations and volumes) for sessions that start with 0% of the file during the first five days. We classify these sessions into two categories denoted “completed” and “non completed” (see figure13.4). The completed category corresponds to the single session downloads defined previously while the non-completed category corresponds to transfers that never complete or the first session of transfers that will eventually complete. We can observe from figure 13.4 that the download throughputs achieved by non-completed sessions are significantly smaller than the ones of completed sessions while 90% of these non-completed sessions remain less than $10,000$ seconds in the torrent (and 60% less than 1000 seconds) and retrieve less 10% of the file. Thus, we can at least conclude that these uncompleted sessions are not long transfers that result in consistently very low throughputs. They are generally short, and due to the way BitTorrent works (the longer a peer stays, the more chunks he obtains, and the more trading he can do), it is not surprising that they obtained a throughput smaller than completed sessions.



(a) Download throughput (b) Duration (c) Downloaded volume

FIG. 13.4 – Single sessions statistics for sessions in the first 5 days

Let us now further focus on the performance achieved by single transfer downloads. The average download throughput of these $20,584$ peers is close to 1.3Mb/s over the whole trace, which is larger than the throughput achieved by leechers irrespective of their degree of completion of the download. The average download time for these peers is about $30,000$ seconds overall. Such values reveal a high variability in the throughputs achieved by these peers since if they all had a throughput of 1.3Mb/s, the download time would be $\frac{1.77\text{GB}}{1.3\text{Mb/s}} \sim 10,000$ seconds. This is confirmed by the distribution of these download throughputs (see figure 13.5(a)) that clearly exhibits a peak around 400kb/s (a typical value for ADSL lines), a value significantly smaller than the mean.



(a) Global throughput distribution

(b) Throughput distribution per country

FIG. 13.5 – Single session downloads

13.4.3 Geographical Analysis

The tracker log provides the IP addresses of the clients. Based on these addresses, we have used NetGeo (http://www.caida.org/tools/utilities/netgeo/) to obtain an estimation of the origin country of the peers that participated to the torrent. The estimation might be rough as NetGeo is not maintained any more. Overall, we were not able to obtain information for around 10% of the hosts. In table 13.2 we indicate the top five countries for respectively, the first 5 days, the first four weeks and the complete trace. We can observe that this set of countries, as well as the ranking is consistently the same at all these time scales. Most of the clients are US clients while Europe is represented only through the Netherlands. Still, for Netherlands, 50% of the hosts originate from ripe.net (an information also provided by NetGeo), which acts as the Regional Internet Registry for Europe. Thus we can guess that these peers are spread all over Europe (and possibly Africa, Middle East and Asia also).

Countries	First week	First four weeks	Complete trace
United States	44.6%	44.3%	32%
Netherlands (Europe)	14.9%	15.3%	23.9%
Australia	12.7%	12.6%	17.8%
Canada	5.7%	5.8%	4.9%
% of total hosts	77.9%	78%	78.6%

TAB. 13.2 – Geographical origins of peers

We next investigate the relative performance of the four clusters identified above (US, NL. AU and CA). To do so, we consider again the $20,584$ peers that complete the download in a single session. $17,000$ peers out of the $20,584$ initial peers, are in the four clusters (11498 are in US cluster, 2114 in NL, 1995 in AU and 1491 in CA). We plot in figure 13.5(b) the distribution of throughputs achieved by the peers in each cluster. We can clearly notice that the NL cluster is outperformed by

the other ones, which can indicate that clients in the US, Canada and Australia have better access links than in Europe.

13.5 Client Log Analysis

To better observe the local behavior of BitTorrent peers, we have run an instrumented client behind a 10Mb/s campus network access link. Our client joined the torrent approximatively in the middle of the 5 months period (i.e., far from the initial flash crowd). We experienced a transfer time of approximately $4,500$ seconds (i.e., much lower that the average download times previously mentioned) and our client remained connected as a seed for 13 hours. Our client wrote to a log file detailed information about the upload or download of individual blocks. In figure 13.6 we represent the number of peers with whom our host is trading. At time 0, our client knows already around 40 peers whose addresses have been provided by the tracker. We next continuously discover new peers (peers that contacted us) up to the end of the download (at time $4,500$ seconds) where we observe a sudden decrease of the number of peers, most probably because the seeds we where connected to have closed their connection to us by the moment we have completed the download. After the download, we stay connected as seed and service most of the time between 80 and 95 leechers.

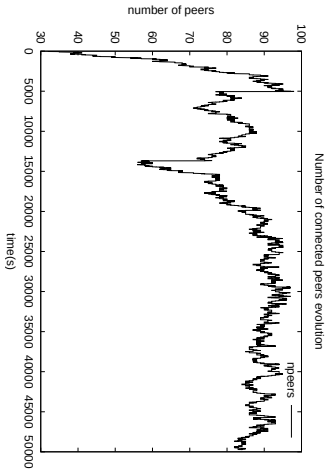


FIG. 13.6 – Number of peers during and after download

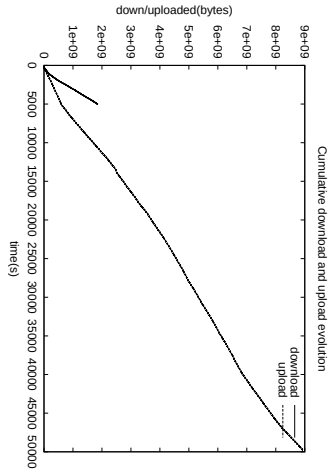


FIG. 13.7 – Complete torrent

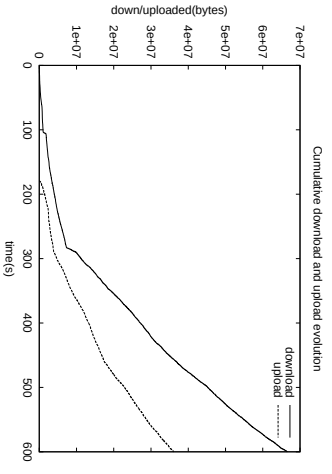


FIG. 13.8 – First 10 minutes of download

Figures 13.7 and 13.8 show the amount of downloaded and uploaded bytes with respect to time for the complete trace and the first 10 minutes of the trace respectively. From these figures, we can draw several conclusions :

- There is a warm-up period (around 100 seconds) to obtain some first chunks, but as soon as we have obtained a few chunks, we are able to start uploading to others peers. This means that the rarest first policy works well, otherwise we would maybe not find anyone interested in our first chunks.
- The download and upload rates are (positively) correlated, which indicates that the tit-for-tat policy works. It also indicates that we always find peers interested in our chunks and peers

from which we can download chunks. Otherwise, we would observe some periods where the curves are flat, indicating that our client is stalled. Also, the way peer sets are built with "old" and "new" peers mixed together must have a positive impact on the efficiency of BitTorrent since it allows the trading between peers and the exchange of fresh chunks between peer sets.
– It takes twice as much the download period to upload the same quantity of bytes (1.77GB) to the system. This illustrates the importance of peers staying as seeds once they have completed the download. This means also that we downloaded at a faster rate than we uploaded. This is due to the fact that we have a high speed link and thus, as explained in section 13.4.1, we should be consistently favored by the seeds that serve us.

We further investigate the type of clients we have been trading with. Overall, we found that approximately 40% of the file was provided by seeds and 60% by leechers. We also observed that more than 85% of the total file was sent by only 20 peers, including the 8 seeds that provided 40% of the file. An interesting remark is that these 20 top uploaders were not in our initial peer set provided by the tracker, but contacted us later. We can thus conjecture that to obtain the best possible performance with BitTorrent, clients should not be behind firewalls and NATs that prevent inbound connections.

We eventually want to assess the efficiency of the tit-for-tat policy. A good tit-for-tat policy should enforce clients to exchange blocks with one another, but with enough flexibility so as not to artificially block transfers when data does not flow at the same rate in both directions. BitTorrent avoids this type of problem by chocking/unchoking connections every 10 seconds—a long period at the time scale of a single TCP connection—and by having each client serve at least 5 other peers at a given time. We have studied the correlations between upload and download throughputs, as well as between sent and received traffic volumes. Results show that, while traffic *volumes* are well correlated (positive correlation close to 0.5), the upload and download *throughputs* are only slightly correlated (close to 0), which means that BitTorrent is flexible. We also computed the correlations between downloaded volumes and downloaded throughputs on one side and uploaded volumes and uploaded throughputs on the other side. Both correlations are high : 0.9 and 0.5 respectively. The high correlation observed between downloaded throughputs and volumes is probably due to the fact that the top three downloaders are seeds that provide 29% of the file. The uploaded throughputs and volumes are also correlated because, once our client become a seed, it continuously seeks for the best downloaders. It is not the case during the download phase where one primarily seeks for peers that have some chunks we are interested in.

13.6 Conclusion

Large content replication has become a key issue in the Internet for residential users as well as content providers that manage large overlay networks. Large companies may also be interested by a large scale replication service for updating large sets of hosts (e.g., virus patches or OS updates). There are currently very few proposals that address this problem. To the best of our knowledge, BitTorrent is the only peer-to-peer application targeted only at large file replication, though other popular peer-to-peer systems like Kazaa or eMule, seek to speed up transfers using parallel downloads. We have extensively analyzed a large torrent (with up to a few thousands simultaneously active clients) over a large period of time. The performance achieved, in terms of the throughputs per clients during download and the ability to sustain high flash-crowd, demonstrate that BitTorrent is highly efficient.
There still remain open questions in the area of large content replication such as : (i) what is the

optimal (at least theoretically) replication policy, in terms of response times, for a given user profile (arrival rates, willingness to stay as seeds), (ii) how to build a robust replication service, i.e. to ensure that all machines eventually complete the downloads (a must for corporate usage), (iii) how to efficiently protect these application against denial of services.

Bibliographie

[1] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[2] Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *ACM SIGMETRICS 2000*, pages 1–12, Santa Clara, CA, USA, June 2000.

[3] B. Cohen. Incentives to build robustness in bittorrent. Technical report, http ://bitconjurer.org/BitTorrent/bittorrentecon.pdf, May 2003.

[4] John Jannotti, David K. Gifford, and Kirk L. Johnson. Overcast : Reliable multicasting with an overlay network. In *Proc. 4-th Symp. on Operating Systems Design and Implementation*. Usenix, October 2000.

[5] J. Lee L. Cherkasova. Fastreplica : Efficient large file distribution within content delivery networks (usits 2003). In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, March 2003.

[6] Sylvia Ratnasamy, Mark Handley, Richard M. Karp, and Scott Shenker. Application-level multicast using content-addressable networks. In *NGC 2001*, pages 14–29, November 2001.

[7] Pablo Rodriguez, Christian Spanner, and Ernst W. Biersack. Analysis of web caching architectures : Hierarchical and distributed caching. *IEEE/ACM Transactions on Networking*, 9(4) :404–418, August 2001.

[8] A. Rowstron et al. Scribe : The design of a large scale event notification infrastructure. In *Proc. NGC 2001*, November 2001.

[9] Rob Sherwood, Ryan Braud, and Bobby Bhattacharjee. Slurpie : A cooperative bulk data transfer protocol. In *Proceedings of IEEE INFOCOM*, March 2004.

Troisième partie

Sous Projet 4

Méthodologie et Echantillonnage

METROPOLIS

Métrologie Pour L'Internet

Projet RNRT Exploratoire

Sous Projet 4
Echantillonnage et Mesures Actives

Livrable
Rapport d'avancement

Coordinateur du sous-projet : Ramon Casellas.

Auteurs : GET : Ramon Casellas, François Roueff, Géraldine Texier, Tijani Chahed

LIP6 : Jérémie Leguay, Timur Friedman, Kavé Salamatian, Artur Ziviani

2003/10/16

Groupe des Écoles de Télécommunications — GET

Laboratoire d'Informatique de Paris 6 — LIP6

Chapitre 14

Introduction

14.1 SP4 : Échantillonnage et Mesures Actives

L'Internet étant très vaste, il n'est pas possible de mesurer la qualité de service offerte à chaque flux individuel à tout moment et en tout point du réseau. Il est donc nécessaire de développer des techniques d'échantillonnage qui permettent de réduire l'étendue (géographique, temporelle, etc) des données à traiter pour avoir une vision fiable de la qualité de service dans le réseau. Le problème de l'échantillonnage consiste précisément à trouver un bon compromis entre le coût du traitement et la fiabilité des résultats obtenus.

L'échantillonnage peut s'effectuer dans la dimension spatiale ainsi que dans la dimension temporelle. Le premier cas est lié au positionnement des points de mesures (aussi bien actives que passives) dans le réseau. Cette répartition doit être effectuée en fonction de critères pragmatiques comme la facilité d'accès aux ressources nécessaires, ainsi que de critères plus statistiques comme la possibilité de généraliser les résultats obtenus à des parties du réseau où aucune mesure n'a été effectuée.

Un second aspect de l'échantillonnage concerne la procédure de choix des instants auxquels les observations sont faites. Cet échantillonnage a pour principal objectif une réduction considérable du volume de données à traiter. La problématique d'échantillonnage temporelle se pose de façon différente en fonction de l'échelle d'observation. En effet, une méthode de mesure doit non seulement déterminer la périodicité des campagnes de mesures à effectuer pour suivre l'état d'un réseau, mais aussi définir la durée d'une mesure et de façon plus fine, la distribution des intervalles inter-sondes dans le cadre de mesures actives, par exemple.

L'échantillonnage temporel a une importance particulière dans le cadre des mesures actives. En effet, dans ce cas, plus le nombre d'échantillons est important, plus la mesure est fidèle, mais en contrepartie, l'augmentation du trafic de mesure perturbe l'objet de la mesure. Il convient ainsi de déterminer dans le cadre d'une méthode de mesure active, une discipline d'échantillonnage temporel qui, en fonction de la charge du réseau, permet d'atteindre une estimation fiable et robuste des métriques de mesure active sans pour autant perturber l'objet de la mesure qu'est le réseau.

L'échantillonnage pose la question de l'interpolation (ou de l'extrapolation) des mesures effectuées à des instants, à des emplacements ou sur certaines classes de trafic à d'autres flots pour

lesquels la mesure n'a pas été effectuée. Cette problématique qui a été investiguée largement dans le cadre du traitement de signal, n'a pas encore été étudiée en profondeur dans le cadre de la métrologie dans les réseaux. Néanmoins certaines recommandations prennent en compte ces aspects. Par exemple l'utilisation d'un échantillonnage de type poissonnien dans les recommandations de l'IETF est motivée simplement par la propriété PASTA que possède cette discipline d'échantillonnage. Néanmoins, la validation de cette recommandation reste à faire et passe par le développement d'une vraie théorie de l'échantillonnage pour le réseau semblable à celle qui est aujourd'hui bien maîtrisée pour les signaux classiques mais prenant en compte le fait qu'un réseau est un système fortement non-linéaire.

Le domaine de l'échantillonnage présente de nombreux problèmes qu'il convient de résoudre afin de simplifier la métrologie sur les réseaux IP. Dans ce cadre, il est nécessaire de dériver une théorie de l'échantillonnage spatial et temporel approprié pour le réseau.

14.1.1 Objectifs du sous-projet

Dans ce sous projet nous souhaitons le développement d'une vraie théorie de l'échantillonnage pour le réseau semblable à celle qui est aujourd'hui bien maîtrisée pour les signaux classiques mais prenant en compte le fait qu'un réseau soit un système fortement non-linéaire.

Pour l'échantillonnage temporel, nous visons une analyse des avantages et inconvénients des approches consistant à évaluer les métriques sur un flux de mesure ou sur le flux usager directement (éventuellement par insertion de paquets permettant de délimiter des bloques de paquets usagers), ainsi que le choix de la loi d'échantillonnage selon la métrique et/ou le type de flux d'usager auxquels on s'intéresse et la conception de techniques spécifiques à certaines métriques, telles que le Packet Pair.

Enfin, l'obtention de méthodes de métrologie dans un contexte MPLS dans lequel la notion de « connexion » peut faciliter l'implémentation des approches d'OAM conçues dans le contexte ATM (en adaptant le fait que les « chemins » MPLS (LSPs) sont unidirectionnels) sont aussi visés.

14.2 Structure du Document

Ce document est structuré en deux parties : d'abord nous détaillons les travaux et études qui sont déjà terminés et ensuite, nous faisons une synthèse des travaux et études en cours de conception et/ou réalisation.

Dans ce contexte, une double problématique a été abordé : tout d'abord, la définition d'une méthodologie d'échantillonnage, concernant les mécanismes, algorithmes et procédures pour la réalisation d'un échantillonnage efficace et optimal par rapport à un certain critère donné, puis la définition de méthodes de traitement et d'analyse, a posteriori, des données et mesures obtenues. Ces deux problématiques sont souvent dépendantes : une analyse peut imposer des contraintes sur la méthodologie d'échantillonnage à définir et inversement. Des contraintes autour d'une méthodologie d'échantillonnage peuvent limiter l'analyse des données.

Chaque chapitre présenté dans ce document a, en plus ou moins grande mesure, évoqué ces deux aspects.

Nous commençons par l'estimation par mesures actives d'une file M/M/1, dont une première contribution concerne l'évaluation des performances d'estimation pour un certain type de modèles (files d'attente) et une deuxième contribution sous forme d'étude expérimentale se concentre sur l'estimation des paramètres d'une file M/M/1 bruitée à partir d'une campagne de mesures actives.

Postérieurement, nous présentons «la localisation géographique dhôtes Internet» (cf. chapitre 16), un service de localisation géographique basé sur une méthodologie d'échantillonnage utilisant des mesures actives de délai.

Ensuite, nous énumérons les travaux en cours tout en faisant une synthèse de leur état d'avancement : le chapitre 17 a comme but de relier la modélisation et l'analyse du trafic TCP au niveau paquet et au niveau flot en corrélant les mesures actives et passives à plusieurs niveaux. Le chapitre 18 porte sur l'estimation de la capacité effective (terme à définir) du processus de service de bout en bout d'un chemin de données.

La majorité des travaux réalisés dans le contexte du SP4 portent sur l'échantillonnage actif à une exception près : le chapitre 19, qui présente la problématique de l'échantillonnage temporel passif, visant à maximiser le nombre de flots «éléphants» détectes.

Finalement, le chapitre 20 porte sur la tomographie de l'Internet : la définition d'une méthodologie d'échantillonnage qui utilise le logiciel *traceroute* pour la détection de chemins entre deux points terminaux et qui au moyen d'un traitement postérieur vise l'obtention de la topologie du réseau ciblé.

Les plates-formes de mesure actives qui ont été déployées seront présentées au rapport dvancement du SP7, «Infrastructure de Mesures».

Sous Projet 4
Echantillonnage et Mesures Actives
Etudes et expérimentations réalisées

Groupe des Écoles de Télécommunications GET
Laboratoire d'Informatique de Paris 6 LIP6

Chapitre 15

Estimation par mesures actives d'une file M/M/1

15.1 Résumé

L'objectif de ce chapitre est d'étudier le problème d'estimation suivant : quels sont les paramètres d'un trafic utilisant un lien à partir de l'observation des délais de paquets-sondes envoyés sur ce lien ? Les observations obtenues sont modélisées comme la somme d'un délai constant, un temps d'attente et un un bruit blanc. Le délai constant est le délai minimal pris par un paquet pour parcourir le lien (c'est-à-dire quand aucun trafic externe ne perturbe ce paquet). Le temps d'attente est modélisé par une file d'attente de type M/M/1 à deux paramètres. Enfin, le bruit additif prend en compte les imprécisions de mesures et sera modélisé par un bruit blanc gaussien. Le paramètre du modèle est donc de dimension quatre (temps minimal, paramètres M/M/1 et variance du bruit). On étudie la mise en oeuvre, l'efficaité et les limites de deux procédures d'estimation : l'une basée sur la méthode des moments, l'autre sur la maximistaion d'une approximation de la vraisemblance. Les procédures sont utilisées sur des mesures réelles de délais par sonde active et ne requierent pas d'attention particulière, sauf sur un point : le trafic des sondes est supposé ne pas influencer le comportement de la file d'attente (par example en supposant que les paquets sondes sont de faible taille et que le trafic sonde est très faible en regard du trafic global). Comme la dépendance des délais n'est pas prise encompte, la méthode d'échantillonnage n'influence pas les estimateurs considérés. En revanche elle peut largement influencer leur performance. Cette influence fait l'objet de l'étude préalable décrite ici dans un cas simple : en absence de bruit additif. Il en ressort que du point de vue de l'information, le schéma d'échantillonage ne joue pas un grand rôle et il s'en suit que dans la perspective d'estimateurs basés sur la loi marginale des observations, les schémas du type poissoninen ou régulier semblent les plus appropriés.

15.2 Introduction

Une campagne de mesure active consiste à envoyer des sondes d'un point à un autre du réseau et à recueillir les données auxquelles celles-ci permettent d'accéder : délais, pertes, informations de routage, etc. L'objectif de ces mesures est de permettre d'estimer des paramètres physiques du réseau ainsi que les paramètres du trafic empruntant la même route que les sondes. Les travaux fondateurs étudiant les possibilités offertes par ce type de mesures remontent à un peu moins d'une décennie [11], [12], [52]. A partir de modèles probabilistes de files d'attente (qui rendent compte de la dynamique des délais dans le réseau), l'estimation des paramètres de réseau par mesure active se ramène à un problème d'estimation statistique des paramètres d'une file d'attente échantillonnée à des instants discrets $\{C(t_k),\, t_1 < \cdots < t_n\}$. Dans ce cadre, une solution simple [8] consiste à utiliser des estimateurs de moments (de la forme $f(1/n \sum_{l=1}^{n} g(C_{t_l}))$) basés sur la loi stationnaire du modèle considéré. Dans le cas de la file d'attente M/M/1 considéré ci-dessous, $1/n \sum_{l=1}^{n} \mathbf{I}(C(t_l) > 0)$ permet (par exemple) d'estimer le taux de charge de la file. Toutefois même pour ces estimateurs de forme très simple, la question des performances d'estimation et, en particulier, de leur dépendance vis à vis du choix des instants d'observations $t_1, \ldots, t_n$ n'a pas de réponse simple. Cette problématique, souvent évoquée dans ce type de problème, concerne le choix des instants d'envoi des sondes. Dans un cadre plus général, nous avons étudié l'impact de la répartition temporelle des observations d'une file d'attente sur l'estimation de ses paramètres. Nous ne détaillerons que le cas, plus simple, où les observations portent sur le nombre de clients (ou tâches) présentes dans la file d'attente à des instants donnés. Le cas où les observations correspondent à des mesures du temps de traitement, aussi appelé "charge" du serveur (dans le cas des réseaux, celui-ci est lié la taille totale des données présentes dans la file d'attente), est plus complexe mais il sera aussi abordé. Les solutions que nous apportons sont de nature algorithmique et permettent l'évaluation numériques des performances statistiques. Ces méthodes apportent de nouveaux résultats concernant l'efficacité relative de certains schémas d'échantillonnage utilisés en métrologie des réseaux (réguliers, en paires, etc.).

La première contribution est consacrée à l'évaluation des performances d'estimation (bornes d'estimation ainsi que performances d'estimateurs particuliers) pour des modèles à temps continu de type "file d'attente" observés à des instants discrets et de manière irrégulière. Cette question pourrait se poser en ces termes : comment utiliser au mieux la dépendance temporelle des observations des délais dans une file d'attente pour en augmenter la valeur informative.

Une seconde contribution concerne une étude expérimentale qui se concentre sur l'estimation des paramètres d'une MM1 bruitée à partir d'une campagne de mesures actives. Cette étude met en place une technique d'estimation qui nécessitera d'être adaptée à la forte variabilité du comportement du trafic au niveau paquet.

Une hypothèse importante de ces travaux consiste à considérer que le processus de mesure utilisé ne perturbe pas l'état du système. En pratique, il est clair que l'envoi de paquets sondes sur le réseau est susceptible de modifier, au moins marginalement, son état [8]. Dans un premier temps, il nous semble toutefois justifié d'investiguer une technique de mesure supposée non intrusive, quitte ensuite à s'assurer que tel est le cas.

15.3 Information et échantillonage

Cette partie reprend et complète la contribution [16] de O. Cappé et F. Roueff, TSI - Get / Telecom Paris.

15.3.1 Contexte

Dans la suite nous considérerons une file d'attente du type M/M/1. Soit $(C(t))_{t\in\mathbb{R}^+}$ la chaîne de Markov à temps continu qui représente l'évolution du nombre de clients dans la fille d'attente. On suppose que l'on dispose de mesures de l'état du processus à des temps $t_1, \ldots, t_n$. La principale difficulté posée par ce modèle statistique est que les observations ne fournissent qu'une observation très partielle du système. A contrario, si l'on observait toute la trajectoire (à temps continu) entre les dates t_1 et t_n, l'estimation au sens du maximum de vraisemblance se ramènerait à problème bien maîtrisé. Ce problème est concrètement posé par le fait que les matrices de transition de la chaîne échantillonnée sont plus difficiles à manier pour le calcul des quantités habituellement utilisées en théorie de l'estimation. Ce type de question a été étudié dans le contexte de l'estimateur par maximum de vraisemblance des processus Markovien de sauts (voir [21]). Procédons à un rappel très succinct sur la file M/M/1. Le lecteur pourra avec profit se reporter aux nombreux ouvrages existants (citons [9]). Une file d'attente modélise un système de traitement de tâches. Celles-ci arrivent à des temps modélisés par un processus d'arrivée et à chacune d'elles est assigné un temps de traitement. La stratégie de traitement considérée est PAPS (premier arrivé, premier servi). Pour une file M/M/1 les arrivées, disons à partir de $t = 0$ (l'état de la file étant donné à l'instant $t = 0$ par des conditions initiales), sont modélisées par un processus ponctuel de Poisson homogène d'intensité λ marqué par des temps de traitements i.i.d. de loi exponentielle de paramètre μ et indépendants des temps d'arrivées. Le processus du nombre de clients $(C(t))_{t\in\mathbb{R}^+}$ est un processus de vie et de mort, les clients arrivant aux instants définis par le processus de Poisson des arrivées déjà mentionné, et sortant (tant que la file est non-vide) à des instants correspondant à un processus de Poisson indépendant d'intensité μ. Nous considérons le cas où la file est stable c'est-à-dire, $\lambda < \mu$. Dans ces conditions, la file converge géométriquement vers sa loi stationnaire et repasse régulièrement vers l'état vide (en un temps d'espérance finie).

15.3.2 Information de Fisher

Considérons une observation $\mathbf{X}_n = (X_0 \ \cdots \ X_n) \in \mathcal{X}^{(n+1)}$ associée à un modèle statistique paramétrique dominé $\{\mathbb{P}_\theta = p_\theta(\cdot)\, d\gamma(\cdot)\,,\ \theta \in \Theta\}$ où $\Theta \subset \mathbb{R}^d$. On définit la matrice d'information de Fisher par

$$\mathbf{I}_n(\theta) = \mathbb{E}_\theta \left[\nabla_\theta L_\theta(\mathbf{X}) \nabla_\theta^T L_\theta(\mathbf{X})\right] = \mathrm{cov}_\theta \left(\nabla_\theta L_\theta(\mathbf{X})\right), \tag{15.1}$$

où $L_\theta \stackrel{\mathrm{def}}{=} \log p_\theta$. Sous des hypothèses supplémentaires d'intégrabilité uniforme des dérivées, on a de plus $\mathbf{I}_n(\theta) = -\mathbb{E}_\theta \left[\nabla^2_{\theta_{\theta^T}} L_\theta(X)\right]$. L'inverse de la matrice d'information de Fisher permet de minorer la variance des estimateurs sans biais de θ ("borne de Cramer-Rao") et fournit, dans les modèles réguliers, un équivalent asymptotique de la variance de l'estimateur du maximum de vraisemblance quand le nombre d'observations n tend vers l'infini. Dans la suite, nous nous ramènerons toujours (éventuellement par discrétisation et troncature) au cas où $\mathcal{X}$ est un ensemble fini (de cardinal noté r) et l'on ne fait donc pas figurer la mesure de domination γ dans les expressions ci-dessous.

On s'intéresse ici au cas particulier où $\mathbf{X}_n$ correspond à l'observation d'un nombre fini de valeurs successives d'une chaîne de Markov inhomogène, à valeur dans un ensemble $\mathcal{X}$ fini. Plus précisément,

$$\mathbb{P}_\theta(x_0, x_1, \ldots, x_n) = \nu_\theta(x_0) \prod_{k=0}^{n-1} P_{\theta,k}(x_k, x_{k+1}),$$

$\{P_{\theta,k} : \mathcal{X}^2 \to [0,1],\ k \in \mathbb{N}\}$ étant une séquence de matrices de transition paramétrées par θ, ν_θ la distribution initiale (elle aussi paramétrée par θ). Pour une mesure ν et une matrice de transition P, la notation νP désigne classiquement la mesure définie par $\sum_x \nu(x)P(x,\cdot)$ (les mesures sont donc assimilées à des vecteurs lignes). L'élément d'indices x, x' de la matrice de transition $P_{\theta,k}$ correspond à $\mathbb{P}_\theta(X_{k+1} = x' | X_k = x)$. Soient $I_\nu(\theta)$ la matrice d'information de Fisher associée à ν_θ et $I_k : \mathcal{X} \times \Theta \to \mathbb{R}^{2d}$ la matrice d'information conditionnelle associée à la loi $P_{\theta,k}(x,\cdot)$, c'est-à-dire,

$$I_k(x,\theta) = \mathrm{cov}_\theta \left(\nabla_\theta \log P_{\theta,k}(x, X_{k+1}) \right).$$

Par conditionnements successifs, il est aisé de vérifier que la matrice d'information de Fisher a une structure additive (en le nombre d'observations) :

$$\mathbf{I}_n(\theta) = (I_\nu(\theta) +) \sum_{k=0}^{n-1} \mathbb{E}_\theta \left[I_k(X_k, \theta) \right]. \tag{15.2}$$

Dans la suite, la partie entre parenthèses (liée à la condition initiale) ne nous intéressera pas dans la mesure où, lorsque le nombre d'observations n augmente, elle est clairement négligeable face au second terme.

On considère dorénavant le cas où l'on échantillonne un processus de Markov homogène à temps continu $(X(t), t \geq 0)$, de loi paramétrée par θ, à des instants $t_0, \ldots, t_n$. De par la propriété de Markov, on est ramené dans le contexte étudié précédemment à condition de considérer les noyaux de transition

$$P_{\theta,k}(x,x') = \mathbb{P}_\theta(X(t_{k+1}) = x' | X(t_k) = x).$$

L'inhomogénéité est ici causée uniquement par le fait que le processus à temps continu n'est pas (nécessairement) échantillonné de manière régulière. Si Q_θ désigne le générateur infinitésimal associé au processus à temps continu [9], on dispose de l'expression explicite suivante

$$P_{\theta,k} = \exp\left[(t_{k+1} - t_k) Q_\theta \right], \tag{15.3}$$

où $\exp$ désigne l'exponentielle matricielle. Une autre conséquence importante, est que dans ce cas particulier, les matrices de transition $P_{\theta,k}$ bien qu'elles soient inhomogènes, possèdent toutes la même loi stationnaire qui est celle du processus à temps continu $X(t)$ (que nous noterons π_θ). En particulier, si la loi initiale ν_θ coïncide avec π_θ, l'espérance qui figure dans l'équation (15.2), se fait toujours par rapport à la loi π_θ puisque $X_k = X(t_k)$ suit la loi π_θ (quel que soit l'indice k). Nous nous placerons dans ce cas par la suite. Il est alors clair que l'ordre des termes figurant dans la somme de l'équation (15.2) peut être modifié sans que la matrice d'information de Fisher ne soit modifiée. Par exemple, le schéma d'échantillonnage "en paires" $(t_{2i} = i, t_{2i+1} = i + \epsilon)$ pour $i = 0, \ldots, \ell$ (avec $\epsilon < 1$) conduit à la même matrice d'information que $(t_i = i\epsilon)_{0 \leq i \leq \ell}$ suivi de $(t_i = \ell\epsilon + (i+1)(1-\epsilon))_{0 \leq i \leq \ell}$ (échantillonnage à une cadence rapide, puis lente). Rappelons une fois encore que l'on ne considère ici que les performances statistiques en ignorant les perturbations apportées au trafic par la mesure (aspect sous lequel les deux schémas évoqués ci-dessus sont probablement assez distincts).

15.3.3 Méthodes de calcul numérique

On considère maintenant diverses approches permettant d'évaluer numériquement la matrice d'information de Fisher donnée par (15.2) dans le cas de l'échantillonnage d'un processus de Markov à temps continu stationnaire. D'après la discussion qui précède, il suffit alors de parvenir à évaluer le terme générique de (15.2) qui s'écrit en utilisant (15.1) et (15.3)

$$I_{(h)}(\theta) = \sum_{x,x' \in \mathcal{X}} \pi_\theta(x) P_{\theta,(h)}(x,x') \nabla_\theta L_{\theta,(h)}(x,x') \nabla_\theta^T L_{\theta,(h)}(x,x'),$$

où $P_{\theta,(h)} \stackrel{\text{def}}{=} \exp(hQ_\theta)$ et $L_{\theta,(h)} \stackrel{\text{def}}{=} \log P_{\theta,(h)}$, $h \in \mathbb{R}^+$ désignant l'écart générique entre deux instants d'échantillonnage successifs.

Nous avons mis en œuvre trois méthodes différentes permettant d'évaluer numériquement cette information de Fisher :

Evaluation numérique directe Tous les environnements de calcul numériques comportent en général une fonction permettant de calculer directement l'exponentielle matricielle. La méthode réputée la plus fiable numériquement utilise une approximation de type Padé [48]. On peut donc calculer directement $P_{\theta,(h)}$ et évaluer approximativement $\nabla_\theta \log P_{\theta,(h)}$ par différences finies (en évaluant l'exponentielle matricielle en d points "proches" de θ). Cette approche a le mérite d'être facilement implémentable et de ne requérir aucune connaissance sur la dépendance en θ de Q_θ (autre que d'être capable de l'évaluer numériquement). Elle est toutefois lourde à mettre en œuvre quand $\mathcal{X}$ est un ensemble de cardinal r élevé et fournit une évaluation du gradient dont la précision est impossible à quantifier précisément (dépend de la dérivée seconde dans la direction des perturbations).

Décomposition propre et calcul opérateur Une autre solution consiste à calculer numériquement une décomposition propre de Q_θ sous la forme $Q_\theta = V_\theta D_\theta V_\theta^{-1}$ où D_θ est une matrice diagonale contenant les valeurs propres $(d_{\theta,k})_{1 \le k \le r}$. Dès lors, on peut évaluer $\exp(hQ_\theta)$ sous la forme $V_\theta D_{\theta,(h)} V_\theta^{-1}$ où $D_{\theta,(h)}$ est une matrice diagonale dont les éléments valent $\mathrm{e}^{hd_{\theta,k}}$. Par ailleurs, l'exponentielle matricielle admet également une représentation spectrale sous la forme

$$\exp(hQ_\theta) = \frac{1}{2i\pi} \int_\Gamma \mathrm{e}^s (sI - Q_\theta)^{-1} ds,$$

où Γ est un contour de $\mathbb{C}$ englobant les valeurs propres d_k [32]. En utilisant la formule des résidus, il est possible de montrer que (ici pour un paramètre θ scalaire)

$$\frac{\partial}{\partial \theta} \exp(hQ_\theta) = V_\theta \left\{ \left[V_\theta^{-1} \left(\frac{\partial}{\partial \theta} Q_\theta \right) V_\theta \right] \odot G_{(h)} \right\} V_\theta^{-1}$$

où $G_{(h),kk} = h\,\mathrm{e}^{hd_k}$ et $G_{(h),kl} = (\mathrm{e}^{hd_k} - \mathrm{e}^{hd_l})/(d_k - d_l)$ pour $k \neq l$ (le symbole $\odot$ désigne le produit dit de Hadamard ou terme à terme). Cette méthode, qui suppose de savoir dériver (analytiquement) Q_θ par rapport à θ fournit une forme relativement explicite de la dérivée de Q_θ. Elle est également moins coûteuse en temps de calcul en cas d'évaluations répétées pour des valeurs de h différentes puisque la décomposition propre n'est calculée qu'une fois. Elle est malheureusement moins stable numériquement [48], en particulier pour des grandes valeurs de r.

Discrétisation temporelle Lorsque h tends vers 0, on sait que $P_{\theta,(h)}$ est approximativement égal à $\widehat{P}_{\theta,(h)}$ où $\widehat{P}_{\theta,(h)}$ est une matrice de transition dont le terme d'indices k, l (pour $k \neq l$) vaut $hQ_{\theta,kl}$ [9].

En fixant un pas de discrétisation temporelle $1/l$ (avec $l \gg 1$), on peut donc approximer $P_{\theta,(h)}$ par

$$\widehat{P}_{\theta,(h),l} = \prod_{i=1}^{\lfloor h/l \rfloor} \widehat{P}_{\theta,(1/l)}. \tag{15.4}$$

Cette méthode est liée à celle (plus générique) qui consiste à tronquer le développement en série entière de $\exp(hQ_\theta)$ mais elle garantit que l'approximation de $P_{\theta,(h)}$ obtenue est bien une matrice de transition. On peut montrer que cette méthode est stable dans le sens où pour tout h, et toute loi initiale ν,

$$\|\nu\widehat{P}_{\theta,(h),l} - \nu P_{\theta,(h)}\|_1 \leq C_\theta/l,$$

C_θ étant une constante qui ne dépend que de r et de $\max_{x\in\mathcal{X}}\{-Q_\theta(x,x)\}$. Pour évaluer les dérivées, on utilise simplement une relation de récurrence obtenue en dérivant (15.4) par rapport à θ. Cette dernière approche a le défaut d'être sensible à la valeur absolue des taux $Q_\theta(x,x')$, une discrétisation plus fine devenant nécessaire lorsque les taux augmentent. Elle a toutefois le mérite de montrer qu'il est possible d'approcher la vraisemblance (et ses dérivées) par un calcul récursif, ce qui suggère une possible implémentation (approchée) adaptative de l'estimateur du maximum de vraisemblance.

15.3.4 Simulations et résultats pour le processus du nombre de clients

A titre d'exemple, nous considérons ici le processus du nombre de clients $(C(t), t \in \mathbb{R}^+)$ associé à la file d'attente M/M/1 dans laquelle le processus d'arrivée des clients est de taux λ et le processus de traitement de taux μ.

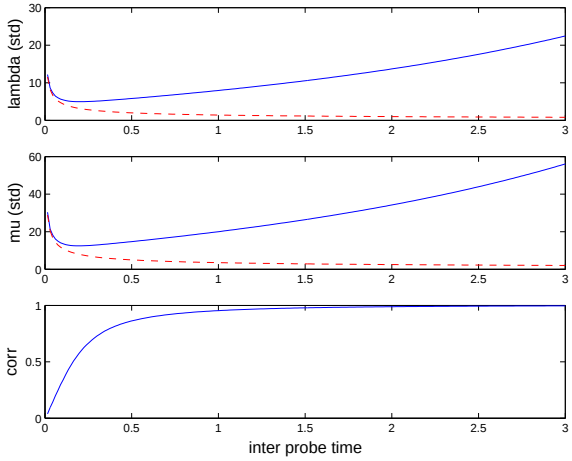


FIG. 15.1 – De haut en bas : Ecarts-types pour λ, μ et coefficient de corrélation en fonction de h. Superposés en tirets, les comportement théorique des écarts-types pour les faibles valeurs de h.

Les figures 15.1 et 15.2 montrent les résultats obtenus pour $\lambda = 2, \mu = 5$ en utilisant la troisième méthode d'évaluation numérique mentionnée au paragraphe précédent, avec $l = 500$. Notons que pour l'implémentation, il est également nécessaire de tronquer l'espace d'état du système, les figures 15.1 et 15.2 correspondent en fait à un modèle M/M/1/20 (ou le nombre maximal de clients est

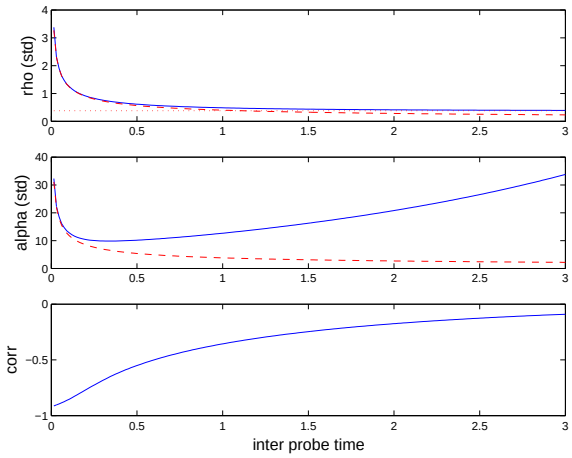


FIG. 15.2 – De haut en bas : Ecarts-types pour ρ, α et coefficient de corrélation en fonction de h. Superposés, en tirets, les comportement théorique des écarts-types pour les faibles valeurs de h et, en traits pointillés, le comportement de l'écart-type sur ρ pour les fortes valeurs de h.

20), mais cette troncature est sans effet sur les résultats présentés ici. Sur ces figures, on représente les éléments de $I_{(h)}^{-1}(\theta)$ plutôt que ceux de $I_{(h)}(\theta)$ en prenant la racine carré pour les termes diagonaux pour obtenir des grandeurs homogènes à des écarts-types et la forme normalisée (coefficient de corrélation) pour le terme hors diagonale.

La figure 15.1 concerne la paramétrisation originale $\theta = (\lambda, \mu)$. Sur ces courbes, on constate que les performances d'estimation sont optimales pour des valeurs de h de l'ordre du temps moyens entre deux événements (arrivée ou départ) dans la file étudiée. Les performances se dégradent très rapidement lorsque h augmente avec des erreurs fortement corrélées sur λ et μ. En effet, la file étant stable, $P_{\theta,(h)}$ tend assez rapidement vers la loi stationnaire du système, or cette dernière (loi géométrique de paramètre λ/μ) ne dépend que du rapport λ/μ. Pour les faibles valeurs de h, il est possible de faire une analyse simplifiée en considérant que les trois seuls événements de probabilités significatives susceptibles de se produire pendant la durée h sont l'arrivée d'un client (probabilité λh), le départ d'un client (probabilité μh, sauf si la file est déjà vide), où le fait que l'état de la file ne change pas. Sur la base de ce raisonnement, on obtient

$$I_{(h)}(\theta) = \begin{pmatrix} h/\lambda & 0 \\ 0 & (\lambda/\mu)h/\mu \end{pmatrix} + O(h^2),$$

qui correspond aux courbes représentées en traits pointillés.

Compte tenu de ces résultats, il est plus sain de considérer la paramétrisation $\theta = (\rho, \alpha) = (\lambda/\mu, \mu - \alpha)$ comme sur la figure 15.2. Le plus frappant ici est que les performances d'estimation pour ρ s'améliorent de manière monotone avec h pour atteindre (pour les grandes valeurs de h) l'information correspondant à la loi stationnaire $I_\pi(\rho) = 1/(\rho(1-\rho)^2)$. La figure 15.2 suggère que pour l'estimation de ρ, l'écart temporel entre les observations importe peu, du moment qu'il est suffisant, et que la situation la plus favorable correspond au cas de quasi indépendance entre les observations successives (grandes valeurs de h). Pour la différence des taux α, les performances d'estimation sont faibles et se dégradent assez rapidement lorsque h augmente. Il est à noter que

dans le contexte de la mesure active de réseau où l'observation se fait par émission d'un paquet sur le réseau, la plage de h qui semble la plus favorable sur la figure 15.2 (temps de l'ordre du temps moyen entre deux événements) correspond en fait à des régimes tous à fait irréalistes dans lequel le trafic généré par les paquets sondes n'est plus du tout négligeable face au trafic réel de données. Par conséquent, la partie la plus pertinente de la courbe est la partie droite qui montre une sévère dégradation pour les performances d'estimation de α (lorsque h augmente). Qualitativement, on obtient les mêmes conclusions pour toutes les files M/M/1 nettement stables (telles que ρ soit nettement inférieur à 1) ce qui correspond au cas le plus représentatif pour les mesures de trafic de bout en bout.

15.3.5 Extensions au processus de charge

Les méthodes numériques proposées ci-dessus peuvent être utilisées pour le processus de charge $W(t)$ de la file M/M/1 (qui est un processus de Markov à temps continu).

Proposition 1 (Résultat d'approximation) *On note $(X_H(t))_{t\in\mathbb{R}^+}$ la séquence de processus à temps continu avec espaces d'états $\{0,\dots,K \stackrel{\text{def}}{=} \lfloor W_M H\rfloor\}$, distribution initiale ν_H et générateurs*

$$Q(i,i-1) = H \qquad \text{(for } i \geq 1\text{)}$$
$$Q(i,i+j) = \lambda(1-\mathrm{e}^{-\mu/H})\,\mathrm{e}^{-\mu(j-1)/H} \qquad \text{(for } 0 \leq i \leq K-1, 1 \leq j \leq K-i\text{)}$$

Soit $(\nu_H)_{H>0}$ une famille de distributions initiales sur l'espace $\{0,\dots,\lfloor W_M H\rfloor\}$ telle que $\nu_H(H\cdot)$ converge étroitement vers ν quand $H\to\infty$. Alors $H^{-1}X_H(t) \stackrel{\text{fidi}}{\longrightarrow} W(t)$ quand $H\to\infty$.

On peut donc utiliser la même approche que précédemment pour calculer numériquement la matrice d'information d'un couple d'observations du processus de charge en fonction du temps h séparant ces deux observations, avec toutefois des dimensions beaucoup plus importantes (par exemple pour $\lambda = 2$ et $\mu = 5$, $H = 150$ et $K = 350$ sont nécessaires). De plus, on obtient facilement des approximations analytiques dans le cas d'échantillons très rapprochés ou très éloignés :

Pour les faibles valeurs de h Comportement identique au cas précédent (arrivée d'un paquet de loi $\mathrm{Exp}(\mu)$ avec probabilité λh).

Pour les grandes valeurs de h La loi stationnaire $(1-\rho)\delta_0 + \mathrm{Exp}(\alpha)$ dépend maintenant des deux paramètres, l'information se stabilise vers une valeur limite (avec une corrélation < 1).

Contrairement au processus du nombre de clients, l'information dans la paramétrisation (λ,μ) ne dégénère pas quand les observations sont éloignées. Cela provient du fait que ces paramètres sont identifiables à partir de la loi stationnaire des observations. De plus, il est intéressant de remarquer que la courbe des écarts-types pour λ n'est pas monotone (voir figure 15.3), ce qui signifie qu'il existe un écart optimal entre les échantillons pour l'estimation de ce paramètre. Au contraire Avec la re-paramétrisation $\theta = (\rho,\alpha) = (\lambda/\mu, \mu-\lambda)$, l'estimation de ρ et de α s'améliore de façon monotone avec h (voir figure 15.4).

Finalement, nous avons utilisé ces résultats pour comparer trois stratégies d'échantillonage (poissonien, déterministe équiréparti, déterministe sous forme de paires de paquets).

Dans le cadre de cette expérience, l'estimation des deux paramètres est satisfaisante, à peu près quel que soit le schéma d'observation. De plus cette expérience a montré que l'échantillonnage

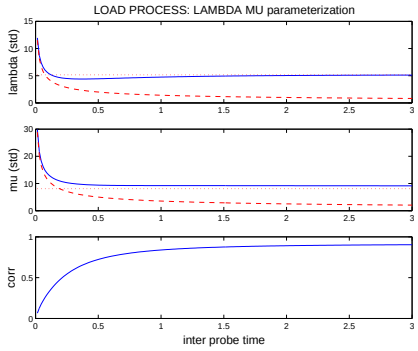


FIG. 15.3 – Processus de charge, de haut en bas : Ecarts-types pour λ, μ et coefficient de corrélation en fonction de h. Superposés en tirets, les comportement théorique des écarts-types pour les faibles et les grandes valeurs de h.

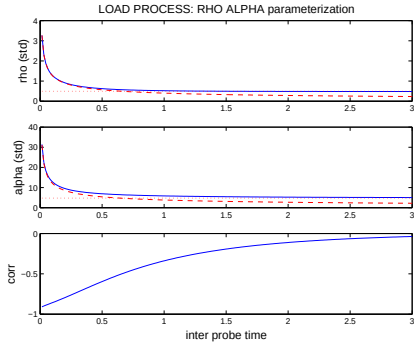


FIG. 15.4 – Processus de charge, de haut en bas : Ecarts-types pour ρ, α et coefficient de corrélation en fonction de h. Superposés en tirets, les comportement théorique des écarts-types pour les faibles et les grandes valeurs de h.

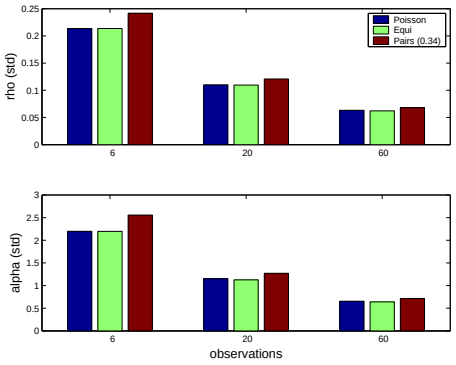


FIG. 15.5 – Les différentes stratégies d'échantillonage en fonction du nombre d'observations.

régulier associé aux simples estimateurs de moments est en fait quasi optimal (sauf pour le processus client et les taux).

15.3.6 Conclusions

L'approche présentée ci-dessus fournit une évaluation numérique des performances statistiques qui s'avère fructueuse en ce qu'elle permet effectivement de comparer divers schémas d'échantillonnage proposés dans la littérature, dans le cadre d'un modèle (Markovien) de file d'attente. Il est également possible d'étendre directement cette approche au cas du processus de charge de la file M/M/1 stable grâce à un résultat d'approximation du générateur. La méthode utilisée repose sur deux hypothèse forte : la première est l'utilisation d'un modèle de file d'attente (la M/M/1) Markovien à temps continu, la seconde consiste à négliger l'effet de l'envoi de sondes sur le trafic estimé. L'obtention de résultats aussi précis que ceux décrits ici dans un cadre plus général semble une question difficile. Enfin, la conclusion principale de ce travail indique que l'estimateur des moments basés sur la loi stationnaire à partir d'un échantillonnage régulier ou poissonnien est quasi-optimal pour l'estimation des paramètres du modèle étudié.

15.4 Estimation d'une file M/M/1 bruitée

Contribution de Jean-Charles Redoutey et François Roueff, Telecom Paris/GET.

15.4.1 Modèle considéré

Nous nous sommes basés sur des infrastructures de mesures existantes et fournissant déjà des données exploitables, notamment RIPE et l'infrastructure développée dans le cadre du projet Metropolis.

Les mesures qui serviront de base pour la suite du travail sont montrées sur la figure 15.6 : elles représentent des temps de parcours unidirectionnel espacés de l'ordre de la minute pour un lien transatlantique observé à travers RIPE. La précision des mesures est inconnue. Le fait que les machines de départ et d'arrivée sont synchronisées par GPS nous assure de l'absence de dérive due aux horloges des machines sources et destination. En revance le fait qu'il n'existe pas de délai minimum à ces mesures semble indiquer que la modélisation du délai comme la somme d'un délai constant incomprésible et d'un temps d'attente dans une file d'attente ne peut être strictement appliqué (ceci sera détaillé dans la section suivante). Le buit observé autour des basses valeurs peut-être du à la présence d'autres files moins sur le lien ou bien d'une imprécision dans l'estampillage des temps de départ et/ou d'arrivée. Dans les deux cas (en invoquant un effet de type limite centrale de la superposition de plusieurs sources d'erreur indépendantes), il nous a semblé légitime de modéliser ce bruit par une loi gaussienne. Le temps passée dans la file *principale* est modélisé par une file M/M/1 stable de paramètres $0 < \lambda < \mu$ dans l'état stationaire. Le problème du choix du meilleur modèle de représentation pour un lien réseau est un vaste problème qui mérite d'être traité indépendamment. Nous verrons les insuffisances du modèle considéré. Néanmoins, ce modèle reste exemplaire pour deux raisons. Il peut servir de base à un modèle localement stationaire. La généralisation consiste alors à expliquer les données localement par un modèle simple et de suivre l'évolution des paramètres

de ce modèle au cours du temps. Ou bien il peut servir de base à une modélisation stationnaire non-paramétrique. Dans ce cas les hypothèse sur le temps d'attente dans la file principale serait de type queue lourde et le bruit pourrait être simplement modélisé par un bruit symmétrique à queue exponentiel. Bien sûr, dans ces deux cas les problèmes d'identification sont plus complexes et leur interprétation moins évidente.

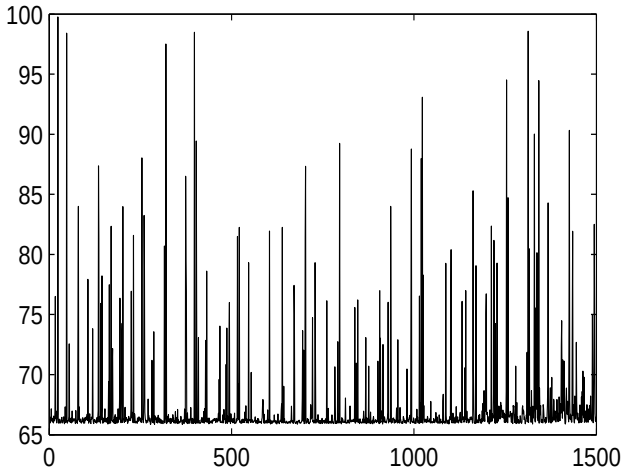


FIG. 15.6 – Temps de parcours unirectionnel sur un lien transatlantique

15.4.2 Estimateur de charge pour des mesures non-bruité

Dans la suite, on pose $\rho = \lambda/\mu$ et $\alpha = \mu - \lambda$, où λ et μ sont les paramètres de la file M/M/1. En se basant sur les résultats de base des files M/M/1, on a le résultat suivant sur la probabilité d'occupation de la file (W_t) dans l'état stationnaire :

$$P(W_t = 0) = 1 - \rho \tag{15.5}$$

Il s'en suit une estimation statistique de la charge de la file d'attente par un simple estimateur de moyenne empirique

$$\widehat{\rho} = \frac{1}{N}\sum_{i=1}^{N} I(W_i > 0) \tag{15.6}$$

Cet estimateur semble pouvoir donner des résultats intéressants puisqu'il est très simple et d'une précision qui peut être facilement chiffrable (les observations sont celles d'une chaîne de Markov inhomogène pour laquelle un théorème de la limite centrale s'applique). Le seul problème est qu'il se base entièrement sur le fait que l'on puisse estimer pour une mesure donné si elle est représentative d'une file vide ou non.

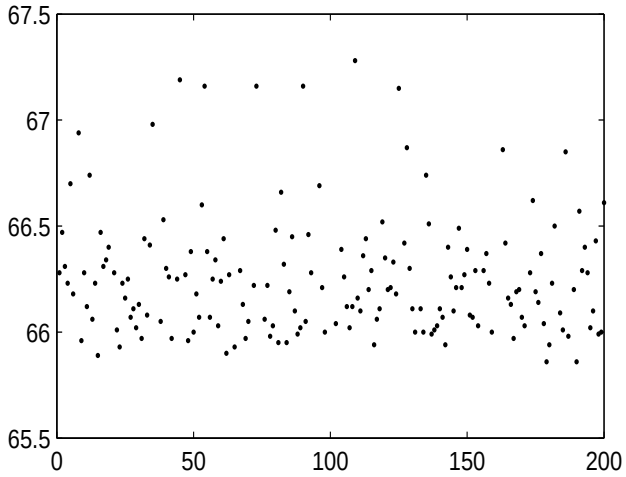


FIG. 15.7 – Détail des temps de parcours autour de la valeur minimale pour le lien transatlantique

Dans la pratique, si on note X_i les différentes mesures obtenues, celles-ci représentent

$$X_t = D + W_t \tag{15.7}$$

où D est le temps de parcours physique du lien réseau. Dans un modèle de ce type, la valeur de D peut être obtenu en prenant la plus petite valeur observée pour X_t. Si on se réfère au graphique 15.7 des mesures que l'on a grossi sur la partie stable qui semble être cette valeur seuil D et qu'on le compare à la série temporelle d'origine 15.6, il est clair qu'une telle procédure n'est pas valide à cause de la présence de variations faibles autour de la valeur minimum observée sur 15.6. Ces variations seront modélisées dans la suite par un bruit additif.

Ces variations autour de la valeur minimale du temps de parcours peuvent être dues à des évolutions physiques du lien réseau influant sur le temps de parcours des données, à des imprécisions intrinsèques aux systèmes de mesure, ou encore à une file d'attente non vide mais extrêmement peu chargé. Si ces fluctuations ne posent pas de problème pour une estimation sattistique de D elles posent clairement un problème pour estimer le paramètre ρ par un estimateur du type (15.6).

Dans la suite nous incluerons donc un bruit additif dans notre modèle, qui permettra de déterminer des estimateurs plus justifiés que (15.6) pour les données que l'on observe. Le modèle devient donc

$$X_t = D + W_t + \varepsilon_t \tag{15.8}$$

c'est à dire qu'au temps de parcours et au temps d'attente s'ajoute un temps ε_t qui représente le bruit associé aux contraintes des mesures. Dans une hypothèse de simplicité et à défaut d'avoir un modèle parfaitement déterminé pour le bruit en question, nous utiliseront un bruit gaussien centré de variance inconnue, $\varepsilon_t := N(0, \sigma^2)$. La loi marginale de (W_t) vérifie en régime stationnaire :

$$\left\{ \begin{array}{c} P(W_t = 0) = 1 - \rho \\ (W_t | W_t > 0) \sim \alpha e^{-\alpha w}\, dw \end{array} \right\} \tag{15.9}$$

A partir de ce modèle, on peut calculer la densité de probabilité marginale de X_t par rapport à la mesure de Lebesgue

$$p(x) = (1-\rho)p_0(x) + \rho p_1(x) \tag{15.10}$$

où p_0 est la densité contitionnelle de la mesure du temps de parcours du lien, en sachant que la file est vide (ce qui est de probabilité $(1-\rho)$) et p_1 est la densité conditionnelle de la même variable sachant que la file est non-vide. p_0 est donc simplement la densité gaussienne centrée en D de variance σ^2,

$$p_0(x) = \frac{e^{\frac{-(x-D)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \tag{15.11}$$

et $p1$ est la convolution de la loi marginale de la file M/M/1 stationnaire dans l'état non-vide et du bruit gaussien, soit

$$\begin{aligned} p_1(x) &= \int_0^\infty \alpha e^{-\alpha\varpi} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-D-\varpi)^2}{2\sigma^2}} d\varpi \\ &= \frac{\alpha e^{\frac{\alpha^2\sigma^2}{2}+\alpha(D-X)}}{\sqrt{2\pi\sigma^2}} \int_0^\infty e^{-\frac{1}{2\sigma^2}(\varpi+\alpha\sigma^2-x+D)^2} d\varpi. \end{aligned}$$

or

$$\int_0^\infty e^{-\frac{1}{2\sigma^2}(\varpi+\alpha\sigma^2-x+D)^2} d\varpi = \frac{\sqrt{2\pi\sigma^2}}{2}\left(1-\text{erf}\left(\frac{\alpha\sigma^2-x+D}{\sigma\sqrt{2}}\right)\right)$$

donc

$$p_1(x) = \frac{\alpha e^{\frac{\alpha^2\sigma^2}{2}+\alpha(D-X)}}{2}\left(1-\text{erf}\left(\frac{\alpha\sigma^2-x+D}{\sigma\sqrt{2}}\right)\right). \tag{15.12}$$

Soit, finalement,

$$p(x) = (1-\rho)\frac{e^{\frac{-(x-D)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} + \rho\frac{\alpha e^{\frac{\alpha^2\sigma^2}{2}+\alpha(D-X)}}{2}\left(1-\text{erf}\left(\frac{\alpha\sigma^2-x+D}{\sigma\sqrt{2}}\right)\right) \tag{15.13}$$

Concrètement, en comparant par rapport à un modèle M/M/1 classique, on a une densité de probabilité plus *arrondie* en D. Ainsi, les imprécisions auteur de la valeur que l'on pense être D sont intégrées et peuvent correspondre au modèle. On remarque de plus que, si les paramètres de la file correspondent à des temps d'attente bien supérieur au bruit, les queues positives de distribution des cas bruité et non-bruité coïncident. Ce principe peut être retenu dans le cas d'un modèle de file d'attente différent.

15.4.3 Paramètres à estimer

Les paramètres d'intérêt du modèle (15.8) sont λ et μ ou, dans une paramétrisations différentes, α et ρ. La méthode utilisée est une procédure par maximum de contraste. La fonction de con,traste retenue est celle du maximum de vraisemblance simplifiée obtenue en reprenant les marginales du modèle mais de façon indépendantes plutôt que Markovienne. Le principe est donc de trouver les paramètres $D, \rho, \alpha, \sigma^2$ définis par

$$\left(\widehat{D}, \widehat{\rho}, \widehat{\alpha}, \widehat{\sigma}^2\right) = \underset{D,\rho,\alpha,\sigma^2}{\arg\max}\left(\log\left(\prod_{i=1}^N p_{D,\rho,\alpha,\sigma^2}(x_i)\right)\right) \tag{15.14}$$

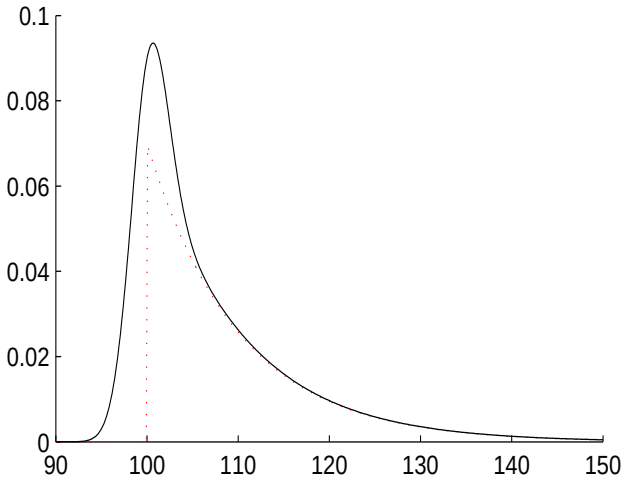


FIG. 15.8 – Densités de probabilités d'une file M/M/1 classique (rouge pointillée) et bruitée (noire plein). Les paramètres ont été ici choisis : $D = 100,\ alpha = 0.1,\ rho = 0.7$ et $sigma = 2$ dans le cas bruité .Dans le cas non-bruité, on n'a pas représenté la densité parpport à la mesure de Dirac en D.

mais cela est en fait beaucoup moins immédiat car il s'agit d'une maximisation à 4 inconnues d'équations complètement non linéaires. Afin d'éviter de passer tout de suite à une résolution numérique qui introduirait assurément des imprécisions supérieures à ce que l'on pourrait admettre. En fait du fait de la présence d'un état caché défini comme l'alternative "file vide" ou "file non-vide", la fonction à maximiser correspond à une loi de mélange. Il est dans ce cas connu que l'algorithme EM (**E**xpectation-**M**aximisation) donne des résultats satisfaisants.

15.4.4 L'algorithme EM

L'algorithme EM est particulièrement adapté dans les cas où il existe une variable cachée. Dans notre cas, cette variable estdéfinie par

- $V = 0$ quand la file est vide
- $V = 1$ quand la file est pleine.

L'algorithme EM utilise la fonctionnelle suivante

$$\left\{ \begin{array}{c} Q(\theta, \theta') = E\left[\log\left(p(X, V; \theta')\,|X = x, \theta\right]\right. \\ \theta = \left(D, \rho, \alpha, \sigma^2\right) \end{array} \right\}, \tag{15.15}$$

où, ici, $p(x, v; \theta)$ représente la densité jointe de (X, V) (par rapport à la mesure de Lebesgue tensorisée avec la mesure de comptage) sous le paramètre θ. L'objectif de l'algorihtme EM est de maximiser itérativement la fonction

$$\theta \rightarrow Q\left(\theta, \theta^k\right) \tag{15.16}$$

soit

$$Q\left(\theta^{k+1}, \theta^k\right) = \max_{\theta \epsilon \Theta} Q\left(\theta, \theta^k\right)$$

pour ainsi obtenir une suite de $(\theta)_k$ qui va converger par construction de la fonction Q vers une valeur de θ stationnaire pour la fonction de contraste utilisée en (15.14). De plus, à chaque étape, ce contraste est augmenté.

En utilisant les expressions de $p_0(x)$ et $p_1(x)$ définies respectivement en 15.11 et 15.12 et qui sont égales respectivement à $p(x|V=0;\theta)$ et $p(x|V=1;\theta)$, on obtient l'expression (on note par souci de concision et de précision, $p^{\theta}_{v=\epsilon} = p(\cdot|V=\epsilon;\theta)$) :

$$Q(\theta',\theta) = \sum_{i=1}^{N} \frac{(1-\rho)\, p^{\theta}_{v=0}(x_i) \log\left((1-\rho')\, p^{\theta'}_{v=0}(x_i)\right) + \rho p^{\theta}_{v=1}(x_i) \log\left(\rho' p^{\theta'}_{v=1}(x_i)\right)}{(1-\rho)\, p^{\theta}_{v=0}(x_i) + \rho p^{\theta}_{v=1}(x_i)} \tag{15.17}$$

Là encore, si on veut trouver directement les 4 inconnues, on est contraint de faire une résolution numérique pure, ce qui n'est pas très satisfaisant. Par contre, si on considère que les 2 inconnus $\left(D, \sigma^2\right)$ sont obtenues par une méthode différente et sont constantes au cours des étapes de maximisation de $(\theta)_k$, alors il est possible d'obtenir une expression formelle pour (ρ, α). Cela est certes une restriction puisqu'on fixe ces 2 valeurs mais si on utilise un regard plus concret, cela est acceptable : en effet, D et σ^2 sont 2 données du système indépendantes de la charge du lien réseau ; elles sont représentatives des caractéristiques physiques du lien réseau et de l'imprécision intrinsèque aux instruments de mesure. Il doit donc y avoir au moins une méthode pour déterminer ces valeurs de manière indépendante des mesures que nous utilisons que l'on exploite ainsi uniquement pour calculer des paramètres d'utilisation du réseau.

On note

$$A(\theta, x) = (1-\rho)\, p^{\theta}_{v=0}(x) + \rho p^{\theta}_{v=1}(x) \tag{15.18}$$

$$f_0(\theta, x) = \frac{(1-\rho)\, p^{\theta}_{v=0}(x)}{A(\theta, x)} \tag{15.19}$$

$$f_1(\theta, x) = \frac{\rho p^{\theta}_{v=1}(x)}{A(\theta, x)} \tag{15.20}$$

Un calcul direct, omis ici par souci de concision, donne les expressions maximisant ρ' et α' :

$$\rho' = \frac{\sum_{i=1}^{N} f_1(\theta, x)}{\sum_{i=1}^{N} f_1(\theta, x) + \sum_{i=1}^{N} f_0(\theta, x)} \tag{15.21}$$

$$\alpha' = \frac{\sum_{i=1}^{N} f_1(\theta, x)}{\sum_{i=1}^{N} \left(f_1(\theta, x) + f_0(\theta, x)\right) x - D \sum_{i=1}^{N} \left(f_1(\theta, x) + f_0(\theta, x)\right)} \tag{15.22}$$

15.4.5 La méthode des moments

Nous avons mis en oeuvre l'algorithme EM qui est réputé bon pour les problème de maximisation de vraisemblance avec données cachées. Néanmoins cette méthode itérative requiert des valeurs

initiales correctes pour donner de bons résultats. En outre, on ne dispose pas pour le moment d'une méthode extérieure pour calculer $\left(D, \sigma^2\right)$. C'est pour pallier à ces problèmes que nous avons mis en place la méthode des moments, qui est assurément moins bonne que l'algorithme EM mais qui permet de fixer des valeurs initiales correctes initiaux et surtout des estimés des paramètres de nuisance pour D et σ.

La méthode est relativement simple, il s'agit de calculer les moments d'ordre 1 à 4 de manière numérique et formelle et de résoudre le système à 4 équations qui en découle. Le système étant polynomial, des méthodes algorithmiques standardes pourront être utilisées. En se basant sur les résultats des moments d'ordre 1 à 4 d'une loi exponentielle de paramètre α

$$E\left[W_t^k | W_t > 0\right] = \frac{k!}{\alpha^k}$$

et d'une loi gaussienne centrée et de variance σ^2

$$E\left[\varepsilon\right] = E\left[\varepsilon^3\right] = 0, \quad E\left[\varepsilon^2\right] = \sigma^2 \text{ et } E\left[\varepsilon^4\right] = 3\sigma^4,$$

on peut calculer les moments d'ordre 1 à 4 de la loi $X_t = D + W_t + \varepsilon_t$ (le détail des calculs est ommis), à savoir :

$$\begin{aligned}
E\left[X\right] &= D + \frac{\rho}{\alpha} \qquad (15.23)\\
E\left[X^2\right] &= D^2 + \sigma^2 + \frac{2\rho}{\alpha^2} + \frac{2\rho D}{\alpha}\\
E\left[X^3\right] &= D^3 + 3D\sigma^2 + \frac{6\rho}{\alpha^3} + \frac{3\rho D^2}{\alpha} + \frac{6\rho D}{\alpha^2} + \frac{3\rho\sigma^2}{\alpha}\\
E\left[X^4\right] &= 3\sigma^4 + D^4 + 6\sigma^2 D^2 + \frac{24\rho}{\alpha^4} + \frac{12\rho\sigma^2}{\alpha^2} + \frac{4\rho D^3}{\alpha} + \frac{24\rho D}{\alpha^3} + \frac{12\sigma^2\rho D}{\alpha} + \frac{12\rho D^2}{\alpha^2}
\end{aligned}$$

15.4.6 Résultats numériques

La méthode d'estimation des différents paramètres caractéristiques de la charge du réseau va donc se dérouler en 2 étapes :

1. Calcul de $\left(D, \sigma^2\right)$ et première estimation de (ρ, α) par la méthode des moments

2. Itérations par l'algorithme EM pour affiner les valeurs de (ρ, α)

Nous avons aussi reporté l'volution du contraste considéré en fonction de l'itération de l'algorithme EM.

L'estimation donne :

- $D = 66.1$
- $\sigma = 1.21$
- $\alpha = 0.08$
- $\rho = 0.1$

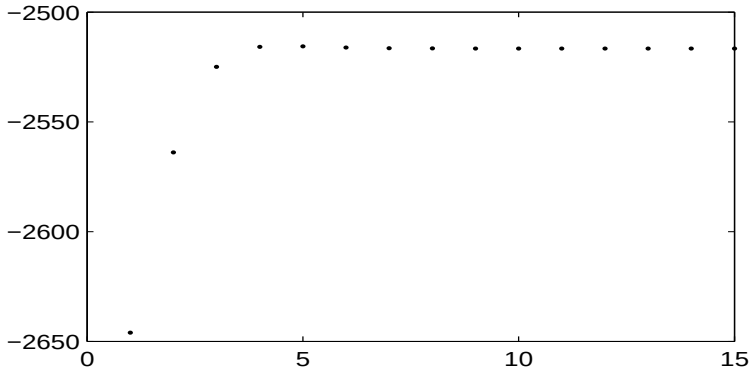


FIG. 15.9 – Evolution de la fonction de contraste à chaque itération de l'algorithme EM.

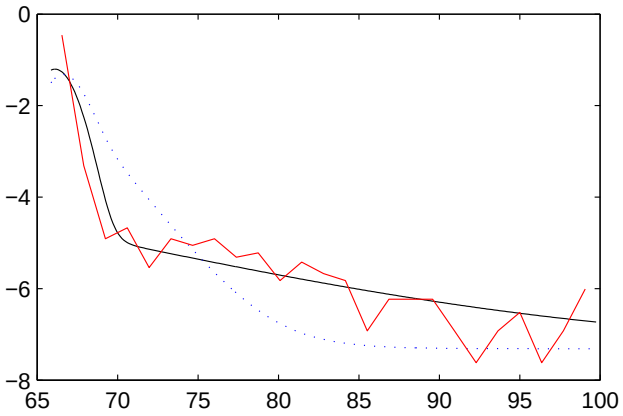


FIG. 15.10 – Log-histogramme (rouge plein), et log-densités estimés ;bleu pointillé : estimation par méthode des moments ; noir plein : estion après 15 itérations de l'algorithme EM.

D, ρ et σ correspondent bien à l'idée que l'on pouvait avoir à vue d'oeil.En revanche, les valeurs de α et ρ impliqueraient des tailles de paquets très grosse. Il y a donc de bonnes raisons de penser que par cette méthode, on n'estime non pas le comportement des délais à un niveau paquet mais à un niveau agrégé. Pour remédier à une agrégation temporelle, c'est-à-dire pour *désagrégé* les observations en temps, il serait intéressant d'utiliser cet estimateur de façon locale mais ceci nécessiterait un trafic de mesure suffisemment dense. Une autre méthode consiste à rechercher un mélange plus riche dans les données globales, en décomposant les délais comme une somme de files d'attentes indépendantes (approximation de files en tandem) avec des paramètres λ_μ différents.

15.4.7 Conclusion

Cette méthode d'estimation de la charge d'un lien réseau permet a été validé. Leur interprétation et leur adéquation aux phénomènes réellement observé sur un lien nécessite de compléter ces expériences par l'utilisation simultanée de mesures actives et passives sur des liens que l'on contrôle

parfaitement. Il est cependant vraisemblable au vu des résultats que l'on a que ce modèle doit être adapté. Il reste néanmoins une bonne base en vue de méthodes qui s'adapteraient à des hénomènes non-stationnaires (en l'utilisant comme modèle local) ou à une modélisation plus complexe (en enrichissant la modélisation par un mélange additif plus riche). En outre, le fait de pouvoir calculer D et σ de manière vraiment indépendante permettra d'affiner les résultats puisque dans le cas présent, on se sert de mêmes séries de mesure pour déterminer ces paramètres pour lesquelles des mesures mieux adaptées sont possibles.

Chapitre 16

Localisation Géographique d'Hôtes Internet

16.1 Introduction

La possibilité de connaître la localisation géographique d'un hôte Internet en fournissant uniquement son identifiant ouvre la voie pour une classe toute entière d'applications conscientes de la localisation des hôtes. Nous nous concentrons sur un service de localisation géographique d'hôtes Internet basée sur des mesures de délai. La localisation d'hôtes Internet est inférée en comparant des profils de délais d'hôtes de référence (*landmarks*) distribués qui ont leurs localisations géographiques bien connues avec le profil de l'hôte à localiser. Nous évaluons le niveau de corrélation entre la distance géographique et le délai réseau. De plus, nos expérimentations ont pour objectifs d'évaluer des propriétés de base pour un service de localisation géographique d'hôtes Internet basé sur des mesures.

Des applications conscientes de la localisation des hôtes Internet prennent en compte d'où les utilisateurs font leur accès et pour cela peuvent offrir de nouvelles fonctionnalités dans l'Internet. Des exemples d'applications conscientes de la localisation sont : la publicité ciblée dans les pages web, la sélection automatique d'un idiome pour montrer un certain contenu, le contrôle d'accès au contenu basé sur la localisation et la vérification de sécurité basé sur la localisation.

L'adoption de la distance géographique entre hôtes Internet comme une métrique de distance réseau est évaluée en [34]. Lakhina *et al.* [46] se concentrent sur la localisation géographique des ressources Internet en visant le développement de méthodes basées sur la géographie pour la génération de topologies réseau plus réalistes. Cependant, dans l'Internet actuel, il n'y a pas une relation directe entre l'identifiant d'un hôte et la localisation physique de cet hôte. Les nouvelles applications conscientes de la localisation ont besoin d'un service de localisation géographique pour les hôtes Internet.

16.2 État de l'Art

Une approche basée sur DNS pour fournir un service de localisation géographique est proposée dans le RFC 1876 [20]. Toutefois, l'adoption de cette approche est limitée parce qu'elle demande des modifications dans les registres DNS et les administrateurs ne sont pas motivés à enregistrer les nouveaux registres de localisation. Des outils tels que IP2LL [63] et NetGeo [49] font des requêtes vers des bases de données Whois pour obtenir des indices sur la localisation géographique d'un hôte. Cette information, néanmoins, peut être imprécise ainsi que n'être pas à jour. De plus, si un grand bloc d'adresses IP qui se trouvent géographiquement dispersées sont allouées à une seule entité, la base de données Whois peut contenir seulement une seule entrée concernant cette entité. Par conséquence, une requête sur la base de données Whois fournit la localisation enregistrée pour cette entité qui contrôle le bloc d'adresses IP, malgré que les hôtes puissent se trouver dispersés géographiquement.

Padmanabhan et Subramanian [51] étudient trois importantes techniques pour inférer la localisation géographique d'un hôte Internet. La première technique estime la localisation d'un hôte à partir du nom DNS du hôte ou d'un nœud près de cet hôte. Cette technique est la base de GeoTrack [51], VisualRoute [64], and GTrace [14]. Fréquemment les opérateurs réseau attribuent à ses routeurs des noms qui ont une signification géographique. Par exemple, le nom `bcr1-so-2-0-0.Paris.cw.net` indique un routeur qui se trouve à Paris. La deuxième technique divise un espace d'adressage IP en groupes plus petits d'une façon que tous les hôtes qui se trouvent dans chacun de ces groupes soient co-localisés. En connaissant la localisation de quelques hôtes d'un groupe et en supposant qu'ils sont en accordance, la technique infère la localisation du groupe entier. Un exemple de cette technique est le GeoCluster [51]. La troisième technique est basée sur des mesures de délai et l'exploitation d'une possible corrélation entre le délai réseau et la distance géographique. Cette technique est la base de GeoPing [51]. L'estimation de la localisation d'un hôte suppose que les hôtes ayant des délais similaires vers quelques machines sondes fixes ont une tendance à être co-localisés. Par conséquence, en utilisant un ensemble de hôtes de référence (*landmarks*) ayant une localisation géographique bien connue, l'estimation de la localisation d'un hôte cible est la localisation du *landmark* qui présente le profil de délai le plus similaire à celui du hôte cible.

Dans ce travail, nous nous concentrons sur la technique de localisation basée sur des mesures de délai. Nous identifions deux points clés qui jouent sur la précision de l'estimation de la localisation. Cette précision dépend du placement des hôtes de référence et des machines sondes [66] ainsi que de l'évaluation de la similitude observée parmi les profils de délai mesurés [67].

16.3 Inférence de la Localisation Géographique d'un Hôte Internet

Nous formalisons le problème d'inférer la localisation géographique d'un hôte Internet à partir de mesures de délai de la façon suivante. Supposons un ensemble $\mathcal{L} = \{L_1, L_2, \ldots, L_K\}$ de K *landmarks*. Ces *landmarks* sont les hôtes de référence qui possèdent une localisation géographique bien connue. Supposons un ensemble $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ de N machines sondes. Les machines sondes déterminent périodiquement le délai réseau vers chacun des *landmarks*. Ce délai réseau est en fait le RTT minimal de plusieurs mesures. Chaque machine probe $P_x \in \mathcal{P}$ a alors un vecteur

de délai $\mathbf{d}_x = [d_{1x}, d_{2x}, \ldots, d_{Kx}]^T$, où d_{ix} est le délai entre la machine sonde P_x et le *landmark* $L_i \in \mathcal{L}$. Supposons un hôte cible T à être localisé géographiquement. Un serveur de localisation qui connaît l'ensemble $\mathcal{L}$ de *landmarks* et l'ensemble $\mathcal{P}$ de machines sondes est contacté. Ce serveur de localisation demande aux N machines sondes de mesurer le délai vers la cible T. Chaque machine sonde $P_x \in \mathcal{P}$ envoie au serveur de localisation un vecteur de délai $\mathbf{d}'_x = [d_{1x}, d_{2x}, \ldots, d_{Kx}, d_{Tx}]^T$, c'est-à-dire, le vecteur de délai $\mathbf{d}_x$ plus le délai vers la cible T qui vient d'être mesuré. Après avoir reçu les vecteurs de délai provenant des N machines sondes, le serveur de localisation est capable de construire la matrice de délai $\mathbf{D}$ de dimensions $(K+1) \times N$:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \ldots & d_{1N} \\ d_{21} & d_{22} & \ldots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{K1} & d_{K2} & \ldots & d_{KN} \\ d_{T1} & d_{T2} & \ldots & d_{TN} \end{bmatrix} \tag{16.1}$$

Les vecteurs de délai reçus par le serveur de localisation des différentes machines sondes correspondent aux colonnes de la matrice de délai $\mathbf{D}$. Le serveur de localisation compare alors les lignes de la matrice de délai $\mathbf{D}$ pour estimer la localisation de la cible T. La matrice de délai $\mathbf{D}$ combinée avec la connaissance de la localisation des *landmarks* de l'ensemble $\mathcal{L}$ compose une liaison entre le délai réseau et la localisation géographique.

La fonction $\mathcal{S}(\mathbf{x}, \mathbf{y}) : \mathbb{R}^N \rightarrow [0, 1]$ est définie pour mesurer le dégrée de similitude entre deux profils de délai $\mathbf{x}$ et $\mathbf{y}$, où N est le nombre de machines sondes. Nous utilisons cette fonction pour évaluer le niveau de similitude entre les profils de délai fournis par les machines sondes pour chaque *landmark* et pour la cible à localiser. En [67], nous évaluons différents modèles de similitude pour implémenter la fonction de similitude $\mathcal{S}(\mathbf{x}, \mathbf{y})$. Une implementation possible est la distance Euclidienne entre les deux profils de délai. Pour formaliser l'évaluation de la similitude, nous définissons aussi un vecteur ligne $\mathbf{1}_i$ de dimension $K+1$ dont tous les éléments sont zéro, sauf par le $i$ème élément qui a une valeur égale à 1. Le *landmark* L qui fournit l'estimation de la localisation d'un hôte cible T est le *landmark* présentant la plus grande similitude

$$\mathcal{S}_{\max} = \arg \max_{i=1,\ldots,K} \mathcal{S}(\mathbf{1}_i \mathbf{D}, \mathbf{1}_{K+1} \mathbf{D}). \tag{16.2}$$

16.4 Expérimentations

Les expérimentations de ce travail ont pour objectif d'évaluer des propriétés de base pour un service de localisation géographique d'hôtes Internet [68]. Pour ces expérimentations, nous utilisons 9 boîtiers de mesures de la plateforme NIMI (National Internet Measurement Infrastructure) [54] comme nos machines sondes. Ces boîtiers sont géographiquement distribués de la façon suivante : 5 en Europe, 3 aux États-Unis et 1 au Japon. Des travaux récents [51, 57] indiquent que 7 à 9 dimensions suffisent pour bien représenter la distance réseau. L'ensemble expérimental de *landmarks* est composé par des machines RIPE [3], qui ont des cartes GPS, et par des serveurs universitaires [2]. Cet ensemble expérimental totalise 2335 *landmarks* qui sont distribués dans le monde de la façon suivante : 1143 en Amérique du Nord (États-Unis et Canada), 724 en Europe Occidentale, 173 en Asie, 102 en Europe de l'Est, 73 en Océanie, 64 en Amérique Latine et 57 en Afrique et au Moyen Orient.

Cette distribution représente, au moins approximativement, la distribution des utilisateurs (hôtes) à localiser.

Nous évaluons la corrélation entre la distance géographique et le délai réseau. Nos résultats montrent qu'un petit nombre de *landmarks* présentent des délais très élevés, probablement dus à un niveau faible de connectivité, ce qui mène à une corrélation faible, caractérisée par un coefficient de corrélation R=0.1138. Pour éviter la considération de ces points possédant un délai hors du commun, nous considérons les données comprises jusqu'aux 98ème, 95ème et 90ème percentiles du délai réseau mesuré. La corrélation entre la distance géographique et le délai réseau dans ces cas devient beaucoup plus forte, résultant en des coefficients de corrélation R=0.6229, R=0.8093, et R=0.8767, respectivement. Nous observons également qu'une connectivité pauvre rend la corrélation entre la distance géographique et le délai réseau plus faible. Nous identifions alors les *landmarks* localisés en Amérique du Nord et en Europe Occidentale. Ces régions offrent la connectivité la plus riche pour interconnecter leurs hôtes. Nous observons une corrélation encore plus forte dans ces régions bien connectées. Ce résultat indique que la corrélation devient plus forte lorsque la connectivité devient plus riche.

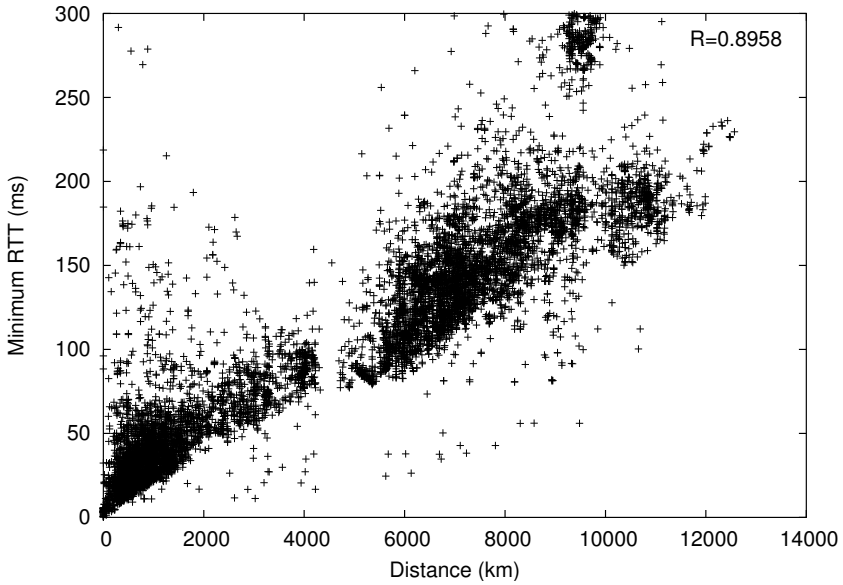


FIG. 16.1 – Corrélation entre la distance géographique et le délai réseau.

Nous comptons conduire l'analyse de la façon suivante. Nous devons évaluer la précision de l'estimation de la localisation. En choisissant comme cible à localiser un des *landmarks* à la fois et en utilisant les autres *landmark* pour localiser le premier, nous pouvons évaluer l'erreur de l'estimation. L'impact de l'utilisation de moins de dimensions, c'est-à-dire, moins de machines sondes, sur la précision de l'estimation doit aussi être évaluée.

Sous Projet 4
Echantillonnage et Mesures Actives
Etudes et expérimentations en cours

Groupe des Écoles de Télécommunications GET
Laboratoire d'Informatique de Paris 6 LIP6

Chapitre 17

Différentes échelles de modélisation et d'analyse du trafic TCP

Classification de l'étude

Type : Méthodologie d'échantillonage		
Type : Analyse de Résultats		
Classif : Echantillonnage actif temporel		
Etat : Conception et Plannification		
Date et Coord.	2004/01/08 -	- TC <tijani.chahed@int-evry.fr>

17.1 Problématique

Le but de ce travail est de relier la modélisation et l'analyse du trafic TCP au niveau paquet à un niveau supérieur, qu'on appellera flot, plus proche de celui de l'usager. Cette étude est motivée par le fait que les mesure empiriques, notamment actives, sont plus aisées et donc plus fréquentes au niveau paquet alors que la planification, le dimensionnement et l'ingénierie de la qualité de service font souvent appel à une vision orientée application.

17.2 Analyse

17.2.1 Problématique M/G/1 PS

Dans ce contexte, nous observons le développement du trafic TCP au niveau flot dans une file régie par PS (Processor Sharing). Dans ce cas, la ressource C est partagée entre les différents flots TCP présents dans le noeud d'une manière équitable (sous les hypothèses que les flots qui utilisent la même version de TCP, ont le même RTT, etc).

Soit $N(t)$ le nombre de flux en cours simultanément sur un même lien (ce nombre varie selon l'index de temps t). Avec PS, le débit par source est "proportionnel" à C/N.

17.2.2 Problématique M/M/1 FIFO locale

Suite aux résultats obtenus par [1] dans un cadre stationnaire, nous nous intéressons à l'estimation empirique "locale" du taux moyen d'arrivée λ et du taux moyen de service μ d'une file M/M/1 avec une discipline FIFO, une abstraction du lien observé. Estimation "locale" signifie que l'on utilise un modèle localement stationnaire, basé sur l'approximation suivante : sur des horizons de temps courts, le temps d'attente d'une file d'attente est celui d'une M/M/1(λ, μ). L'idée de stationnarité locale implique que λ et μ varient progressivement dans le temps. Dans ce modèle μ devient donc un paramètre dépendant du temps $\mu(t)$ qui rend compte du temps de traitement d"un paquet à l'instant t. Il semble donc naturel de le comparer à la bande passante disponible modélisée dans le modèle au niveau flot par $C/N(t)$.

17.2.3 TCP basé sur l'équation

Dans [50], le débit D de TCP est donné en fonction de la perte p par l'équation suivante :

$$D(p) = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_o \min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \tag{17.1}$$

où le RTT (Round Trip Time) est le temps entre l'émission d'un paquet TCP et la réception de l'acquittement correspondant, T_0 est le temps time-out du timer associé au paquet envoyé, b est le nombre de paquets acquittés d'une manière cumulée par un seul ACK. Cette équation considère uniquement les phases slow-start et congestion avoidance. Des versions plus simplifiées de cette formule sont aussi utilisées.

L'équation

$$D(p) = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_o 3\sqrt{\frac{3bp}{8}}p(1+32p^2)} \tag{17.2}$$

sert de borne supérieure à l'émission dans le cadre du contrôle de congestion basé sur l'équation [26]. Le contrôle de congestion basé sur l'équation vise notamment les applications temps-réels qui trouvent que TCP réagit trop sévèrement à la congestion (fenêtre d'émission divisée par 2 en cas de congestion). La paramètre p, appelé dans ce cas événement taux de perte, ne représente plus la perte mais pluôt l'inverse d'une somme moyenne pondérée s calculée à partir des taux de perte des n derniers intervalles de perte s_i. Ce calcul doit permettre un compromis entre des mesures effectuées sur des petites périodes de temps (et la rapidité de réaction à la congestion) et des mesures sur de plus longues périodes de temps (et l'obtention d'un meilleur signal).

17.3 Mesures

17.3.1 Paramètres

Le RTT [7], la perte (simple [6] ou élaborée [43]), le délai [5] ainsi que des paramètres inhérents au transfert d'un flux [47] sont définis par le groupe de travail IPPM (IP Performance Metrics) de l'IETF.

Dans ce travail, nous nous intéressons aux métriques pertinentes au niveau paquet, le délai par exemple, et celles du niveau flot, tel que le temps de transfert.

17.3.2 Précision des mesures

Nous souhaitons dans un premier temps vérifier la précision des horloges utilisées dans les mesures afin de pouvoir tirer des conclusions précises sur les résultats de celles-ci.

Pour cela, une série de mesures actives et passives d'un même flux test est requise. La comparaison entre les deux ensembles de résultats déterminera la précision des mesures.

17.3.3 Paquet versus flux

Notre but est de faire des mesures actives et passives, au niveau paquet et au niveau flot et de corréler si possible les résultats obtenus. La corrélation vise plutôt le niveau paquet versus le niveau flot (et non pas active versus passive).

Pour les mesures actives, les mesures au niveau paquet peuvent être faites avec l'outil NIMI implémenté sur les boîtiers RIPE de Metropolis. Pour le niveau flot, nous baserons nos mesures sur un nouvel outil développé par l'ENST.

Pour les mesures passives, l'idée est d'envoyer un flot f composé d'une succession de M packets P_i, $i = 1, .., M$, d'un boîtier A (LIP-6) à un boîtier B (LAAS). Les paquets P_i traversent la carte DAG A du site A à des instants de temps t_i^A et la carte DAG B du site B à des instants t_i^B.

Les temps inter-arrivées des paquets sont données par $t_i^A - t_{i-1}^B$. Le débit instantané de la source est l'inverse du temps inter-arrivée.

Le délai d_i par paquet P_i est défini par $t_i^B - t_i^A$. C'est sur l'observation du processus de délais $\{d_i)_i$ que se base la modélisation M/M/1 FIFO soit à partir de données actives (sondes) ou par enregistrement passif. Dans l'un et l'autre cas, on dispose d'un échantillon à des temps donnés du délai d'un paquet modélisé comme la somme d'un temps d'attente dans une file M/M/1 plus un délai constant, plus un, éventuellement, un bruit dû à l'imprécision des mesures et/ou de la modélisation par une unique file.

Le temps total du transfert du flot est donné par $t_M^B - t_1^A$ et est composé de deux parties : un temps d'envoi $t_M^A - t_1^A$ et d'un temps de transit dans le réseau. L'observable d'intérêt dans le cadre du modèle au niveau flot PS est le temps de séjour dans la file. En supposant que le délai d'un paquet individuel est composé de la somme du temps passé dans la file modélisée et d'un temps "incomprésible", le temps de séjour d'un flot est donné par $t_M^A - t_1^A$.

Chapitre 18

MetroWidth : Échantillonage et Mesure de la bande passante

Classification de l'étude

Type : Méthodologie d'échantillonage		
Type : Analyse des mesures actives		
Classif : Echantillonnage actif temporel		
Etat : Expérimentations préliminaires		
Date et Coord.	2003/10/10 -	- RC <casellas@infres.enst.fr>

18.1 Introduction

La caractérisation de la bande passante dans un contexte de "bout en bout" reste un problème d'importance notable, avec de nombreuses applications. Citons, par exemple, la vérification préalable avant la transmission d'un flot non élastique de la capacité résiduelle afin d'adapter le codec à utiliser, ou le contrôle d'admission pour ce type de flots.

Les techniques pour l'estimation de la bande passante totale/disponible sur un chemin de données sont diverses. Un certain nombre de ces techniques utilisent des mesures passives et des mesures actives. En ce qui concerne les mesures actives, des algorithmes comme le "Packet Pair" [42] et ses variantes sont souvent utilisées.

18.2 Contexte de nos travaux

Dans le contexte de certains projets de recherche, toujours dans l'axe de l'évolution vers l'Internet de nouvelle génération multiservice et convergent, nous avons conçu à l'ENST des mécanismes théoriques permettant la mise en place d'une ingénierie de trafic basée sur les mesures

(re-configuration dynamique de mécanismes réseau en fonction de l'état de ce dernier). Une telle approche devient nécessaire dû à la difficulté pour bien prévoir à différentes échelles de temps l'évolution du trafic

En particulier, nous avons conçu et développé des mécanismes de partage de charge adaptatifs dans un réseau MPLS. Ces mécanismes utilisent l'estimation de bandes passantes effectives et de capacités effectives. (cf. [22] or [41])

Cette étude englobe les deux aspects du SP4 : d'une part, l'adaptation de méthodologies existantes pour la mesure de la capacité de bout en bout, avec développement d'outils et logiciels sur mesure et d'autre part, l'analyse et interprétation des mesures pour la validation des modèles théoriques développés.

18.3 Définition de la Capacité Effective

De façon similaire à la définition classique de bande passante effective (ou équivalente), on peut définir la notion de capacité effective, associée à un processus de service comme la transformée de Log-Laplace du processus modélisant le travail servi par une file d'attente pendant l'intervalle de temps $[0, t)$ (noté $c[0, t)$)

$$\begin{aligned} \kappa(s,t) &\triangleq \frac{-\Lambda_t^c(-s)}{st} \\ &= -(st)^{-1} \log \mathbb{E}[e^{-sc[0,t)}] \end{aligned} \tag{18.1}$$

Propriétés de la capacité effective

- La capacité effective est décroissante avec s.
- Quand $s \to 0$, la capacité effective associée au processus de service tend vers sa capacité moyenne :

$$\lim_{s \to 0} \kappa(s,t) = \frac{\mathbb{E}[c[0,t)]}{t} \tag{18.2}$$

- Quand $s \to \infty$, la capacité effective associée au processus de service tend vers sa capacité minimale :

$$\lim_{s \to \infty} \kappa(s,t) = \frac{\operatorname{ess\,inf}(c[0,t))}{t} \tag{18.3}$$

Dans le contexte de nos travaux, ce processus de service modélise la capacité "de bout en bout" associée à une connexion (par exemple un LSP MPLS), de nature variante dans le temps due au trafic externe présent dans les liens de transmission traversés. Autrement dit, nous modélisons une connexion MPLS comme une file d'attente virtuelle à capacité variable.

La caractérisation des processus en utilisant les bandes passantes effectives et capacités effectives permet le calcul de certains paramètres de QoS sous certains régimes asymptotiques (paramètres difficilement calculables de façon exacte), dont l'optimisation permet d'obtenir la répartition optimale du trafic. bien entendu, si l'on fait des hypothèses "a priori" concernant certains modèles de trafic et de service, les problèmes d'ingénierie de trafic évoqués dans la section précédente peuvent être

résolus de façon purement théorique. La figure 18.3 illustre le modèle du système utilisé pour le calcul du partage de charge optimal.

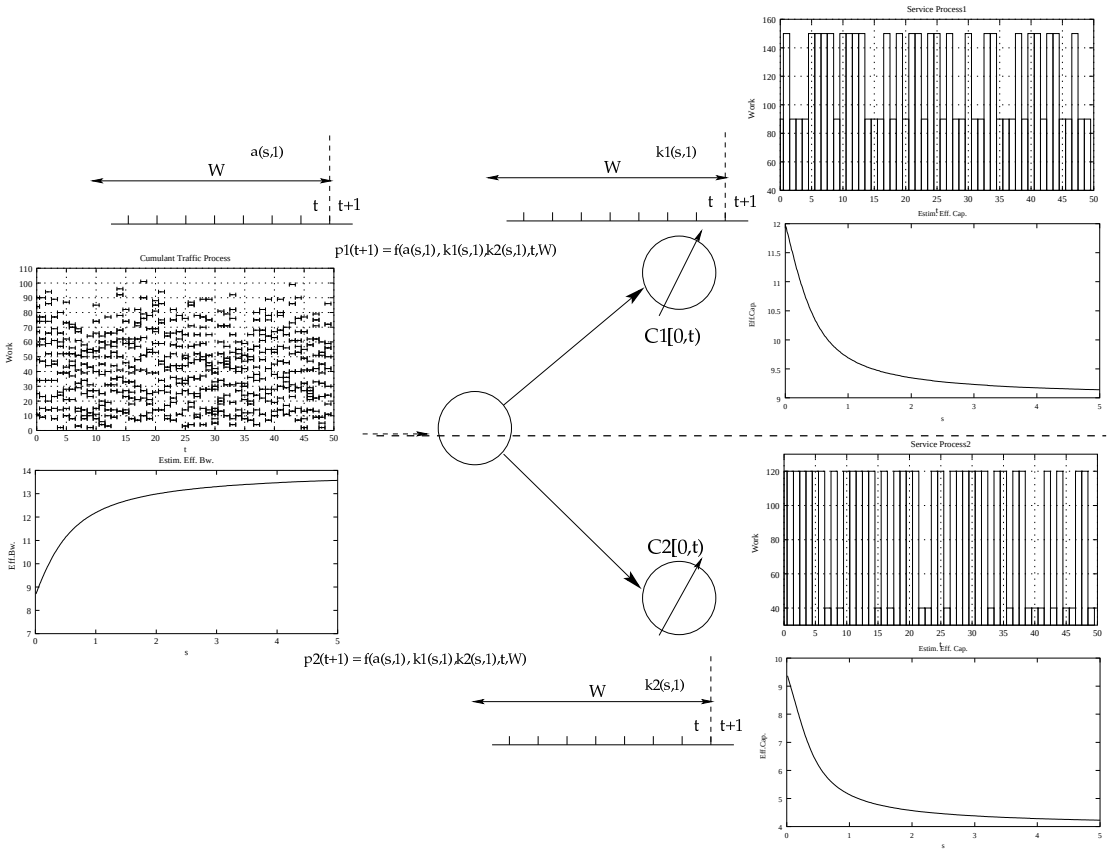
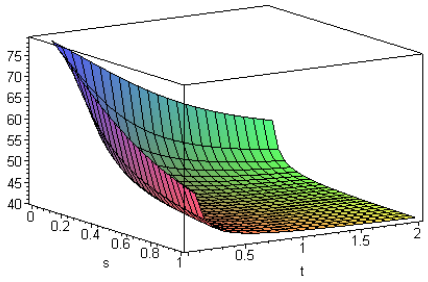


FIG. 18.1 – Modèle du système utilisé pour le calcul du partage de charge

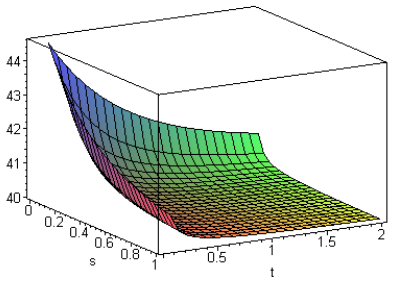
Étant donnée la complexité analytique de certains modèles de trafic (de service) et leurs bandes passantes (capacités) effectives, il est possible d'utiliser des approximations en utilisant la borne de Hoeffding [41], qui nécessite la connaissance de la valeur moyenne et leurs bornes $Cmin, Cavg, Cmax$. La figure 18.2 illustre deux exemples de capacité effective en utilisant cette borne.

Néanmoins, afin de permettre la re-configuration dynamique citée, le calcul et l'estimation de bandes passantes et capacités effectives à partir de traces devient un problème d'intérêt croissant dans la communauté scientifique.

Cet étude consiste donc en deux parties. D'une part, la conception et le développement d'un logiciel de mesure active permettant l'échantillonnage de la capacité de bout en bout en utilisant des algorithmes récents. D'autre part, l'utilisation de ce logiciel pour la caractérisation de la capacité effective entre les différentes sites partenaires du projet Metropolis et la validation des modèles théoriques cités.



(a) $Cmin = 40, Cmax = 100, Cavg = 80$



(b) $Cmin = 40, Cmax = 100, Cavg = 45$

FIG. 18.2 – Capacités Effectives : Bornes Linéaires

18.4 Méthodologie d'echantillonnage : Metrowidth

Le projet MetroWidth est un projet logiciel, ayant comme objectif le développement d'un applicatif pour l'échantillonnage et la mesure de la bande passante dans un contexte de bout en bout. (cf. `http://metrowidth.sourceforge.net`)

Plusieurs algorithmes ont été envisagés. Citons par exemple le Packet Pair ou IGI (Initial Gap Increasing), développé par Ningning Hu and Peter Steenkiste [33].

Ce logiciel a été développé dans le contexte du projet Metropolis par les élèves suivants : Vincent Strubel (Project Manager), Thomas Vuillemin (Documentation), Benoit Casoetto (Intégration), Olivier Bonnet (Analyse technique), Jean-Marc Coic (Gestionnaire des versions).

État Actuel

A ce jour, le logiciel a été implémenté et plusieurs tests préliminaires ont été réalisés.

Architecture

Le logiciel MetroWidth a été conçu de façon très modulaire, permettant l'ajout de nouveaux algorithmes de mesure et estimation de bande passante et la récupération et le traitement des données obtenues.

Le logiciel a été développé suivant le paradigme client/serveur. Le serveur *metrowidthd* doit être lance sur les machines (typiquement 2) participant à l'expérimentation. Le client est contrôlé en utilisant une console *metroshell*.

Le diagramme UML du logiciel peut être consulté ici : `http://metrowidth.sourceforge.net/doc/code.php`. Également, les auteurs ont développé une interface graphique qui facilite le contrôle et la manipulation des données. Pour plus de renseignements, le lecteur est invité à consulter la page web du projet.

18.5 Expérimentations Prévues

Expérimentations préliminaires

Plusieurs expérimentations ont été envisagées et réalisés, notamment la caractérisation de la bande passante :

- Entre deux machines appartenant ou pas au même sous-réseau.
- idem, mais en utilisant des mécanismes de lissage et de conditionnement du trafic, disponibles sous Linux.
- Entre un STA et un AP d'un canal 802.11 (WiFi).

Expérimentations en cours

Nous avons déjà commencé a mettre en place la plate-forme de test. Cette plate-forme fait intervenir plusieurs PC sous Linux dont le noyau a été étendu pour la prise en charge de l'architecture MPLS (cf. `http://mpls-linux.sf.net`). Le logiciel MetroWidth (client-serveur) permettra la mesure de la capacité instantanée des différents LSP à établir, et à partir de ces mesures, nous réaliserons l'estimation des capacités effectives nous permettant d'implémenter et de valider sous une plate-forme réelle les mécanismes de partage de charge conçus.

Chapitre 19

Échantillonnage Passif et détection de connexions TCP

Classification de l'étude

Type : Méthodologie d'échantillonage		
Classif : Echantillonnage passif temporel		
Etat : Conception et Plannification		
Date et Coord.	2004/01/08 -	- RC <casellas@infres.enst.fr>

19.1 Motivation

Cette étude a été motivée par l'intérêt d'un opérateur réseau (en particulier, le partenaire du projet Metropolis GIP-RENATER) pour concevoir, développer, évaluer et mettre en place des heuristiques et méthodes d'échantillonnage passif temporel visant à optimiser un certain critère. Cette étude concerne également le SP-6 car, comme nous allons le détailler dans la suite, ce critère d'optimisation a un rapport direct avec la validation des SLA et la tarification.

19.2 Introduction

De façon synthétique, les opérateurs réseau utilisent de façon régulière des mécanismes fournis par les différents constructeurs d'équipements réseau pour l'échantillonnage passif temporel (en mode paquet). Comme modèle de référence, nous citons le mécanisme "Netflow" développé par Cisco, qui échantillonne de façon déterministe 1 paquet sur 10. Un des objectifs de cette étude est l'analyse comparative de Netflow et des heuristiques et algorithmes alternatifs à proposer et à évaluer.

Ces opérateurs réseau ont constaté (logiquement) que ces mécanismes déterministes ne permettent pas de détecter un certain nombre de flots. A ce sujet, le problème a été formulé de la façon suivante : identifier, proposer et évaluer des mécanismes d'échantillonnage passif temporel plus performants que l'échantillonnage déterministe, sous le critère (défini par GIP-RENATER pendant les réunions de travail) de la maximisation du nombre de connections TCP détectées, sous conditions similaires (c.-à.-d. 'charge de travail', taux moyen de paquets échantillonnés, etc.). Dans la mesure du possible, les connexions TCP de caractère long (des éléphants) sont à détecter en préférence.

19.3 Contexte de nos travaux

Étant la théorie de l'échantillonnage dans le contexte de la Métrologie des réseaux très vaste : temporel/spatial, actif/passif, mode paquet (un paquet est retenu selon un critère donné) et mode flot (sous l'hypothèse qu'il est possible d'identifier des micro-flots individuels et à quel flot appartient un paquet quelconque, on retient l'intégralité d'un flot selon un critère donné), cette étude est limitée à l'échantillonnage passif temporel en mode paquet, sous l'hypothèse d'un dégrée important de multiplexage (lien d'opérateur) et optimisation du critère évoqué dans la section précédente.

19.4 Logiciel de traitement de traces : TcpTrace

Pour valider les propriétés et évaluer les performances des différentes heuristiques d'échantillonage, nous utiliserons le logiciel "TcpTrace" [4] développé à l'université américaine d'Ohaio. Citons : *TcpTrace is a tool written by Shawn Ostermann at Ohio University, for analysis of TCP dump files. It can take as input the files produced by several popular packet-capture programs, including tcpdump, snoop, etherpeek, HP Net Metrix, and WinDump. tcptrace can produce several different types of output containing information on each connection seen, such as elapsed time, bytes and segments sent and recieved, retransmissions, round trip times, window advertisements, throughput, and more. It can also produce a number of graphs for further analysis.*

19.5 Commentaires et conclusions

Cette étude a démarré très récemment. A ce jour, nous nous concentrons sur l'application des méthodes d'échantillonage présentées dans les sections précédentes à de nombreuses collections de traces. Ensuite, nous évaluerons le comportement de la fonction objectif en fonction des différents paramètres, tout en essayant d'en déduire des corrélations et des régressions.

Chapitre 20

Le graphe du Web à l'aide de Traceroute

Introduction

The Internet is the world wide network that everyone use everyday to use applications like web browsing or e-mail from everywhere around the world. We may think that this communication "tool" designed by engineers in the seventies is now well known and that not secrets remain about its functioning and structure. Unfortunately, lots of open problems still exist about this fascinating network in order to understand how it works.

This work will concentrate first on one open problem that concern the Internet : the topology. We will perform a state of the art in order to see the existing means for acquiring the topology and the studies already done on it. Then, we will study in detail this topology on the data provided by an American project. And finally, still with the same data, we will try to characterize of paths and their connections with the topology in this network.

20.1 State of the art

The first thing to understand is why acquire and study the Internet topology is so important. Because we don't know how this world-wide network evolve and is exactly structure it is hard to simulate some phenomena or monitor the network. Indeed, it is hard for system designers to perform some tests of their applications or protocols before deployment. We will see that some topologies generators that can be used to fill this lack. Other points that could motivate this comprehension are network management and siting. The first concern all the monitoring tasks on the network and the second deal with the localization of some phenomena in the network. Note that it can be also use to detect the result of natural disasters, wars or DoS[1] attacks. In [18], authors were able to observe a loss of network connectivity in Yugoslavia during the NATO bombing in 1999. In this chapter, we

[1] Denial of Service

will first detail how the Internet is organized, then we will describe the way to acquire its topology and finally we will try to summarize works that have been done on the analysis of this topology.

20.1.1 Organization of the Internet

History and global structure of the Internet

The Internet is born because of the request in 1962 of the US Air force who has asked to a small research group to work on the creation of a nuclear attack resistant network. The concept of this network was based on a decentralize system in order to survive if one or several nodes were destroyed. The first version of the Internet called ARPANET was achieved in 1972. It was later partition in two parts, the military one and the public one which has become the Internet. The way it has grown up [19] explains many things of its actual structure. The Internet is now a huge network divided into domains that are connected together. Each domain is a collection of hosts interconnected via transmission and switching facilities. Domains that share their resources with other domains are called network service providers (or just providers). Domains that utilize other domain's resources are called network service subscribers (or just subscribers). A given domain may act as a provider and a subscriber simultaneously. In a more precise way, domains are called autonomous systems (AS) and are composed of a network or a group of networks. Each AS is under a common administration and has its own routing policy. The relations between AS follow commercial or political agreements, which specify in particular the way packets are transmitted between them. AS interconnects them at peering points. One peering point is Parix[2], an exchange point in Paris that connects among other things Tiscali (AS 3257), Telia (AS 1299), Easynet (AS 4589), NERIM (AS 13193) and PROXAD (AS 12322). Figure 20.1 shows a map of the AS topology. This map and lots of other beautiful maps are referenced by a research project called Cyber-Geography[3].

Addresses allocation

This part will detail the way IP[4] are managed on the Internet, it is important because we will deal in section 20.2 and section 20.3 with an IP graph.

Both IPv4 and IPv6 addresses are assigned in a delegated manner. Users are assigned IP addresses by Internet service providers (ISPs). ISPs obtain allocations of IP addresses from a local Internet registry (LIR) or a national Internet registry (NIR), or from their appropriate Regional Internet Registry (RIR) like :

- APNIC (Asia Pacific Network Information Center) for the asia and the pacific region
- ARIN (American Registry for Internet Numbers) for north America and sub-Sahara Africa
- LACNIC (Regional Latin-American and Caribbean IP Address Registry) for Latin America and some Caribbean Islands
- RIPE (Réseaux IP Européens) - Europe, the Middle East, Central Asia, and African countries located north of the equator

[2] http ://www.parix.net/
[3] http ://www.cybergeography.org/atlas/topology.html
[4] Internet Protocol

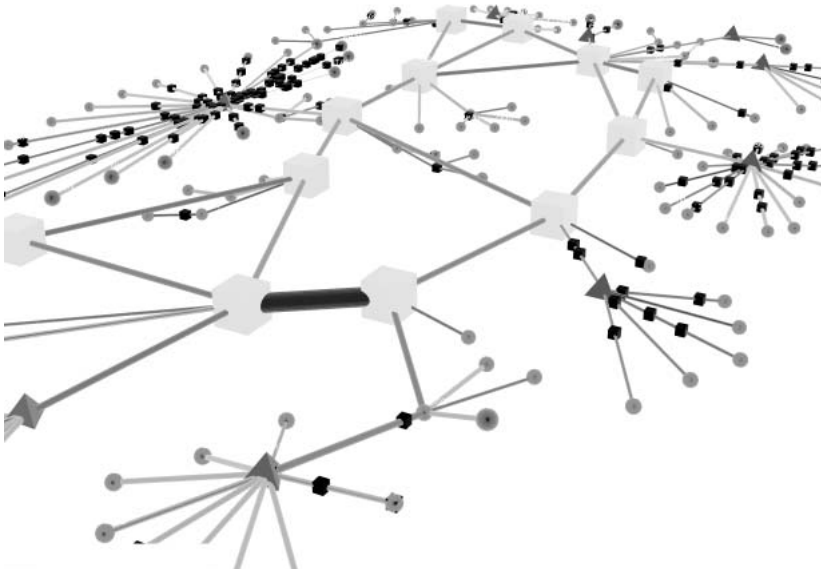


FIG. 20.1 – Interconnection in the Internet

The IANA[5]'s role is to allocate IP addresses from the pools of unallocated addresses to the RIRs according to their established needs.

We will just describe the organization of the IPv4 address space because we will not deal with the ipv6 protocol. IPv4 addresses are partitioned into 5 classes described in the RFC 1466 :

Class	Net mask	Address range
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
D	-	224.0.0.0 - 239.255.255.255
E	-	240.0.0.0 - 247.255.255.255

FIG. 20.2 – IPv4 space

This system is being replaced by the Classless Inter-Domain Routing (CIDR) (RFC 1517 to RFC 1520) system to face the inefficient use of the address space. The inefficiencies are mainly in the block assignments. For example, if an institution gets a class C IP range and only use a hundred of those, there are 154 unused and unavailable addresses. Another problem that motivates the CIDR system is the overtaxing of the routing tables. If an organization handles several consecutive class C IP ranges, routers have to store one route per IP range. With the CIDR system, an IP address range might look like this 212.80.191.0/24. This means that the first 24 bits in the address are used to identify the network while the remaining 8 bits are used to identify the host.

5

This approach is in use nowadays and is being encouraged. Implementation of the CIDR has been vital for the growth of the Internet, allowing more organizations and users to take advantage of this increasingly global networking and information resource. Those problems have been solved by the IPv6 standard but IPv4 is still dominant.

We have to note that in IPv4, some IP addresses are reserved for special purpose (see Figure 20.3).

Use	IP range
Loopback	127.0.0.1
Private networks (RFC 1918)	10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16
Reserved for future use	218.0.0.0/8 - 223.0.0.0/8 et 240.0.0.0/8 - 255.0.0.0/8
Reserved multicast	224.0.0.0/8 - 239.0.0.0/8
broadcast	255.255.255.255

FIG. 20.3 – Special addressed in IPv4

Routing

We will use active measurements (see section 20.1.2),so packets send to perform this task are under the law of routing policies that is one reason to explain a bit how information is route in the Internet. The other reason is the fact that we will study in section 20.3 paths in the Internet that are the consequences of these same routing policies.

The main function of routers is to make packets arrive at their destination. Each router has its policy to treat information. This policy can depends on commercial agreements, on the bandwidth of adjacent links, on congestion of certain parts of the networks, etc... The policy of a router is generally partition into two parts : a static one set by an administrator and a dynamic one handle by a routing protocol. There are two families of routing protocols use on the Internet depending on the kind of routers they are installed on. Indeed, there are routers called *border routers* that handle traffic between AS and routers called *intern routers* that handle traffic in a subnet as shown on 20.4.

The Border Gateway Protocol (BGP) is use to exchange routing information between AS on *border router* while an Interior Gateway Protocol (IGP) such as RIP[6], OSPF[7] or IS-IS[8] is use to exchange routing information within AS on *internal router*.

RIP sends routing-update messages at regular intervals and when the network topology changes. When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route. The metric value for the path is increased by 1, and the sender is indicated as the next hop. RIP routers maintain only the best route (the route with the lowest metric value) to a destination. This protocol just considers distance as the number of hops between two hosts but it does not take into account delay or bandwidth of links. This protocol is now outdated but it was one of the first.

[6]Routing Information Protocol
[7]Open Shortest Path First
[8]Intermediate System to Intermediate System

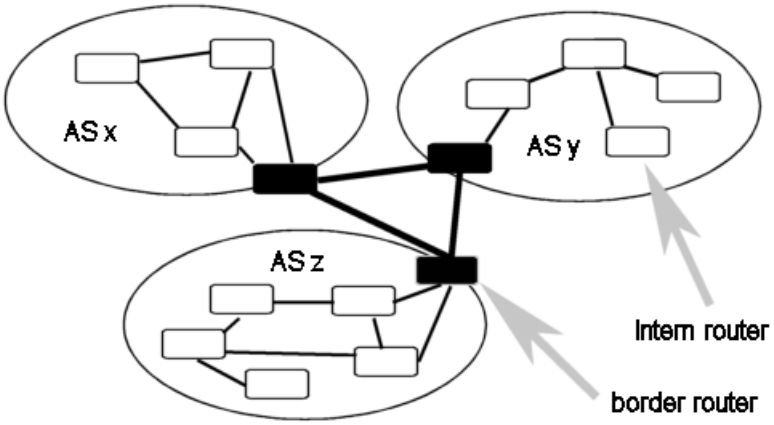


FIG. 20.4 – The different types of routers.

OSPF is more efficient than RIP and starts to replace it slowly. In opposite to the RIP protocol, routers don't propagate their distances in term of hops but the state of their adjacent links. Each router is able to compute the best path by knowing a map of the network. Furthermore, less information is transmitted over the network.

The main thing to remind here is that routing protocols are design to provide the best path to packets, meaning the shortest path inside AS and the faster or most efficient between them. Remind also that there is static information that can influence routing. In particular, at the AS level, commercial agreements called *peering sessions*.

Levels in the topology

Since we study the physical topology of this world-wide network, let's describe levels of abstraction that are commonly studied with the help of Figure 20.5.

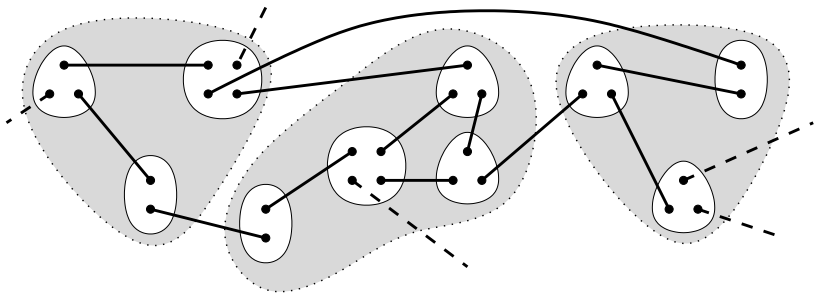


FIG. 20.5 – The three levels of the Internet architecture. Black dots represent interfaces, blank shapes stand for routers and shaded areas for AS. The (plain or dotted) lines correspond to physical links (always between two interfaces).

Indeed, the topology of Internet may be studied as a graph at three different levels. The first level concerns the way AS are interconnected. The second level is the interconnection of routers on the Internet. It represents cables, satellite or radio links, etc... This physical infrastructure is the one over which information is routed. Finally, at a kind of microscopic level, one may consider the IP graph, composed of the interfaces (of routers) which exchange information since each interface owns an IP on the Internet. All these three levels of the Internet topology are obviously strongly linked.

We will see that each level have been studied in section 20.1.2. In the experiments we have realized (see section 20.2 and 20.3), we will mainly deal with the interface level and a bit with the AS one.

20.1.2 Existing acquisition methods and research projects

Since each acquisition methods of the network topology acts usually at only one level this part will treat each level separately.

At the AS level

One of the most efforts to get snapshots of the Internet topology at this level is the Route Views project from the University of Oregon[9]. This project uses BGP information from many BGP routers to collect routes between AS. Because a BGP router sees a set of routes by receiving information from several other BGP routers. Historically, the Route Views project was originally backed by telecommunication operators in order to monitor the visibility of their prefixes and their AS space. But, Route Views data have served many other interesting projects. For example, it has been used for AS path visualization or to study IPv4 address space utilization [27]. All papers that deal with the topology of Internet at the AS level are based on those data. But note that this is not the only source of AS information. For instance, we can get AS information from RIPE.

At the IP level

This is at this level that the most number of techniques or tools exist. There are several methods to acquire the IP graph as [56] said :

- *SNMP*[10] : With this protocol, each router can provide a list of its neighbors. The main advantage of this method is that it generates a very low overhead on the network. Unfortunately, in most of case, SNMP is not available because routers don't support this feature or allow a restricted access to it.
- *Path forwarding* : This is the technique use by the *traceroute*[11] tool. The *traceroute* tool is based on the use of the TTL (Time To Live) of IP packets. Recall that the TTL is an IP header field that is designed to prevent packets from running in loops. Every router that handles a packet subtracts one from the packet's TTL. If the TTL reaches zero, the packet has expired and is discarded. *traceroute* depends on the common router practice of sending an ICMP[12]

[9] http ://www.routeviews.org
[10] Simple Netwok Management Protocol
[11] http ://www.traceroute.org
[12] Internet Control Message Protocol

time exceeded message [37], back to the sender when this occurs. *traceroute* first sends a packet to the destination with a TTL value 1 and iterate this operation with increasing values of the TTL. This gives the list of intermediate interfaces on the way to the destination. Some implementations of this tool can be found at http ://www.traceroute.org.

– *broadcast ping* : A ping is just an ICMP Echo request [37] to an host in order to know if it is alive or not. This simple tool can also be used to send a ping request to a broadcast address in order to get answer of all the alive machines in the subnet. There are some heuristics in [56] to discover subnet netmask and to get information about the topology.
– *DNS*[13] *transfer zone* : Most DNS servers respond to a Zone Transfer command by returning a list of every name in the domain.

[56] comes to the conclusion that the most efficient is the path forwarding method if it is done at a large scale. A lot of academic projects have dealt with this method :

– *Mercator*[14] : Mercator is a program from the SCAN project at the University of Southern California that uses *traceroute* like methods to infer the Internet topology. This program works on a single computer and does not need any input since it uses random address probing to explore the all IP address space. It also uses source-route capable routers wherever possible to enhance the quality of the resulting map. With this technique the sender of a packet can specify the route that a packet should take through the network. This method allows discovering cross links. Unfortunately, few routers support this feature. In [31], they found that approximately 8% of all Internet routers were capable of source-routing.
– *Nec* : Nec[15] is a software developed at the University Louis Pasteur[16] by Damien Magoni. It uses public *traceroute* servers to acquire quickly a short snapshot of the topology.
– *Skitter* : Skitter is a project from CAIDA [17]. It provides to the research community a free access to the data collected. This measurement platform is composed of about thirty servers which perform traceroutes toward thousands of destinations every day. Destinations are chosen to span all the CIDR prefixes in use. The CIDR prefixes in use can be found in BGP tables (see section 20.1.2).

To our knowledge the Skitter [29] project is at the state of the art since it is the most ambitious. Our data for the studies of the section 20.2 and section 20.3 come from Skitter. In section 20.2.1, more detailed about Skitter can be found.

At the router level

This level of study strongly depends on the interface level because the only mean to acquire it is to start from the interface topology and to infer it with the help of some heuristics. We forgot here the possibility to ask ISP for this information because of confidentiality reasons. For example the ones described in [31]. The main aim of this technique is to merge interfaces into routers. The trick is to discover if two interfaces belong to the same router or not. Here is a brief list of the heuristics developed in [31], let's analyze two of them :

[13] Domain Name Server
[14] http ://www.isi.edu/scan/mercator/mercator.html
[15] http ://clarinet.u-strasbg.fr/ magoni/nec/
[16] Strasbourg - France - http ://www-ulp.u-strasbg.fr
[17] the Cooperative Association for Internet Data Analysis - www.caida.org

– Usually routers have an interface called the loopback interface from which they use to respond to *traceroute* probes. So, to know if two interfaces belong to the same router, it is possible to send to each of them an UDP[18] packet to an unused port and to compare the address source of the ICMP port unreachable packets. If they are the same, they belong to the same router.
– Interfaces have usually DNS names and are often called like X.1.Y.domain.com and X.2.Y.domain.com for example in the case of two interfaces. So you can guess by looking at their name if they belong or not to the same router. This method is not so accurate because DNS content is often obsolete and this convention of naming is not so used nowadays.

Cross-level methods

Cross layering is possible but difficult and a lot of mistakes are possible. We have seen in the previous paragraph that from the IP graph you can infer the router one by using some heuristics. Some tools have implemented such techniques [28] but they don't obtain accurate results. Mercator and Nec have also this feature. In the same way, it is possible to go from the IP level to the AS graph with the help of whois databases like RADB [19] or a continental registry like RIPE [20] for Europe but it is a difficult task. The IP graph and the information of these databases have to be acquired at the same period and many problems appear like the fact that some IP don't belong to any AS and some routers have different interfaces belonging to different AS. Furthermore, note that obsolete data appear in whois databases.

20.1.3 Limits of acquisition methods

Dynamical aspects

As [53] said, the Internet is *an immense moving target* so measurements are hard to perform. In addition to the size of this large network, this phenomena is emphasized by its structure. Indeed, the Internet has not been built in a centralized way, it allows heterogeneous networks with very different administrative policies to interoperate. Consequently, researchers have resigned to believe that it is impossible to have a complete snapshot of the network at a certain moment. This is a quite fresh research subject and it does not exist a solid solution yet. Let's see some ways to analyze this dynamic and complex system :

– By collecting a huge amount of data on a certain period, the shortest as possible, like Skitter does. Then one can start to study those data and assume that they are representative of reality as we did in our study (see section 20.2). The entire problem here is to choose the period. Nobody has the answer. One can say that the graph is quite stable during a month so you can aggregate data collected during a month, another can argue that very important changes can occurs suddenly like new links or new routing policy that change radically the graph.
– By studying the dynamic of this graph and its evolution on a small data collection by making the assumption that they are representative. Nec is an example of a tool use to study the evolution of the core of the network for instance.

[18] User Datagram Protocol
[19] Routing Assets Database
[20] Réseaux IP Européens

Quality of the data

All the methods previously described in section 20.1.2 related to the acquisition of the Internet topology are disturbed by some distortion phenomena. At the AS level, many papers point out problems with instabilities of BGP [35], at the Interface level the path forwarding method suffers also of various issues and finally at the router level the heuristics used don't work in all the case. Because all the data studied in this thesis are the result of millions of *traceroute* executions, let's concentrate on problems inferred by the path forwarding method.

As seen in section 20.1.2, the path forwarding method triggers ICMP errors along a path to discover it by playing[62] with the TTL IP field. Unfortunately, many tricky problems go against this like :

- The desire to keep confidential the topology or other reasons that motivate administrators to configure routers to not answer to *traceroute* probes.
- Route changes or parallel load balanced links that make packets of a *traceroute* probe take different paths. Leading to the creation of bad links in the IP graph as shown in Figure 20.6.

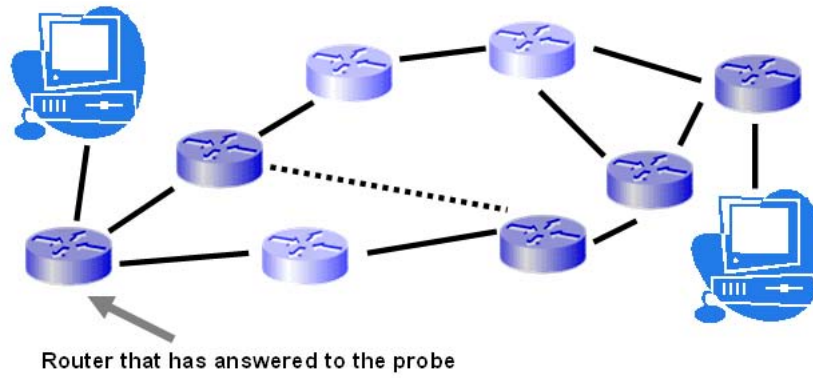


FIG. 20.6 – Fake links that can be seen with *traceroute*.

- Intrusion detection systems triggered alarms because of the traffic generated. Indeed, the traffic generated may looks like to port scan probes if UDP is use. The solution is to use only ICMP.
- Bad answers to *traceroute* probes are sent. Special IPs appear in the traces. Due to the fact that :
  - Sometimes NAT[21] is not well performed for the sender address in IP packets. As a consequence, packets circulate in the Internet with a private IP for the sender. Packets like this generally reach their destination because routers don't take care about the sender address.
  - Some administrators configure routers to reply with their own private IP because they want it to be visible as a hop in a route but not to be reachable from the outside of the subnet.
- The fact that routers contain bugs. Some examples of *traceroute* related problems can be found in appendix 20.4.

Finally, other facts that have an impact on the quality of data but at a more structural level because this method study the topology at the IP level :

[21] Network Address Translation

– Figure 20.7 shows that the logical topology at the IP level is a bit different than the topology at the router level. The paper of Renata Teixeira[55] shows this.

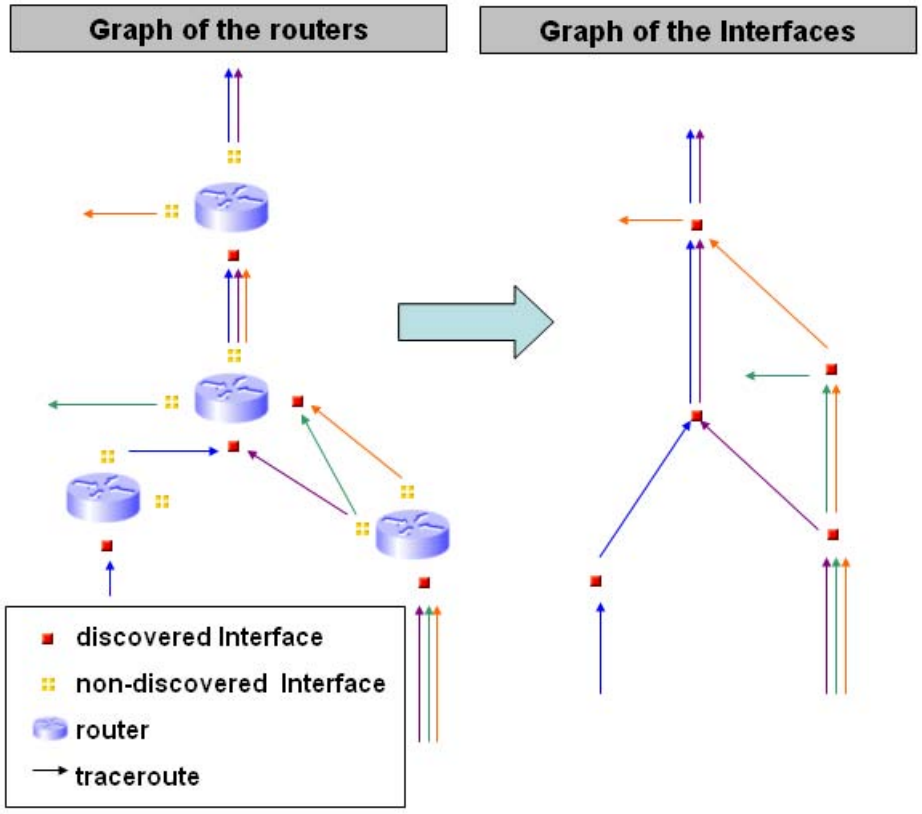


FIG. 20.7 – The difference between the physical topology and the logical one at the IP level.

– Packets sometimes enter ATM[22] or MPLS[23] networks and are encapsulated. It does not allow the path forwarding method to discover intermediate hops. It seems that it is a reason for the existence of nodes with very high degrees in the IP graph (see chapter 20.2).

20.1.4 Existing analysis of the Internet topology

Properties already observed

Since this topology is a graph, all the analysis made on this graph deal with graph theory.

The first thing the community has pointed out is the existence of power laws between properties of this graph. This was introduced by the Faloutsos brothers in [25] and impressed the network

[22] Asynchronous Transfer Mode
[23] Multiprotocol Label Switching

community. The first power-law relationship they found is between the out-degree of a node v (d_v) and its rank (its index in order of decreasing outdegree) r_v : $d_v \alpha$[24] r_v^R. It reflects the way domain connects. The second power-law relationship is between the outdegree d and its frequency f_d : $f_d \alpha d^O$. This shown that the degree distribution on the Internet is not arbitrary, it also shows that there is a huge number of nodes with a very small outdegree and some nodes with a high degree. Finally, the third one they found links the eigen value λ_i and the order i : $\lambda_i \alpha i^k$. This last law allows to distinguish the differences between graph families.

There are several remarks on this. First, the computation of the exponent is subject to large error [17] : small variation on the distribution has a huge impact on the exponent computation. This is why focusing on the exact measured exponent does not seem to be a reasonable aim for model designers. Second, this power-law seems to exist because of the way the graph has been explored [45].

The clustering coefficient has also been studied. The clustering coefficient of a node gives information about connectivity between neighbors of a node. It is given by the following formula :

$$C(u) = \frac{2 * E_u}{k_u * (k_u - 1)}.$$

Here, k_u is the number of neighbors of the node u and E_u is the number of existing links between the neighbors of u. If this coefficient is equal to 1, it means that all possible links between neighbors of a node exist. Meaning that the fewer neighbors are connected together, the less is the clustering coefficient. Some studies like [39] have shown that at the AS level the average of the clustering coefficient is quite «high» (it is at least higher than for traditional random graphs).

This phenomena and the fact that there is a degree distribution in power-law may be an explanation for the small-world phenomena observed on the graph of Internet [39]. Intuitively, having a graph that fits a small-world phenomena means that the length of a path between two nodes chosen randomly is quite short, in the same order of $log(N)$ if the graph has N vertices. This is the case in many complex networks [40] like social networks [38] or in the human language [36]. Small-world graphs are usually obtained by rewiring edges from a regular graph. [61] provides a small JAVA[25] applet that illustrates this phenomena.

In order to understand why this graph is very specific, let's compare it to a random graph :

Property	Random graph	the Internet graph
Degree distribution	Poisson	Power law
Clustering	low	high

FIG. 20.8 – Comparison of the graph of the Internet and a random graph

Existing models of generators

There are several topology generators available to the research community. Some of them mainly aim to generate random topologies, others aim to imitate the hierarchical properties of the Internet,

[24] "'Proportional to"'

[25] http ://java.sun.com

others aim to reproduce degree-related properties of the Internet and some use some engineering or economic related constraints.

Random graph generators Erdos-Renyi [24] developed one of the first topology generators. It creates a random graph by assigning a uniform probability for creating a link between any pair of nodes. Later, Waxman [65] improved the Erdos-Renyi random graph model by including network-specific characteristics such as placing the nodes on a plane and using a probability function to interconnect two nodes which is dependant on the distance that separates them in the plane.

Structural generators In this category of generators, models focus on reproducing the hierarchical structure of the topology of the Internet.

The Transit-Stub [15] model is one the most famous. The background idea is that the Internet can be viewed as a collection of interconnected routing domains. Each routing domain in the Internet can be classified as either as stub domain or as transit domain. A domain is a stub domain if the path connecting any two nodes u and v goes through that domain only if either u or v is in that domain. A transit domain does not have this restriction. So, in the Transit-Stub domain, a connected random graph is first generated using the Waxman method. Then, each node in the graph represents an entire Transit domain. Each transit domain node is expanded (see Figure 20.9) to form another connected random graph, representing the backbone topology of that transit domain. Next, for each node in each transit domain, a number of random graphs are generated representing Stub domains that are attached to that node. Finally, some extra connectivity is added, in the form of *back-door* links between pairs of nodes, where a pair of nodes consists of a node from a transit domain and another from a stub domain, or one node from each of two different stub domains.

Another generator that implements models trying to imitate the structure of the Internet is Tiers [23]. The generation model of Tiers is based on a three-level hierarchy aimed at reproducing the differentiation between Wide-Area, Metropolitan-Area and Local-Area networks comprising the Internet.

Degree based generators PLRG[26] [58] is a generator which tries to reproduce the connectivity properties of Internet topologies as reported in [25]. This generator initially assigns node degrees from a power-law distribution and then proceeds to interconnect them using different rules.

The Albert and Barabasi model This model [10] builds a graph by adding nodes one by one : when inserted, the node is linked to a fixed number of already present nodes chosen randomly, proportionally to their current degrees. This is the *rich get richer* or the *preferential attachment* principle.

Others models that take into account the correlation between the connectivity of the network and the density of the population have been established in [44].

Here are two sofwares that implements the previous algotithms :

– *GT-ITM* : the Georgia Tech Internetwork Topology Models generator from the Georgia institute of technology [60].

[26] Power-Law Random Graph generator

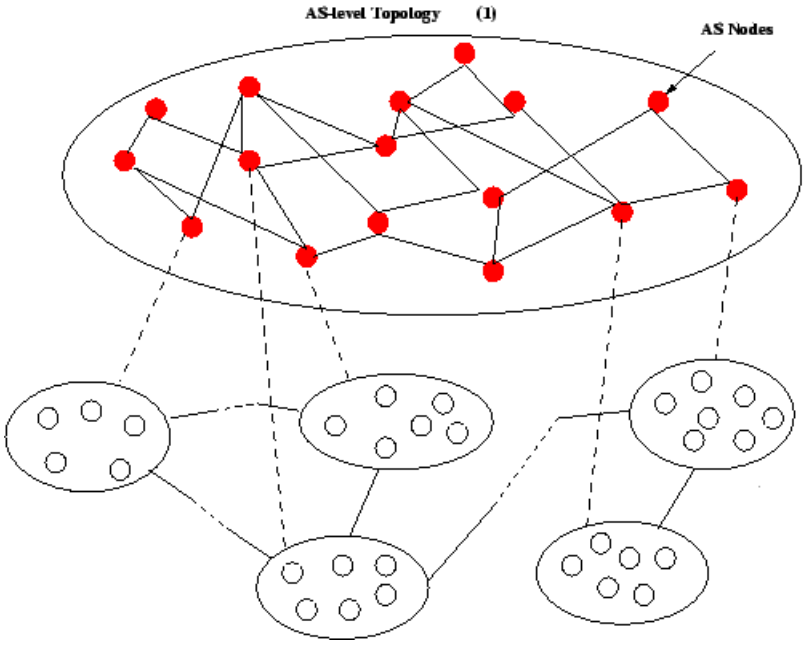


FIG. 20.9 – The Transit-Stub generator.

– *BRITE* : the Boston University Representative Internet Topology gEnerator [30].

The main question that remains open is how to validate a model of the Internet ? One way may be to ask why are we looking for models of the Internet ? The answer to this question will certainly help to determine proper parameters to measure.

20.2 Analysis of the Skitter graph

In this chapter, we will detail the observations we made on the data provided by the Skitter project of the CAIDA group after a presentation of the methodology adopted.

20.2.1 Methodology

We have seen in section 20.1.3 that there are two methods to analyze data collected with the path forwarding method. A quasi-static one consists to consider data on a short period and a dynamic one considering data on a large period. We have concentrated ourselves with the static method by taking data collecting by Skitter during a day. We will describe in this part how Skitter works and what we can expect of its data and we will give information about the tools we have used to analyze these data. Finally, we will end with some explanations about implementation of some algorithms.

Skitter

We have seen in section 20.1.2 that Skitter performs millions of traceroutes per day from their servers. CAIDA currently maintains thirties Skitter hosts all over the world. However, not all Skitter monitors are running the full destination set at all times. So we have chosen a day that contains data for 23 servers. Each server performs the measurements described in 20.2.1 by targeting a destinations defined in 20.2.1 using the path forwarding method with ICMP[27].

Servers' location 20.10 and 20.11 show the location of the servers.

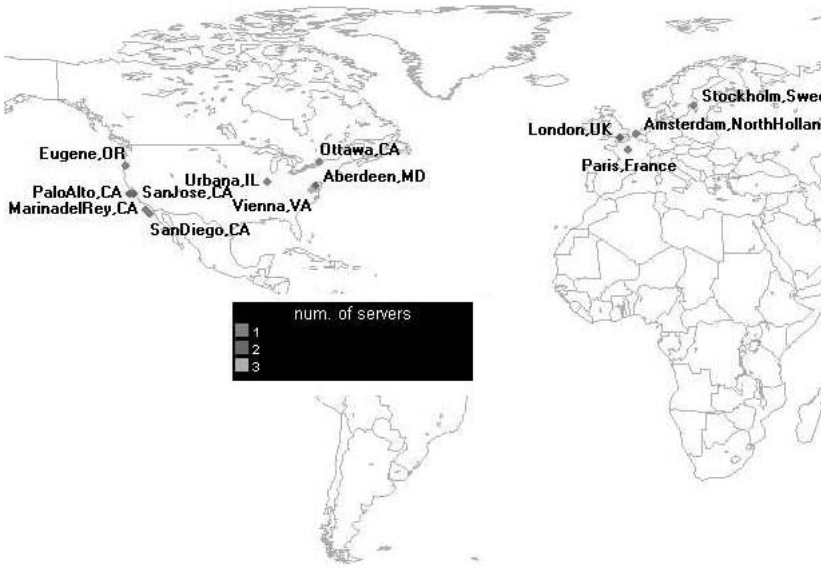


FIG. 20.10 – Positions of Skitter servers

Measurements Here is a list given by [29] of the measurements done by Skitter :

- *Forward IP Paths* : Skitter records each hop from a source to many destinations. By incrementing the "time to live" (TTL) of each IP packet header and recording replies from each router (or hop) leading to the destination host.
- *Round Trip Time* : Skitter collects round trip time (RTT) along with path (hop) data. skitter uses ICMP echo requests as probes to a list of IP destinations.
- *Persistent Routing Changes* : Skitter data can provide indications of low-frequency persistent routing changes. Correlations between RTT and time of day may reveal a change in either forward or reverse path routing.
- *Visualization of the Network Connectivity* : By probing the paths to many destinations IP addresses spread throughout the IPv4 address space, skitter data can be used to visualize the directed graph from a source to much of the Internet

[27] http ://www.caida.org/tools/measurement/skitter/packets/

Name	Town		Name	Town	
a-root	Herdon,VA	US	apan-jp	Tokyo	Jp
b-root	Marina del Re	US	cdg-rssac	Paris	Fr
champagne	Urbana,IL	US	d-root	College Park,MD	US
e-root	Moffett Field,CA	US	f-root	Palo Alto,CA	US
g-root	Vienna,CA	US	h-root	Aberdeen,MD	US
i-root	Stockholm	S	iad	Washington,DC	US
k-peer	Amsterdam	Nl	k-root	London	UK
kaist	Taejon	Kr	lhr	London	UK
m-root	Tokyo	Jp	mwest	San Jose,CA	US
nrt	Tokyo	Jp	riesling	SanDiego, CA	US
sjc	San Jose, CA	US	uoregon	Eugene, OR	US
yto	Ottawa	Ca			

FIG. 20.11 – Lists of Skitter' servers use for our studies.

Targets The pool of replying destinations is constantly changing and decreasing due to firewalls, changing IP addresses and other reasons. They need to refresh the destinations lists every 8 to 12 months. Currently, CAIDA use two kinds of destination lists[28] : DNS list and IPv4 list. The two families of list in 20.12 have two different goals :

- *IPv4 list* : Provide a representative coverage of the routable IPv4 space for topology measurements. Ideally, this list should have an IP address in each populated /24 prefix of the global Internet space.
- *DNS list* : Provide a representative coverage of clients querying DNS root name servers.

Family	Date	Size	Monitors
IPv4 list	02/25/2003	147,016	champagne, kaist, riesling
IPv4 list	02/25/2003	365,605	apan-jp, iad, lhr, nrt, sjc, yto
IPv4 list	02/25/2003	814,356	mwest, uoregon
DNS Clients list	01/13/2003	147,943	a-root, b-root, d-root, e-root, f-root, g-root, h-root, i-root, k-root, m-root, k-peer, cdg-rssac

FIG. 20.12 – List of destinations provided by Skitter (fall 2003).

Format of the traces Traces available to the research community provided by Skitter are stored in a binary format called Arts++[29]. This format defined by the CAIDA group is dedicated for the storage of traces from active measurements on networks. CAIDA has provided a C++ class library to access data in this format and an example [30] that use it. This small example is entirely sufficient to

[28] http ://www.caida.org/analysis/topology/macroscopic/list.html
[29] http ://www.caida.org/tools/utilities/arts/
[30] http ://www.caida.org/tools/measurement/skitter/sample_code/

extract basic information (see section 20.5) collected in the traceroute fashion by Skitter. Please note that Skitter affects a key for each trace which determines its quality. Here are its possible values :

- *N* : no reply was received from the destination although a partial path may have been recorded. The RTT has no meaning in this case.
- *I* : Skitter got a reply from the destination, but did not receive a reply from every intermediate hop on the path. The RTT to the destination is valid. *I* stands for *incomplete*.
- *C* : The destination and all intermediate hops in the path all replied. The RTT to the destination is valid. *C* stands for *complete*.

Activity The activity of Skitter is irregular. It can append that a monitor goes down for a month and then goes up back again. For our studies we have chosen a day during which 23 monitors have worked. The $2th$ of July 2003. All monitors of 20.11 have performed about 13 millions of traceroutes where 8 millions are of type *C* toward about 600 000 destinations.

We have done some reverse engineering on the data in order to know how skitter proceed. We will not give all the details of this study in this report but just an example for the monitor called *a-root*.

Figure 20.13 shows the number of traceroutes by type this server have done. For our study we have only keep traces of type *C* because it was simpler to deal with.

Nb total of traceroutes 478983	
where :	
Type N	134049
Type I	66325
Type C	278609
Type C & q	158

FIG. 20.13 – Consistency of the data collected by *a-root*.

Monitors generally target a same destination several times in a day. Figure 20.14 shows that they usually perform 3 traceroutes toward a destination. This figure also shows the number of different routes discovered when there are several routes in the data set between a Skitter server and a destination. We can observe that in many case, there are more than one route toward a destination discovered per day.

Tools used

BOOST The BOOST[31] library is a set of portable C++ libraries. The library we have used is the BOOST Graph Library (BGL) developed by Jeremy Siek. It provides generic components and algorithms for manipulations of graphs and offers genericity like the STL[32] and optimizations. Algorithms and data structures are generic, meaning that they can be adapted to any kind of problem. Here are some algorithms from graph theory that this library implements :

[31] www.boost.org
[32] Standard Template Library

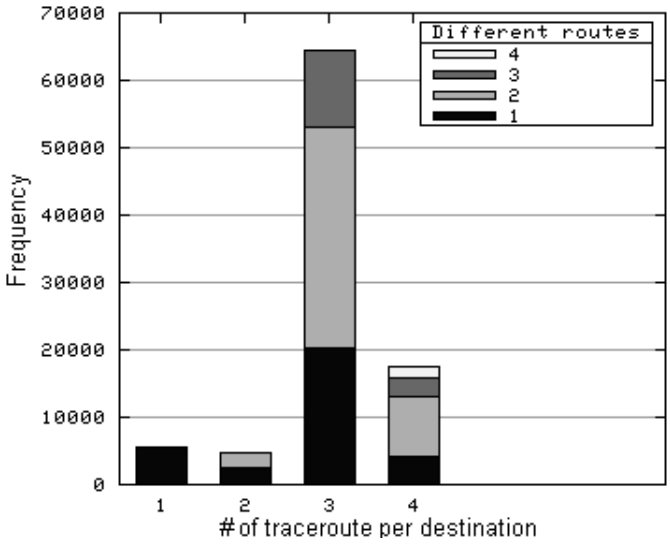


FIG. 20.14 – Frequency and stability of traceroutes done by *a-root*.

- Breadth first search,depth first search and uniform cost search
- Dijkstra's shortest paths, Bellman-Ford and Johnson's shortest paths
- Kruskal's Minimum Spanning Tree, Prim's Minimum Spanning Tree
- Connected Components, Strongly Connected Components
- Topological Sort

DOT The DOT[33] format is a very powerful way to represent graphs because it is simple and it has lots of extend that can handles layout related information. Documentation on the language is maintained by AT&T within the framework of the project GraphViz[34] which provides a set of tools for graph visualization (see Figure 20.15). We have chosen this format because of its simplicity and also because the BOOST Graph Library has an interface that allows to load graph store in DOT.

Name	**Function**
dot	makes hierarchical layouts of directed graphs
neato	makes "spring" model layouts of undirected graphs
dotty & tcldot	two customizable interfaces
libgraph	the base library for graph tools

FIG. 20.15 – Content of the Graphviz package.

Here is an Hello World example in DOT :

[33] http ://www.research.att.com/ erg/graphviz/info/lang.html
[34] http ://www.research.att.com/sw/tools/graphviz/

```
digraph g{
    "202.188.126.141"->"210.187.15.253";
    "202.188.126.141"->"202.188.128.144"
    "202.188.128.144"->"210.187.15.253";
}
```

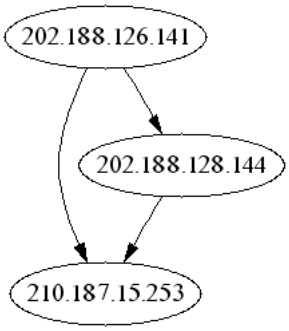


FIG. 20.16 – Example of a graph drawn with Graphviz.

Shell Scripts In addition to the use of C++ programs we have use shell scripts because it is often more efficient to use small utilities already developed and tested. Existing tools are often better than the ones you intend to make. So, we have chosen to implement our tools as a mix of gnu tools like *sort*, *awk* or *gnuplot* and C++ programs embedded in Tcsh[35] shell scripts.

Implementations

Creation of the Skitter graph The goal here was to take the entire traces offer by Skitter and to build a graph in DOT. The question has been regarded before our study. An algorithm developed at LIP6 before this study in C++ was available. Taking all traceroutes and adding links to a graph structure one by one. Unfortunately, this algorithm was too slow, it would have taken more than one month to obtain the graph. So, we have come with a very simple and efficient solution that builds the graph in DOT in one hour :

1. For each trace (with *awk*)
   – print each link into a file
2. sort uniquely this file using the *sort*
3. make some post processing and arrangement to fit the DOT format

This solution which consists to manipulate text files appeared to be faster that the previous approach.

We have seen in section 20.1.3 that this graph contains special IPs that should not be seen on the Internet. We need to remove them because they create fake nodes with high degree. This operation is a part of the post processing.

Manipulations of the Skitter graph All operations that only concern the Skitter graph have been realized within the C++ programs such as :

– degree distribution

[35] http ://www.tcsh.org

– clustering distribution
– shortest paths computation

The graph is first loaded with the ReadGraphviz function in a directed or in an undirected fashion. Then, algorithms are developed simply with BOOST functionalities and data structures. In term of performance, our program needs about 300 Mo of RAM memory to use the Skitter graph. It is able to load it and to run algorithms in few minutes.

Manipulations of the Skitter data Shell scripts are used to perform manipulations of the Skitters data because it allows calling all the efficient tools that already exist and that can solve our problems. For some of our studies, the algorithms use both shell scripts and the C++ program.

A complete list of the functionalities developed is available in section 20.6.

20.2.2 Observations

Data consistency

The Skitter graph freshly build from the set of traceroutes provided by Skitter contains special IPs (see section 20.1.1) that should not appear in this graph as shown in section 20.1.3. Indeed, there are

– 12077 (1.3%) IPs are private ones.
– 42390 (3.2%) links in the graph imply at least one private IP.
– 237 links imply the 127.0.0.0/8 addresses.
– 97 links imply the 0.0.0.0 address.
– 210 links imply broadcast IPs.

After purification, the Skitter graph is composed of :

– Vertices : 885 435
– Edges : 1 259 039

All the following studies have been made on this graph.

Degree distribution

As we analyze a graph, a common and useful distribution which characterize it is its degree distribution. We have seen in section 20.1.4, that on the graph of the Internet, this distribution follows a power-law $f_d \alpha\ d^O$ where f_d is the frequency of the degree d and O a constant around -2. This observation made in 1999 by the Faloutsos brothers is shown on Figure 20.17 on a directed IP graph.

We have recomputed this statistic on Skitter data considering our graph in a directed fashion and also in an undirected fashion. The question of the orientation of the graph will be discuss in section 20.2.2. The exponents of the power-law found are : -1.9 for the study on the directed graph and -2.2 for the study on the undirected graph. Our distributions show a distribution with a heavy tailed. This was not pointed out by [17].

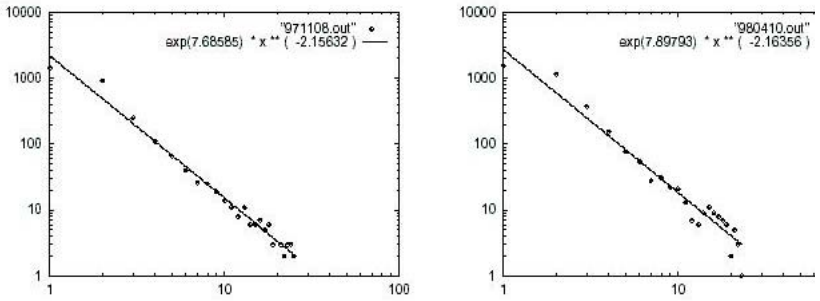


FIG. 20.17 – Two observations of the degree distribution done by the faloutso brothers in [25].

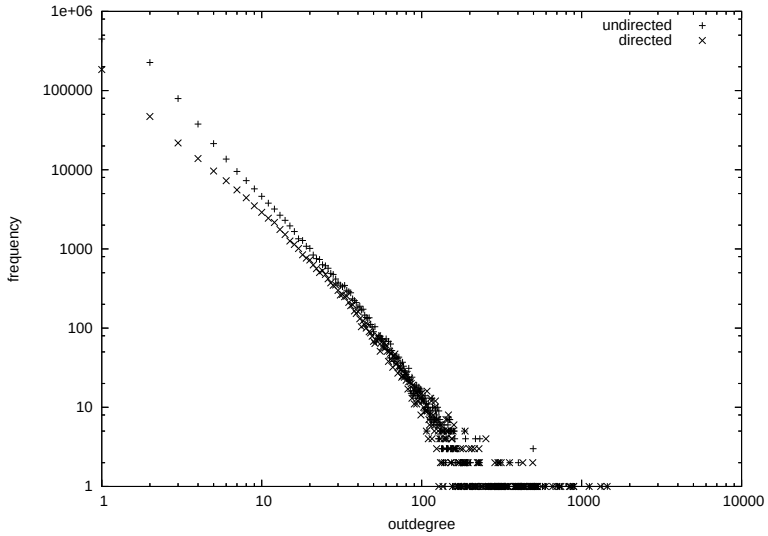


FIG. 20.18 – Out-degree distribution of the Skitter graph.

Clustering coefficient

The clustering coefficient is also a tool generally use to describe graph. The formula is given in section 20.1.4. The value of the clustering itself is not sufficient to compare two graphs because it is influenced by the number of vertices and the number links of the graphs. Therefore, it is useful to compare two graph of the same size. That is why in this part we have computed the average clustering coefficient for the Skitter graph and we have compared it to the one for a random graph of the same size generated with the Erdos-Renyi model (see section 20.1.4). We have found a value of 0.0347379 for the Skitter graph and $1.29822e-06$ for the random graph. This experiment has shown that the Skitter graph is highly clustered. Meaning that there is a strong connectivity in this graph.

Length of paths

The length of shortest paths between nodes in a graph is also an important parameter. We have again made this study both in a directed way and in undirected. The source of paths correspond here to the Skitter' monitors and the destinations to destinations of these servers because we want to know what is the typical distance between the extremities of the graph. Figure 20.19 shows us that shortest paths are quite small. We meet again here the small-world behavior described in section 20.1.4. Note that the difference between the distribution done in directed and in undirected is not so important. We will discuss of the orientation in section 20.2.2. Lengths of shortest paths in the directed graph are obviously longer in average than in the undirected graph because some detours are taken.

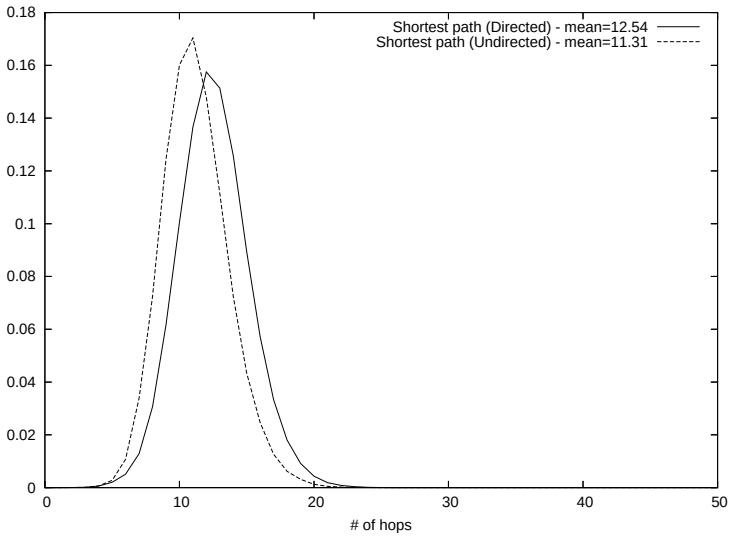


FIG. 20.19 – Probability distribution function of the shortest path lengths.

Core of the Internet

Here is just a trial to analyze the core of the Internet. Our hope was to extract the core of Internet by different manners and to analyze them by doing a degree distribution. We intend to not find a power law. Figure 20.20 shows the results for the different methods explained in the legend. You can see that the power-law stands again.

Orientation of the graph

This subject has been the center of many discussions during this work and is not simple. Shall we consider our graph as oriented or not ? Shall we perform our analysis or experimentations on a directed graph or not ? We have begun to look at works already done [13] and we have seen that they always consider the Skitter graph or the IP graph as directed. We come to the conclusion that since it is the IP graph we are dealing with, the only possibility is to take it as directed. But we

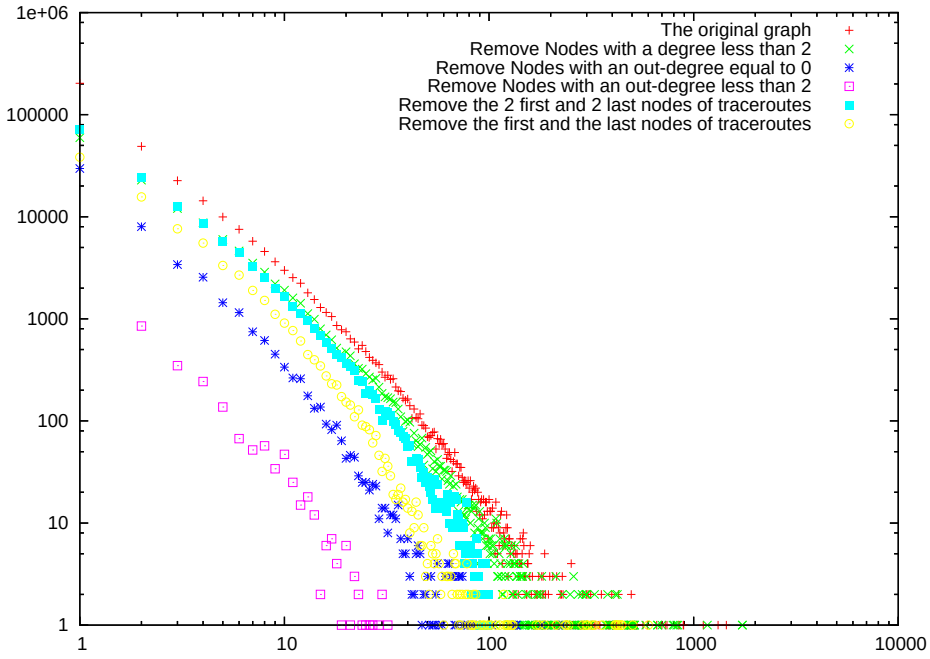


FIG. 20.20 – Degree distributions on the different versions of the core of Internet extracted.

can make some assumptions depending what we are studying that allow to consider this graph as undirected. We will do that for our study in chapter 20.3 on paths in the Internet Indeed, we can assume that if a link between two interfaces exists in a way it should exists in the way back but it has not been discovered. So, it is not crazy to consider the Skitter graph as undirected since properties like distances in term of number of hops are conserved.

20.3 What is a route on the Internet ?

20.3.1 Observations

Let's recall that we are working on the Skitter graph which has been construct from the merging of millions of traces from traceroute. We have performed some statistical analysis to describe the Skitter graph in chapter 20.2. Furthermore, we have performed other statistical analysis to described routes in this graph.

Length of routes

We plot in Figure 20.21 the length distributions of both routes and shortest paths, together with the distribution of their difference, that we now call δ. These curves have been computed both on the directed and undirected graphs.

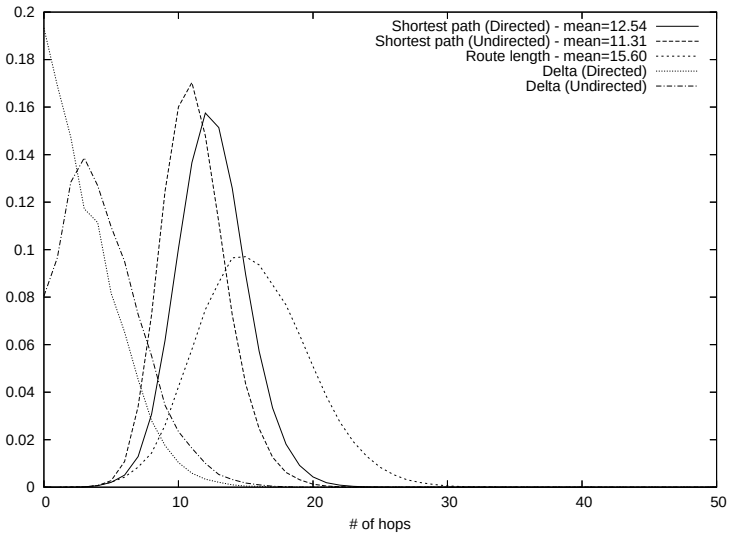


FIG. 20.21 – Length distribution of routes, shortest paths and their difference (δ) both in the directed and the undirected fashion.

Through the δ distribution, Figure 20.21 shows that routes have not always the same length the shortest paths. We have computed the shortest path and the δ distributions both in the directed and the undirected fashion because the two manners are interesting. So, experimentally, we bring the evidence that routing protocols does their best effort to choose routes closer as possible to shortest paths but fail at least in more than 80 percent of the cases. This value is given by the δ curve done with on a directed version of the Skitter graph, it certainly gives a lower bound for this percentage. According to the discussion in section 20.2.2 where we come to the conclusion that shortest paths in the undirected graph are more realistic than in the directed one, it is also interesting to look at the undirected version of the δ distribution because it may give a value close to the real value of the previous percentage. Meaning that shortest path are in 95% of the cases longer than in shortest path. Furthermore, because lots of edges are certainly missing in our graph, this percentage may be higher in reality. This curve shows that routes are generally 3 hops longer than shortest paths.

Intuitively, the value of δ is not independent of the length of the shortest path length because the values of δ for shortest paths of 5 hops may be shorter than the values of δ for shortest paths of 15 hops. Actually, as the Figure 20.22 shows, we have observed that the average of δ arraises slightly with the shortest paths lengths . This figure was plot for shortest path lengths between 9 and 16 which represents more than 85 percents of the cases. We have also plotted quantiles in order to back this result, they show that values are mainly close to the mean.

The analysis of routes in terms of shortest paths in the network assumes a global knowledge of the network. In the *real* network, on the contrary, a route is discovered step after step, each router taking is own decision, without a global knowledge of the whole Internet topology. This may reveal special local properties of routes, like correlations between nodes along a route, we have studies in next sections.

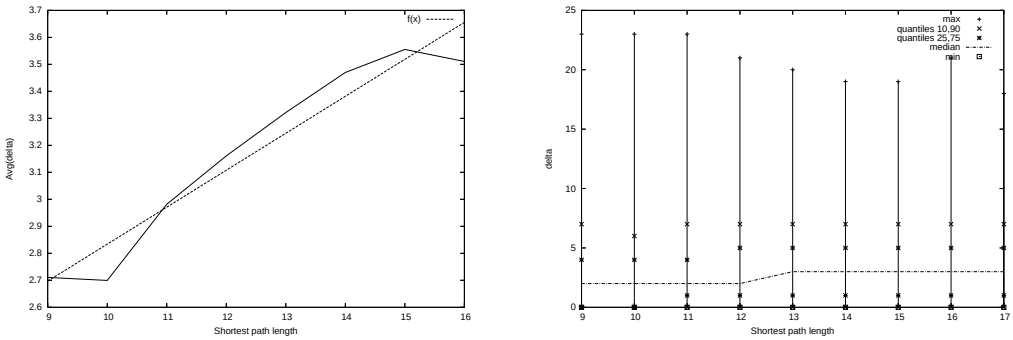


FIG. 20.22 – δ function of shortest path lengths.

Degree evolution along routes

In the willing to extract local properties of routes we have come to the following question : how does the degree evolve along a route ? To answer this question, we plot the evolution of the degree along routes in Figure 20.24. To this question, before doing the experiments, we came with the expectations of Figure 20.23.

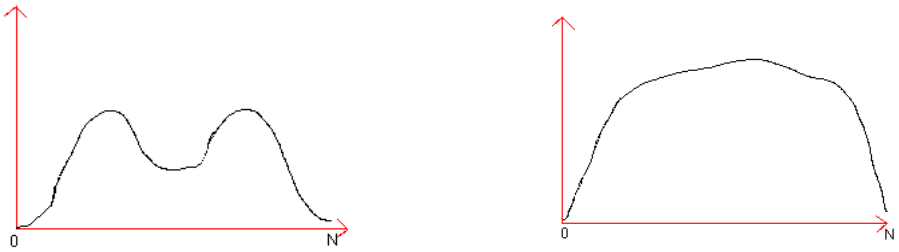


FIG. 20.23 – Our expectations before doing experiments for the degree evolution along traceroutes of length N+1

The curve in Figure 20.24 presents the experiments we have made for routes that have a length of 15. This example has been chosen because 15 is one of the most common route length in our data set and because the shape and the values shown in this Figure are very similar to the ones observed for length between 9 and 20. The median value of the out-degree is surprisingly constant in the core, around 10. Theses results break the intuitive idea that nodes' out-degree increase and decrease with a peak at the middle of the route. It also appears that nodes close to the border but not at the extremities of routes seem to have slightly a higher out-degree.

In Figure 20.25, we plot the evolution of the average degree which is mainly disturbed by sparse values.

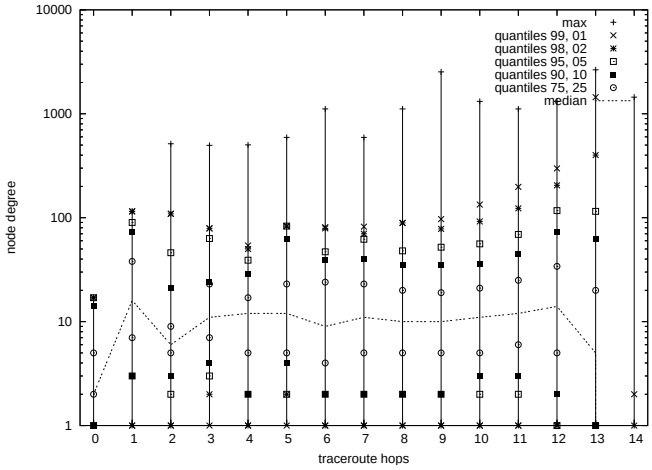


FIG. 20.24 – Evolution of the degree along routes.

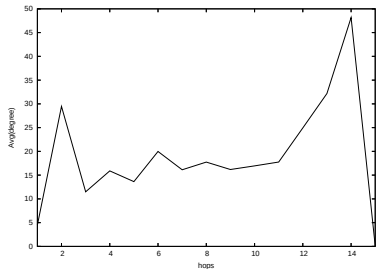


FIG. 20.25 – Evolution of the average degree along routes.

Local properties of routes

We try to answer here the following question : do routers choose their best-connected neighbor as the next step on the route ? To answer this, we have for each nodes with an out-degree N classified decreasingly its out-edges function of the targeted node' out-degree in order to attribute a rank to each of them. Thus, the neighbor that has a rank equal to 1 has the highest out-degree. Then, we have tried to see if routes usually follow low rank edges rather than high rank edges. In the case of several neighbors have the same out-degree, we consider that they have the same importance.

We finally observed in Figure 20.26 that there is a slight trend for routes to use links that leads to the neighbor that have the highest degree. Therefore, from an out-degree of 8, the curve still is a bit different. Routes follow preferably edges that lead to a neighbor that has one of an high out-degree compare to the others which is not always the highest out-degree neighbor as shown in Figure 20.27.

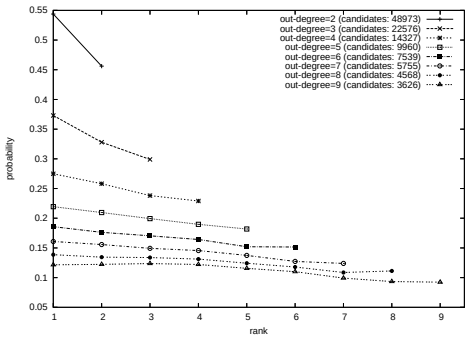


FIG. 20.26 – Choice of each router with respect to its neighbor's degrees.

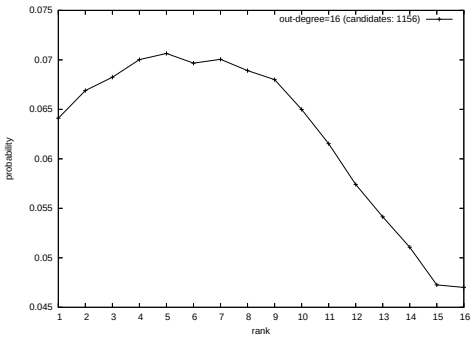


FIG. 20.27 – Choice at router having an out-degree of 8.

Properties of links

We have seen in section 20.3.1 that routes are in most of the cases longer than shortest paths and we have tried to reveal some characteristics that could explained this phenomena. In addition, we compute statistics shown on Figure 20.28 to see in which proportion links in routes go away from the server, approach toward the server or remain at a constant distance from the server. This experience has been made on a sample of routes randomly chosen from the Skitter data set.

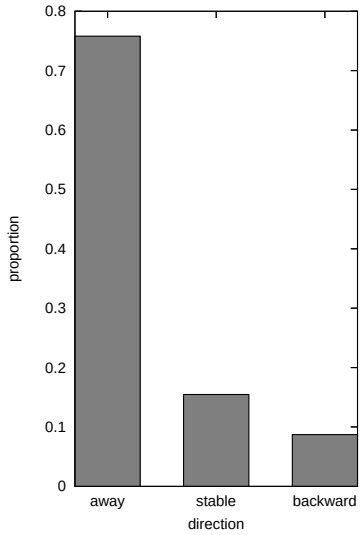


FIG. 20.28 – Direction of links in a sample of routes.

Statistics on Figure 20.28 show that a non negligible proportion of links in routes go backward, meaning that they go back to the server, and that a quite important part of them remain stable. This is an explanation of the fact that routes are longer than shortest paths. Routes may certainly follow some detours. This phenomena has been pointed out at the AS level in [59].

Figure 20.29 shows the evolution of these proportion function of the distance from the server in routes of length 15. We can observed that it is in the middle of routes that there are more floating behaviors. We can suppose that middle of routes correspond to the core of the network. This shows that it is in the core of the network that detours appear.

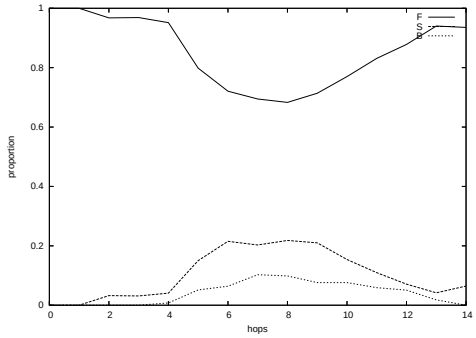


FIG. 20.29 – Direction of links in a sample of routes of length 15.

Note that going away from the server is not the same as going toward the destination. Unfortunately, it was not possible to study the proportion of links that go toward destinations because of computation problems. Indeed, in the Skitter data set, from the point of view of a server, you can treat more traceroutes (thus links) at a time than from the point of view of a destination.

Dynamics of routes

We just have made preliminary studies. We realized for each Skitter' server statistics like the ones of Figure 20.14 showing that if several traceroutes have been perform between a server and a destination, in most of case there are more than one route observed. We did not go deeper in this study and the way these routes are different have to be characterized. Furthermore, studies concerning length evolution of routes are also necessary.

20.3.2 How to model routes ?

On the light of statistics pointed out in the previous section, we have tried to define models to design paths on the Internet. Until now, paths in this world wide network are often modeled by shortest paths because no precise studies have described what a route looks like on the Internet.

One possibility could have been to complicate a lot our models but we have taken the position to make it the simple as possible in order to provide efficient solutions. Let's described the three methods we have designed to build fake routes and the results of simulations.

Methods

Random perturbation based This method tends to generate routes with a random based solution. The shortest path is compute between the starting point and the destination. Then, at each hop we

set a probability that a perturbation appears. If it appears, a routes going through a neighbor of the current node not in the shortest path is searched. The algorithm also avoids two-loops.

Length based This one tried to concentrate on the length of the path given information plotted in Figure 20.21. The shortest path is compute between the starting point and the destination. Then, a δ is randomly chosen by following the distribution of Figure 20.21 and is added to the shortest path length. A path of the previous length is searched.

Local properties based Finally, the last one is a bit more complicated. Assume that for all nodes, predecessors are their highest degree neighbor. From the source and from the destination as well, we replace the current node by its predecessor to find the piece 1 and 3 (see Figure 20.30) of the path until a loop is reached. At the end, a shortest path is compute between the two tops of the trees as shown in Figure 20.30, it corresponds to the part 2.

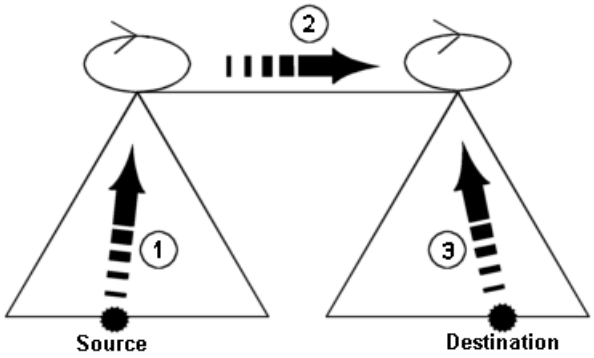


FIG. 20.30 – Arbre.

Simulation results

To evaluate our models, we have try to see if routes generated have properties that are close to the ones studied in section 20.3.1. We had identify four properties :

- the lengths of routes compare to shortest paths.
- the evolution of node' degrees along routes.
- the slight trend in routes to go to the highest degree neihbor.
- the proportion of type of links : forward, backward and stable.

We have generated fake routes on the Skitter graph with the three methods previously seen. We will see the results of the experiments. Unfortunatly, this is just preliminary results.

First, the method based on length was too greedy in computation power, so we are not able to present any results. Indeed, the algorithm stops on some blocking cases like this one : the shortest path between a source and a destination was equal to 14 hops, the value of δ randomly choosen was equal to 5, the algorithm took several days to find that there exist 35968166 paths of length 19 ! One

can imagine some tricks to avoid these blocking cases but it has a great influence on the model and a loss of control on what is exactly generated can appear.

Then, the method based on random perturbations and the one based on local properties have worked well to reproduce length properties of routes. Results are available in Figure 20.31 and Figure 20.33. We found an average traceroute length of 16.77 hops for the random based method and of 14.66 for the local properties based. Concerning the results on the degree evolution along routes plotted in Figure 20.32 and Figure 20.34 for both methods. A preliminary qualitative result is that the method based on local properties gives good results because we find again the result of Figure 20.24.

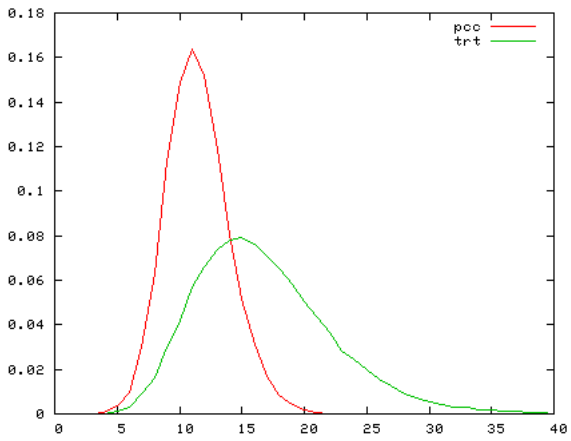


FIG. 20.31 – Probability distribution function of lengths for the random based model.

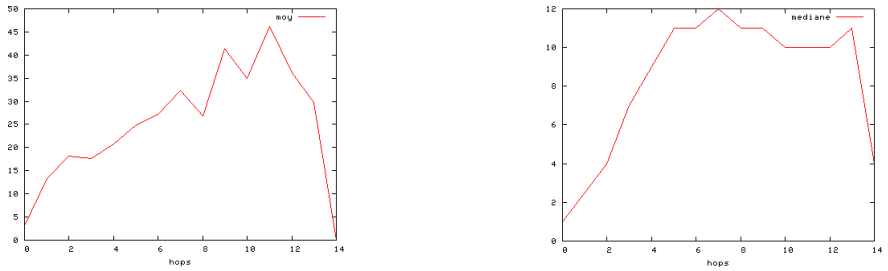


FIG. 20.32 – Degree evolution along routes for the random based model.

From the results obtained experimentally, the best model seems to be the one based on local properties but this need to be confirm.

Naturally, these results and simulation are on going work because we need more statistics on more generated data like the ones on the type of links. We have also begun to generate fake routes on other graphs real and simulated. The real graphs was obtain using the *Mercator* software and public traceroute servers. Generated ones was build with the Erdos-Renyi and the Albert and Barabasi models.

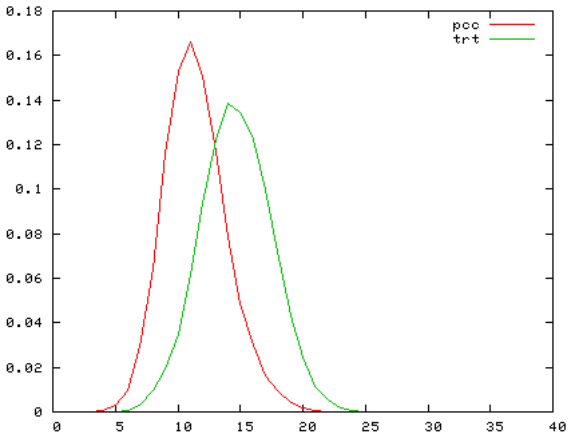


FIG. 20.33 – Probability distribution function of lengths for the local properties based model.

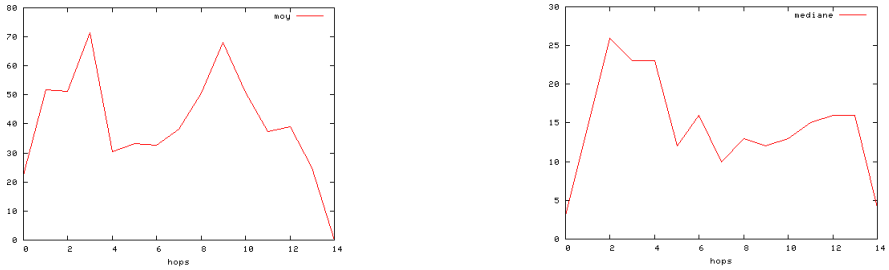


FIG. 20.34 – Degree evolution along routes for the local properties based model.

Conclusion and future works

In this work, after doing a state of the art in topology analysis of the Internet, we have performed some new analysis on the Skitter graph which is a pseudo graph of the Internet, then we have realize a deep study on paths in this network. Our main contribution in this work was to give ideas on what a route on the Internet is and to provide a simple model to improve simulations accuracy. We have also provided tools to manipulate huge IP graphs and to perform analysis on this kind of graph.

For future works, we will try to finish the simulations already started and also to focus on dynamics of routes to answer the following question : If different routes exist between two hosts, do packets usually take the same ? If yes, why ?

Furthermore, we will try to find some solutions to improve the way of grabbing data. The Skitter platform which is at the state of the art must be improved. Indeed, more monitors (*e.g* points of view) are needed to get accurate data and intelligent ways of doing distributed traceroutes have to be found in order to enlarge such measurement platforms and to reduce their overhead on the network.

Appendix

20.4 Some Traceroute related problems

This is an extract of the man page of traceroute.

```
* Note that lines 2 \& 3 are the same.  This is due to a buggy
* kernel on the 2nd hop system -- lbl-csam.arpa -- that forwards
* packets with a zero ttl.
*
* A more interesting example is:
*
*     [yak 72]% traceroute allspice.lcs.mit.edu.
*     traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
*      1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
*      2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms
*      3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms
*      4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
*      5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
*      6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
*      7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
*      8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
*      9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
*     10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
*     11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
*     12  * * *
*     13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
*     14  * * *
*     15  * * *
*     16  * * *
*     17  * * *
*     18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms
*
* (I start to see why I'm having so much trouble with mail to
* MIT.)  Note that the gateways 12, 14, 15, 16 \& 17 hops away
* either don't send ICMP "time exceeded" messages or send them
* with a ttl too small to reach us.  14 - 17 are running the
* MIT C Gateway code that doesn't send "time exceeded"s.  God
* only knows what's going on with 12.
```

```
*
* The silent gateway 12 in the above may be the result of a bug in
* the 4.[23]BSD network code (and its derivatives):  4.x (x <= 3)
* sends an unreachable message using whatever ttl remains in the
* original datagram.  Since, for gateways, the remaining ttl is
* zero, the icmp "time exceeded" is guaranteed to not make it back
* to us.  The behavior of this bug is slightly more interesting
* when it appears on the destination system:
*
*      1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
*      2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  39 ms
*      3  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  39 ms  19 ms
*      4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  19 ms
*      5  ccn-nerif35.Berkeley.EDU (128.32.168.35)  39 ms  39 ms  39 ms
*      6  csgw.Berkeley.EDU (128.32.133.254)  39 ms  59 ms  39 ms
*      7  * * *
*      8  * * *
*      9  * * *
*     10  * * *
*     11  * * *
*     12  * * *
*     13  rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms !
*
* Notice that there are 12 "gateways" (13 is the final
* destination) and exactly the last half of them are "missing".
* What's really happening is that rip (a Sun-3 running Sun OS3.5)
* is using the ttl from our arriving datagram as the ttl in its
* icmp reply.  So, the reply will time out on the return path
* (with no notice sent to anyone since icmp's aren't sent for
* icmp's) until we probe with a ttl that's at least twice the path
* length.  I.e., rip is really only 7 hops away.  A reply that
* returns with a ttl of 1 is a clue this problem exists.
* Traceroute prints a "!" after the time if the ttl is <= 1.
* Since vendors ship a lot of obsolete (DEC's Ultrix, Sun 3.x) or
* non-standard (HPUX) software, expect to see this problem
* frequently and/or take care picking the target host of your
* probes.
*
* Other possible annotations after the time are !H, !N, !P (got a host,
* network or protocol unreachable, respectively), !S or !F (source
* route failed or fragmentation needed -- neither of these should
* ever occur and the associated gateway is busted if you see one).  If
* almost all the probes result in some kind of unreachable, traceroute
* will give up and exit.
*
```

20.5 The format used for Skitter data

```
================================================================================
    sample output (default version)
================================================================================

Key  Source         Destination    Time       RTT    Count  hop1  hop2 ...
--------------------------------------------------------------------------------
C  192.172.226.24  194.225.70.80  978307205  709.657  26 192.172.226.1

Key - parameter characterizing the quality of a trace
-------
  no reply
    N - Noreply
    no reply was received from the destination although a partial
    path may have been recorded. The RTT has no meaning in this case.

  replied
    I - Incomplete
    skitter got a reply from the destination, but did not receive a reply
    from every intermediate hop on the path. The RTT to the destination
    is valid.
    C - complete
    The destination and all intermediate hops in the path all replied.
    The RTT to the destination is valid.

Source and Destination -  IP addresses in standard dotted octet notation.
----------------------
    Source - the IP address of the skitter monitor. The first IP address
    in all paths measured by this monitor.
    Destination - the IP address of the final destination to which the
    packets were sent in a given trace. The last IP address of the
    measured IP path.

Time - UNIX timestamp
-------
    The meaning of this parameter is different in different versions of skitter.
    version 0-9-a3 or earlier (files collected before 2001/02/06):
        In I and C type traces - time when the reply was received
        from the destination
        In N type traces - time when the current cycle (one pass over
            all monitored addresses) of probing started. Same value for all
        N-type traces in a cycle.
    version caida-1.1 (files collected after 2001/02/07):
         In all types of traces - time when the skitter send the first
         probing packet to this destination.

RTT (Round Trip Time) - in milliseconds
---------------------
    The amount of time elapsed between the packet leaving the skitter
    source and the reply from the destination arriving back at the
```

source. Skitter will always take the first such reply from the destination; later replies are thrown away.

Count

The Time To Live (TTL) of the first probe packet for which a reply was received from the destination. Since the packet had to travel through that many routers to reach the destination, this number represents the distance from the source to the destination measured as ‘the number of hops’.

hop1 hop2 ... - IP addresses in standard dotted octet notation or ‘q’.

These are the IP addresses that replied with the ‘TTL expired’ message at each TTL from the source to the destination.
A ‘q’ is printed if no IP address replied at a given TTL.
Sometimes, multiple IPs reply at a given probe TTL. They all are printed in the output, separated by commas. In this case, the number of intermediate IP addresses will be larger than the value of ‘Count’.
It is also possible for a IP address to reply to TTL expired message at the same value as the count.

20.6 Functionalities implemented

For Analysis of traces

Program name	Type	Function
WhereAreTheSpecialIPs	Script	Performs some statistics on placement of special IPs.
Stats	Script	This script performs some statistical analysis on Skitter's data and generates an HTML report for each server.
CreateDotFile	Script	Creates the IP graph from the traces in DOT format, performs some statistics and generates an HTML report.
Bidirectionnalroutes	Script	Extract from Skitter data, traceroutes that exist in one way and also the way back.
ConvertGraphIPToGraphAS	Script	Converts the AS graph from the IP graph, computes some statistics and generates an HTML report.
CreateDotFileWithTroncatedTrt	Script	Creates an IP graph from traces in DOT by removing beginnings of traceroutes and the ends.
Downloads	Script	Looks at the Skitter data available on CAIDA's servers and tells between to dates for a list of servers when full data are available.
DynRoutes	Script	For a server, it follows the evolution of traceroute length toward a set of destination on a long period.

For Analysis of IP graphs

Program name	Type	Function
SkitterAnalysis	C++	Degree Distribution
SkitterAnalysis	C++	Print all degrees of nodes
SkitterAnalysis	C++	Print all clustering coefficients of the nodes
SkitterAnalysis	C++	Several methods to extract the core of the Internet
SkitterAnalysis	C++	Give the proportion of bidirectional links
SkitterAnalysis	C++	Give all the shortest path lengths from a server

For Analysis of traces and IP graphs

Program name	Type	Function
DegreesAlongTrt SkitterAnalysis	Script C++	study the evolution of degrees of nodes crossed by the traceroutes.
AnalyzeLengths SkitterAnalysis	Script C++	Study traceroute lengths versus shortest path lengths
TraficVsDegree SkitterAnalysis	Script C++	Study the correlation between degree of nodes and the traffic
TypeOfLinks SkitterAnalysis	Script C++	Study types of links in traceroutes : Forward, Stable and Backward.
DoesTrtFollowDegree SkitterAnalysis	Script C++	Study the trend for traceroute to choose at a node the highest degree neighbor for the choice of the next node on a route.

For Generation of fake routes

Program name	Type	Function
SkitterAnalysis	C++	Generate traceroutes with the model seen in section 20.3.2
SkitterAnalysis	C++	Generate traceroutes with the model seen in section 20.3.2
SkitterAnalysis	C++	Generate traceroutes with the model seen in section 20.3.2

Chapitre 21

Conclusions

Ce rapport d'avancement est une synthèse des travaux qui ont été réalisés / en cours dans le sous-projet 4, "Échantillonnage et Mesures Actives" du projet RNRT METROPOLIS. Nous avons successivement présenté les parties suivantes : tout d'abord, les travaux réalisés à ce jour, notamment, l'évaluation de performances de l'estimation par mesures actives de la file M/M/1, la caractérisation d'une file M/M/1 bruitée, et le service de localisation d'hôtes IP. Finalement, une synthèse des différents travaux et études en cours a été présentée. Dans chacune des parties nous avons détaillé les difficultés que nous avons rencontrées et nous avons également présenté les solutions que nous envisagions.

Bibliographie

[1] *J.C. Redoutey, F. Roueff, Moment and iid like estimation of an MM1 FIFO from probes delays*. http://www.infres.enst.fr/~casellas/metropolis/metromi/test_mm1iid.html.

[2] *LibWeb*. http://sunsite.berkeley.edu/Libweb.

[3] *RIPE Test Traffic Measurements*. http://www.ripe.net/ttm/, http://www.ripe.net/docs/hostcount.html.

[4] *Tcptrace*. http://irg.cs.ohiou.edu/software/tcptrace/tcptrace.html.

[5] G. Almes, S. Kalidindi, and M. Zekauskas. A one way delay metric for ippm. *Internet RFC 2679*, September 1999.

[6] G. Almes, S. Kalidindi, and M. Zekauskas. A packet loss metric for ippm. *Internet RFC 2680*, September 1999.

[7] G. Almes, S. Kalidindi, and M. Zekauskas. A round-trip delay metric for ippm. *Internet RFC 2681*, September 1999.

[8] S. Alouf, P. Nain, and D. Towsley. Inferring network characteristics via moment-based estimators. In *IEEE Infocom 01*, 2001.

[9] Søren Asmussen. *Applied probability and queues*. Wiley Series in Probability and Mathematical Statistics : Applied Probability and Statistics. John Wiley & Sons Ltd., Chichester, 1987.

[10] Albert-Laszlo Barabasi, Erzsebet Ravasz, and Tamas Vicsek. Deterministic scale-free networks. *Physica A 299, (3-4)*, pages 559–564, 2001.

[11] J. C. Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Jounal of high-speed Networks*, 2(3) :305–323, 1993.

[12] J. C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *SIGCOMM'93*, 1993.

[13] A. Broido and kc claffy. connectivity of ip graphs. 2001.

[14] CAIDA. *GTrace*. http://www.caida.org/tools/visualization/gtrace/.

[15] Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6) :160–163, June 1997.

[16] O. Cappé and F. Roueff. Evaluation numérique de l'information de Fisher pour des observations irrégulières de l'état d'une file d'attente. In *GRETSI 2003*, Paris, France, Septembre 2003.

[17] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. The origin of power laws in internet topologies revisited.

[18] Bill Cheswick, Hal Burch, and Steve Branigan. Mapping and visualizing the internet. pages 1–12.

[19] K. G. Coffman and A. M. Odlyzko. Growth of the internet.

[20] Christopher Davis, Paul Vixie, Tim Goowin, and Ian Dickinson. A means for expressing location information in the domain name system. *Internet RFC 1876*, January 1996.

[21] D. Dehay and J.-F. Yao. On likelihood estimation for a discretely observed markov jump process. submitted.

[22] Amir Dembo and Ofer Zeitouni. *Large Deviations Techniques and Applications*. Jones and Bartlett Publishers, Inc. ISBN 0-86720-291-2, 1993.

[23] M. Doar. A better model for generating test networks, 1996.

[24] P. Erdos and A. Renyi. On random graphs i. *Publ. Math. Debrecen*, 6 :290–297, 1959.

[25] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[26] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *SIGCOMM*, 2000.

[27] The National Laboratory for Applied Network Research (NLANR). http://moat.nlanr.net/IPaddrocc/.

[28] Iffinder from CAIDA. http://www.caida.org/tools/measurement/iffinder/.

[29] Skitter from the Cooperative Association for Internet Data Analysis. http://www.caida.org/tools/measurement/skitter/.

[30] Boston University Representative Internet Topology gEnerator. http://www.cs.bu.edu/brite/.

[31] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.

[32] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[33] N. Hu and P. Steenkiste. Estimating available bandwidth using packet pair probing, 2002.

[34] Bradley Huffaker, Marina Fomenkov, Daniel J. Plummer, David Moore, and k claffy. Distance metrics in the Internet. In *Proc. of the IEEE International Telecommunications Symposium - ITS'2002*, Natal, Brazil, September 2002.

[35] Y. Hyun, A. Broido, and k claffy. Traceroute and bgp as path incongruities.

[36] Ramon Ferrer i Cancho and Ricard V. Solé. The small-world of human language. Technical report, Santa Fe Working paper 01-03-016, 2001.

[37] The Internet Control Message Protocol (ICMP). http://www.freesoft.org/CIE/RFC/792/.

[38] Emily M. Jin, Michelle Girvan, and M. E. J. Newman. The structure of growing social networks. *Phys. Rev. E 64*, (046132), 2001.

[39] S. Jin and A. Bestavros. Small-world internet topologies : Possible causes and implications on scalability of end-system multicast. Technical Report BUCS-2002-004, Boston University, 2002.

[40] Rajesh Kasturirangan. Multiple scales in small-world networks. Technical Report AIM-1663, 1999.

[41] F.P. Kelly. Notes on effective bandwidths. *Stochastic Networks : Theory and Applications*, 1996.

[42] Srinivasan Keshav. Packet-pair flow control, 1994.

[43] R. Koodli and R. Ravikanth. One-way loss pattern sample metrics. *Internet RFC 3357*, March 2002.

[44] A. Lakhina, J. Byers, M. Crovella, and I. Matta. the geographic location of internet resources, 2002.

[45] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in ip topology measurements, 2002.

[46] Anukool Lakhina, John W. Byers, Mark Crovella, and Ibrahim Matta. On the geographic location of Internet resources. *jsac*, 21(6) :934–948, August 2003.

[47] M. Mathis and M. Allman. A framework for defining empirical bulk transfer capacity metrics. *Internet RFC 3148*, July 2001.

[48] C. B. Moler and C. F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20 :801–836, 1979.

[49] David Moore, Ram Periakaruppan, Jim Donohoe, and Kimberly Claffy. Where in the world is netgeo.caida.org ? In *Proc. of the INET'2000*, Yokohama, Japan, July 2000.

[50] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput : A simple model and its empirical validation. *SIGCOMM*, 1998.

[51] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *sigcomm01*, San Diego, CA, USA, August 2001.

[52] V. Paxson. *Measurements and analysis of End-to-end Internet Dynamics*. PhD thesis, U.C. Berkeley, 1997.

[53] Vern Paxson and Sally Floyd. Why we don't know how to simulate the internet. In *Winter Simulation Conference*, pages 1037–1044, 1997.

[54] Vern Paxson, Jamshid Mahdavi, Andrew Adams, and Matt Mathis. An architecture for large-scale Internet measurement. *commag*, 36(8) :48–54, August 1998. http://www.psc.edu/networking/papers/nimi.html.

[55] Stefan Savage Renata Teixeira, Keith Marzullo and Geoffrey M. Voelker. In search of path diversity in isp networks. imc 2003.

[56] R.Sharma R.Siamwalla and S.Keshav. Discovering internet topology. 1999.

[57] Lying Tang and Mark Crovella. Virtual landmarks for the Internet. In *ACM Internet Measurement Conference 2003*, Miami, FL, USA, October 2003.

[58] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators : Degree-based vs structural, 2002.

[59] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker, and Deborah Estrin. The impact of routing policy on internet paths. In *INFOCOM*, pages 736–742, 2001.

[60] the Georgia Tech internetwork topology models generator. http://www.cc.gatech.edu/projects/gtitm/.

[61] Angelos Stavrou. Java Applet to create small-world graph according to (Watts and Strogatz model). http://comet.ctr.columbia.edu/~angelos/smallworld.html.

[62] Tutorial : Using Traceroute. http://www.exit109.com/~jeremy/news/providers/traceroute.html.

[63] University of Illinois at Urbana-Champaign. *IP Address to Latitude/Longitude*. http://cello.cs.uiuc.edu/cgi-bin/slamm/ip2ll/.

[64] Visualware Inc. *VisualRoute*. http://www.visualware.com/visualroute/.

[65] B. M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communications*, pages 1617–1622, 1988.

[66] Artur Ziviani, Serge Fdida, José Ferreira de Rezende, and Otto Carlos Muniz Bandeira Duarte. Demographic placement for Internet host location. In *Proc. of the IEEE Global Communications Conference - GLOBECOM'2003*, San Francisco, CA, USA, December 2003.

[67] Artur Ziviani, Serge Fdida, José Ferreira de Rezende, and Otto Carlos Muniz Bandeira Duarte. Similarity models for Internet host location. In *Proc. of the IEEE International Conference on Networks - ICON'2003*, Sydney, Australia, September 2003.

[68] Artur Ziviani, Serge Fdida, José Ferreira de Rezende, and Otto Carlos Muniz Bandeira Duarte. Toward a measurement-based geographic location service. In *Proc. of the Passive and Active Measurement Workshop - PAM'2004*, Antibes Juan-les-Pins, France, April 2004.

Quatrième partie

Sous Projet 5

Modélisation

METROPOLIS

Métrologie Pour L'Internet

Projet RNRT Exploratoire

Sous Projet 5
Modélisation

Livrable
Rapport d'avancement

Coordinateur du sous-projet : Kavé Salamatian (LIP6)
Auteurs : LIP6 : Serge Fdida, Kavé Salamatian
INRIA : Christine Fricker, Philippe Robert

2003/10/16

Laboratoire d'Informatique de Paris 6 LIP6
Institut National de Recherche en Informatique et Automatique INRIA

Chapitre 22

Préambule

La modélisation du trafic TCP envisagée ici est conduite au niveau des flots et non au niveau des paquets. L'objectif est de mettre en évidence une structure statistique qui permette l'évaluation qualitative de l'évolution du trafic sur des segments du réseau. Les études antérieures et les discussions au sein du projet Metropolis font apparaître assez clairement la nécessité de considérer au moins trois échelles de temps :

- Le niveau paquet. On considère ici uniquement les variables liées au nombre de paquets ou à leur volume transporté. Dans la plupart des cas, la structure de dépendance longue du trafic (voire fractale) ne permet pas une étude mathématique effective comme par exemple de déterminer la probabilité de débordement de la mémoire tampon d'un routeur. Essentiellement, les outils d'analyse markovienne ne peuvent pas être utilisés dans ce cadre. *L'approche traitement du signal semble en revanche appropriée, voir les travaux d'Abry etal.*.
- Le niveau flot TCP. Le trafic Internet est décrit comme une superposition de flots TCP, chaque flots émettant une certain de nombre de paquets sur une plage de temps. Cette description revient à agréger les paquets et "casser" (un peu) la dépendance temporelle qui peut exister au sein du trafic Internet.
- Le niveau session. Les flots TCP sont à leur tour agrégés pour décorréler encore un peu plus le trafic. Cette description est intuitive : chaque utilisateur génère lors de ses connexions un ensemble de requêtes et globalement les utilisateurs se comporte de manière indépendante. Au niveau des mesures cette représentation est en revanche assez délicate à mettre en évidence : une session peut être décrite par un ensemble de $\sim$4-5 paramètres qui ne sont pas faciles à identifier au niveau des traces IP. Un des résultats importants obtenus en 2003 dans le cadre du projet Metropolis concerne une caractérisation indirecte, explicite de cette notion de session (voir la section ci-dessous).

Pour l'étude du trafic TCP, les flots TCP sont divisés en deux classes : les éléphants sont des flots TCP de plus de 20 paquets, les "souris" sont tous les autres flots TCP. Cette dichotomie se justifie de la façon suivante : pour le protocole TCP, le mode majeur de fonctionnement d'un éléphant est l'algorithme "congestion avoidance" et pour une souris c'est l'algorithme "slow start".

Chapitre 23

Modélisation du trafic

L'étude (jointe avec l'équipe de France Telecom Lannion) a commencé avec les traces disponibles, collectées sur le réseau de France Telecom R&D au niveau d'un lien reliant plusieurs plaques ADSL. Une partie très significative de celles-ci est constituée par du trafic "peer to peer" (P2P).

Une étude statistique extensive de ces traces a été conduite. Il a été observé que si la durée entre l'arrivée de deux souris suit une loi exponentielle, les souris n'arrivent pas au lien suivant un processus de Poisson. Cela est dû au fait que les souris arrivent groupées en raison, notamment, des protocoles P2P pour la recherche des serveurs.

Si les souris des trafics non P2P sont agrégées localement par adresse de source, les flots obtenus (appelés macro-souris) ont la propriété de Poisson. Le nombre de souris au sein d'une macro souris a une distribution de Weibull. Le trafic des souris non P2P est donc complètement caractérisé.

Pour les souris P2P, un deuxième niveau d'agrégation est nécessaire au niveau des adresses destination en raison des réponses aux requêtes P2P. Les conclusions sont alors identiques au cas précédent.

Le trafic des éléphants a une caractéristique qui n'a, semble-t-il, jamais été mentionnée dans les études précédentes. La transmission d'un éléphants est régulièrement interrompue pendant une durée de temps de plusieurs secondes. Ses "morceaux" d'éléphants sont alors considérés comme des éléphants distincts (appelés mini-éléphants). Si le volume d'un éléphant a une distribution de Pareto, la durée d'un mini-éléphant suit une loi de Weibull.

Un modèle théorique a été utilisé pour l'analyse quantitative d'un tel modèle de trafic. Le processus des souris est représenté par une file d'attente $M/G/\infty$ où les clients sont caractérisés par leurs instants d'arrivée, leurs durées de service et leurs profiles. En faisant l'hypothèse (vérifiée en pratique) que leur taux d'arrivée est élevé, le processus des souris peut alors être décrit comme un processus gaussien. On a obtenu les expressions explicites de la transformée de Laplace du débit instantané et des corrélations. Des résultats limite au voisinage de la saturation du système ont aussi été démontrés.

Chapitre 24

Interaction des flots TCP

Nous nous sommes intéressés ici à l'interaction des souris et des éléphants sur un lien saturé.

Les souris restant dans la phase "slow start", ces flots n'adaptent pas leur débit à l'état du lien. Schématiquement, les souris occupent une fraction de la capacité de transmission sans contrôle. Les souris sont représentées (voir la section précédente) par un processus d'Ornstein-Uhlenbeck.

À l'inverse, les éléphants adaptent leur débit aux conditions de trafic. Si N éléphants disposent de la capacité C pendant une durée h, chacun d'eux transmettra au débit C/N (caractérisation de partage égalitaire). La capacité C varie bien entendu en fonction de l'activité du trafic des souris.

Mathématiquement cette modélisation revient à considérer une file d'attente $M/G/1$ avec la discipline de service "Processor-Sharing" avec une capacité variable. Ces modèles sont notoirement difficiles à analyser. La plupart des études (de Nunez-Queija notamment) se sont focalisées sur le cas où la capacité varie dans un ensemble discret. Les résultats obtenus sont difficiles à exploiter en raison du nombre de paramètre de ces modèles et des formes explicites des distributions qui font intervenir des expressions matricielles compliquées. Le choix du processus d'Ornstein-Uhlenbeck caractérisé seulement par deux paramètres la moyenne et la variance vise à simplifier la modélisation de tels systèmes.

En raison de la difficulté de ces modèles, on a choisi d'étudier le cas où la capacité de la file d'attente oscille très faiblement autour d'une valeur fixe μ. Des méthodes de perturbations ont été utilisées. La première méthode est analytique, elle consiste à développer la solution d'une équation aux dérivées partielles en fonction d'un petit paramètre. Elle a permis de montrer un principe de capacité de service équivalente pour la file à capacité variable. La deuxième approche est probabiliste : l'impact de l'addition d'un événement à un cycle d'une chaîne de Markov est étudié. Cela permet d'exprimer l'expansion du nombre moyen d'éléphants dans la file à capacité variable jusqu'au second ordre. Le travail actuel se concentre sur les questions de délais de transmission.

Chapitre 25

Modelisation empirique

Felix qui potuit rerum cognoscere causas

Happy is he who can understand the reasons of realities...
Virgile, Georgic

25.1 Introduction

We address, in this paper, the problem of modelling network elements and understanding their behaviour. There exists a complete literature on the art of performance analysis of computer network and a broad spectrum of methods is commonly used to perform capacity planning, modelling and analysis. Analytic modelling, measurement and simulations are frequently used independently or in association to derive accurate solutions able to capture the essence of the system under study. Combining different methods is an interesting solution as it is recognized that each of particular approach is better suited to some target goals. For instance, in computer system modelling, when fine grain analysis is crucial, a first step consists in developing a simulator of the system while, at the same time, a measurement campaign is conducted to extract some important system performance metrics. The later are exploited, in a second step, in order to calibrate the system simulator. We propose a general method that combines measurement and modelling in order to understand some important Internet performance metrics.

Two main classes of modelling approaches have been applied so far to address performances in networking context : the constructive and the descriptive approach.

The constructive approach has been widely used since many years to model systems in general. It is based on the derivation of a model that ideally produces the same output that the system for an outside observer. These models might be based on a description of network elements those are as close as possible to the real network. The network is described as a combination of queues and routers *etc.* that causally relate an input scenario defining the parameters of the system in term of arrival process, capacity, buffer space, *etc.*, to output parameters. The modelling phase is followed by a resolution phase that either relies on simulation or analytical analysis to derive performance metrics for a given set of input parameters. This approach is widely adopted in performance analysis

and queueing theory. The general use of packet level network simulator, as *ns* [1], as simulation tool for analysing complex networks have made possible very precise and detailed modelling of network elements. Constructive approaches have the nice property of relating directly performance metrics as can be perceived by end users to operational traffic engineering parameters that might be controlled by network operators. It can also answer "what if" questions, arising when one want to evaluate the impact of changes in network parameters or architecture in the performance of the system under study. Nevertheless, this approach suffers from a main drawback ; the assumptions made about the structure of the network and the scenario are so strong that it is very difficult to generalize results of constructive approach to the real Internet.

On the other hand, the descriptive approach is based on measurements made over operational networks. It models the observations by describing them by some statistical parameters such as moment of different order (mean, variance, autocorrelation, Hurst parameter, *etc.*). In this approach the network is seen as an opaque black box and no access to its internal structure is supposed. The descriptive approach only describes the observations without explaining the mechanism generating the observations. This process mainly aims toward predicting the QoS experienced by applications under some reproducibility or stationarity assumptions. However, as these models do not integrate the mechanisms generating the observations, they cannot help on predicting what will happen if these stationarity assumptions do not hold anymore. This means that this approach cannot be used for network dimensioning, capacity planning or predicting the QoS improvement consecutive to changes in network parameters.

An important goal in any scientific investigations, is to *interpret* results. By interpretation we mean being able to relate effects to causes. Without interpretation, any measurement results remain at best anecdotal, and no proactive control can be done. However, interpretation is not trivial, as it needs an insight into the observed phenomenon that might be unreachable to the observer with incomplete knowledge. This incompleteness might results from partial observation or from partial understanding of the underlying process leading to the observations.

The situation of the engineer confronted to measurement obtained over the Internet is almost similar to the situation nicely described by Socrates in the famous Plato cavern allegory [16]. The allegory is the story of peoples who have been held prisoner in a den deep inside a mountain. They have been there for so long that the cave has gradually become the only world they know. The only light they can see is the light from a fire that is maintained on the other side of the cave, which is reflected off the rocks of the cavern in front of them. In this manner, their world has become a world of a faint glow of light and of huge shadows of objects that are passed in front of the fire. The shadows thus become a reality to them that in part defines their world, that they react to. If someone manages to break away from the chains, and begins to observe the process that creates the shadows, he will begins to understand that the mythology of the world the had created for him is not real, but was merely a construct of its deduction from the limitations of its perception.

By essence, measurements are always incomplete as they are always bounded in time and space. This fact is truer in the context of the Internet that spread over continent and timescales. Moreover, our understanding of the complex interactions occurring in the network is still embryonic. Even when we have good understanding of some internal mechanism, the combinatorial explosion of the number of influent parameters drive the observer to use a parsimonious and incomplete model of the observed reality. These facts somewhat explain why a lot of observed phenomenon over the Internet still remains without any convincing interpretation. Examples of these unanswered question are the

sources (and the interpretation) of self-similar behaviour and long-range dependences in traffic, or the causes of the power law in the topological graphs of Internet.

Our aim in this paper is to provide a framework for interpreting measurements. This framework might be applied to a large class of measurement problem. The approach will consist of mixing the two previously described modelling approaches : constructive and descriptive. At the first step, we use constructive approach to define some models relating effects to causes, or differently said, relating parameters of unseen (hidden) input scenario to observed and measured parameters. This constructive model will be used as an *a priori* base for interpretation. The interpretation occurs afterthat when descriptive approach and statistical inference are used to infer the hidden input scenario that have led to the actual observations.

This framework enable a formulation of the interpretation problem as an inverse statistical estimation problem. In this problem the goal is to infer the input scenario of an *a priori* constructive model that have more likely led to the observations. This inverse problem might be solved by several methods, as we will explain. Moreover, we will describe some networking problems that appears at first attempt as completely separate problems, but reduce to the same inverse statistical estimation problem by using the described framework, demonstrating its importance as a generic approach.

25.2 General Framework

Suppose that we are interested in a given performance metric $\mathbf{x}$ (vector-valued variable). Unfortunately, $\mathbf{x}$ can not be measured directly (hidden variable), although, we can measure another causally related variable $\mathbf{y}$; by causally related we means that $\mathbf{x}$ might be interpreted as a cause of $\mathbf{y}$ through a model $\mathcal{D}$ (possibly deterministic or probabilistic) relating $\mathbf{x}$ to $\mathbf{y}$ and some other variables $\mathbf{z}$, *i.e.* $\mathbf{y} = \mathcal{D}(\mathbf{x}, \mathbf{z})$. Model $\mathcal{D}$ might come from a constructive approaches as described in the introduction.

It might frequently arise that we have not access to all causal variables,*i.e.* $\mathbf{z}$ or $\mathbf{x}$ might be unobserved. In this case probabilistic modelling is applied ; $\mathbf{x}$ and $\mathbf{y}$ are assumed to be realizations of correlated stochastic processes $\mathbf{X}$ and $\mathbf{Y}$, defined over a probability space $(\mathcal{X} \times \mathcal{Y}, \mathcal{E}, \mathcal{P})$, *e.g.* a sample space $\mathcal{X} \times \mathcal{Y}$, a σ-algebra of events $\mathcal{E}$ and a probability measure $\mathcal{P}$. We suppose that $\mathbf{X}$ and $\mathbf{Y}$ are related through a parametric (with parameter θ) stochastic model $\mathcal{M}$, *i.e.* $\mathbf{Y} \stackrel{d}{=} \mathcal{M}(\theta, \mathbf{X})$; by stochastic model we mean that the model give the probability distribution function (*pdf*) of the resulting random variable and this is emphasized by the use of the sign $\stackrel{d}{=}$ meaning an equality in distribution. Thereafter we will assume that all equality between random variables are by distribution and we will not use anymore the $\stackrel{d}{=}$ sign.

We assume that we have measured a realization $\mathbf{y}$ of the output process $\mathbf{Y}$ and the input process $\mathbf{X}$ is hidden from us. The interpretation problem with an *a priori* interpretation model $\mathcal{M}$ consists of finding the sequence $\mathbf{x}$ of inputs that have more likely led to the observation $\mathbf{y}$ (Fig. 25.1). In this framework the problem under study reduces to an estimation problem where estimation refers to the general subject of making inferences about the value(s) of one (or more) random variable(s) based on observations (measurements) of one or more related random variables [24].

In general terms, we have to find an inference $\delta(\mathbf{y}) = (\hat{\mathbf{X}}, \hat{\theta})$ of the hidden variable $\mathbf{X}$ and the parameter θ based on the observation $\mathbf{y}$ such that an optimality criterion is satisfied. The optimality

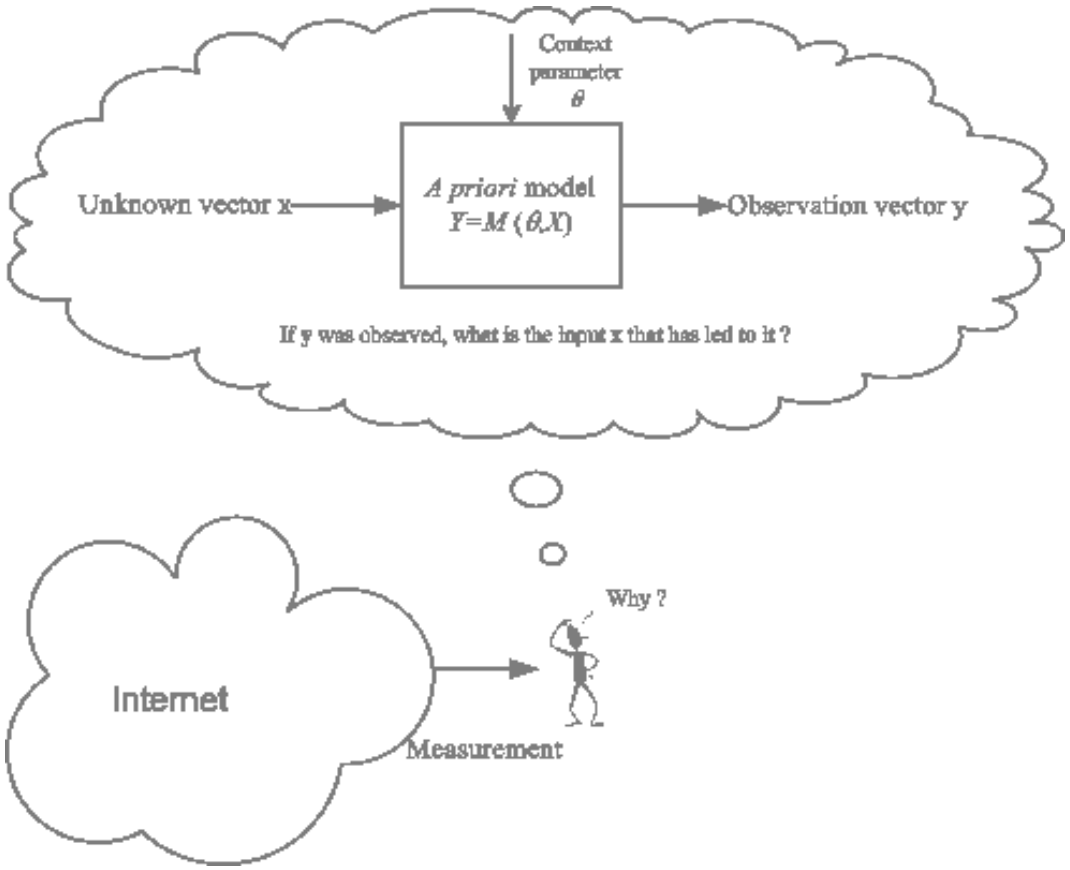


FIG. 25.1 – The general interpretation problem

criterion for dealing with the estimation problem can be defined based on every form of cost function $\mathcal{C}(\mathbf{X}, \hat{\mathbf{X}})$ suitable to a particular applicative need. We will describe further this general case. This general problem can be splitted to two problems that may not be independent : the *modelling problem* and the *interpretation problem*.

In the described framework the modelling problem goal is to infer the parameter θ in relation $\mathbf{Y} = \mathcal{M}(\theta, \mathbf{X})$ that will most likely lead to the observation $\mathbf{y}$. Specific case of this problem is given by descriptive approaches which usually try to describes the observations (measurements) through a stochastic parametric model defined by $\mathbf{Y} = \mathcal{N}(\theta)$ (where no input variable $\mathbf{X}$ is assumed) by inferring the parameter θ. As an example of this kind of approaches we can give the large literature, that assume a model $\mathcal{N}$ where the observation have a multi-fractal or long-range dependent structure, and one have to infer parameters of the model based on the observations. We will give other examples in the forthcoming.

On the other hand the interpretation problem tries to find the vector $\delta(\mathbf{y}) = \hat{\mathbf{x}}$ (realization of the random vector $\mathbf{X}$) that will interpret the measured $\mathbf{y}$ by a cause and effect relationship given by $\mathbf{Y} = \mathcal{M}(\theta, \mathbf{X})$ (where the parameter θ are assumed to be known). Solving this problem is the ultimate goal of measurement interpretation, as based on the *a priori* interpretation model, it gives

the set of causes ($\hat{\mathbf{x}}$) that have more likely lead to the observation ($\mathbf{y}$). This kind of approach has also been applied in a lot of problems in the networking literature. One bright example is the so-called network tomography problem [22, 11, 6], where one have to infer the internal characteristics of the network (Traffic matrix, link delay, link losses, *etc.*) based on some correlated observed value. We will describe more completely this example in next section.

Before going further and giving any explanatory examples, we have to answer a concern that might pop up in the reader mind ; the proposed interpretation framework is based on an *a priori* interpretation model, and the interpretation results are conditioned on this prior model. How can we find a good model for interpretation and what happen if the *a priori* model is bad ? This question is as old as modelling itself. Almost all natural phenomena lack of universal models that are good for all purposes. A specific model and a resulting interpretation might be good in one context and insufficient for another one. The quality of a model cannot be defined *per se*, and it need to be evaluated in relation with the applicative context of the model.

The proposed framework gives a natural way to evaluate the quality of an interpretation. As explained before, the outcome of the interpretation problem is an inferred sequence of input $\hat{\mathbf{x}}$ that most likely led to the observation of $\mathbf{y}$. Application of the sequence of input ($\hat{\mathbf{x}}$) to the model lead to a predicted sequence of observations $\hat{\mathbf{y}} = \mathcal{M}(\hat{\theta}, \hat{\mathbf{x}})$ that might be compared with the observation by the way of a cost function $\mathcal{C}(\mathbf{y}, \hat{\mathbf{y}})$ evaluating the error. Definition of the cost function depends on the particular applicative setting. However, some popular cost function, as the root mean squared error (rms), are frequently used.

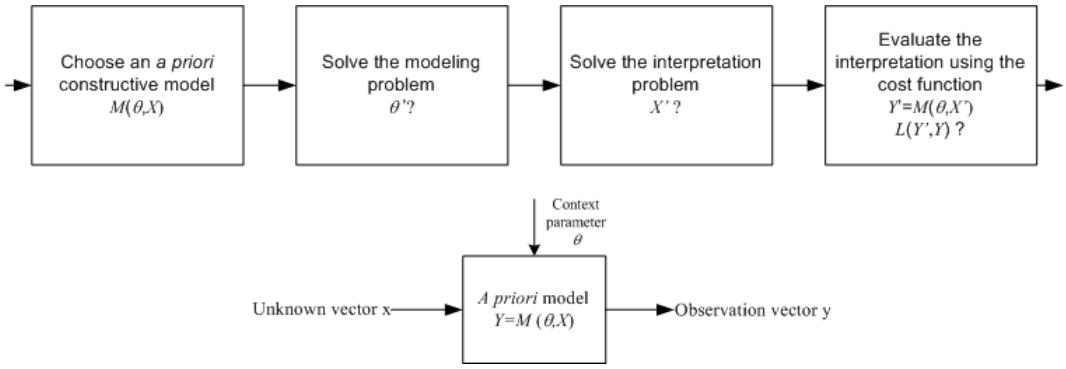


FIG. 25.2 – Needed steps for interpretation of measurements

In next section, we will explain the proposed framework by giving a complete application to two examples : interpretation of active measurements and network tomography.

25.3 Explanatory examples

Previous section has described the interpretation framework in general. In this section we will apply the framework to two well-known problems, and shows that presented solutions to these problems are in fact employing unconsciously the interpretation framework as developed before. We will first describe the interpretation of active measurement problem.

25.3.1 Interpretation of active measurement

In active measurement a probe sending process injects probe packets into the network. At the other end of the network a measurement agent records some metrics on each received probe packet. The collected metrics are used to infer about the QoS that will be seen by other packet flow crossing the network. The rationale behind active measurement is that end to end QoS as sensed by real application can be measured by being in competition with the real application.

The IPPM group of the IETF has defined some end-to-end performance metrics [14] to be collected on probe packets. Three main type of information are extracted from the received probe packet flow : packet size, packet loss process and packet delay process. These information are used to derive more complex metrics as goodput, loss rate, loss run length, jitter, *etc*. The probe packets are usually sent using ICMP (in ping surveys) or UDP. A lot of active measurement surveys has been produced during the recent years [13, 3, 5, 3, 25, 26] and some measurement infrastructures have been deployed [15, 10].

Active measurements are the source of some interesting and challenging problems. We will describe here the interpretation of loss metrics measured by active measurement.

Let $\mathbf{X} = (X_t)_{t=1}^{T}$ be a sampled loss trace measured by an active measurement flow over an Internet link. The loss trace is defined as $X_t = 0$, if the t^{th} active measurement probe reaches its destination and $X_t = 1$, if the probe is lost.

Losses observed by measurement probes are mainly due to buffer overflow in router in the path. The main cause that has led to the observation of losses is the cross traffic competing with the active measurement flow. In this context interpretation means to find the characteristics of the cross traffic that have led to the observed loss trace.

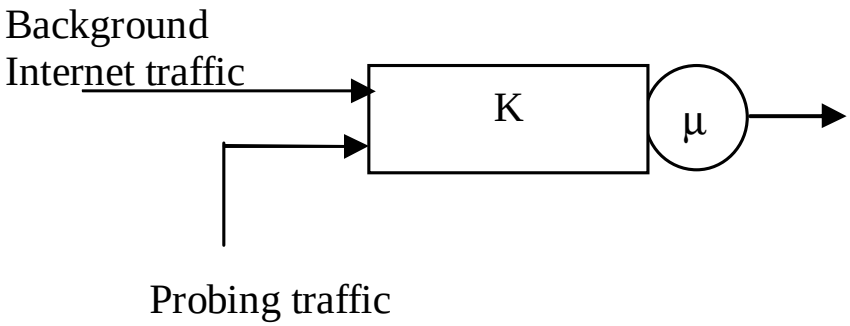


FIG. 25.3 – *A priori* constructive model of the network used for interpretation

For doing this interpretation, we need an *a priori* constructive model of the Internet path relating the causes (competing cross traffic) to the effects (losses observed on the probes). We assume a simple constructive model $\mathcal{M}$ (fig. 25.3) for the network consisting of a single bottleneck model with transmission capacity μ and buffer size K. The bottleneck is fed by the probing flow and by the Internet background traffic. We assume that the background traffic follows a Markov Modulated Poisson Process (MMPP). The MMPP traffic model describes the traffic entering the bottleneck, as the superposition of the background Internet traffic and measurement probe traffic. This model as-

sumes that the traffic switches between K different states following a continuous time homogeneous Markov chain with infinitesimal transition matrix Q. Each state represents an homogeneous poisson process with parameter $\Lambda_i = \lambda_i + \gamma$, $i \in \{1, \ldots, K\}$, where Γ is the measurement probe traffic and λ_i represent the Internet background traffic. This model is very suitable as it is able to approximate every type of traffic conditioned on the number of state is sufficiently large (even long-range dependence can be approximated with large number of state and/or semi separable states).

The *a priori* model $\mathcal{M}$ is governed by the parameters $\theta = (\mu, K, \Lambda, \Gamma)$ and accept as an input the sequence of states of the MMPP background Internet traffic. The output is the loss process measured over the measurement flow.

In this context the previously defined modelling and interpretation problem can be stated as :

Modelling problem : Observing a loss trace over the measurement probes what are the parameter $\theta = (\mu, K, \Lambda, \Gamma)$ of the *a priori* model $\mathcal{M}$.

Interpretation problem : What is the particular sample path of the states of the background Internet traffic that has caused the observed loss trace.

In the modelling problem, the parameter μ might be estimated faithfully by packet pairs (or packet train) approaches in the literature [9]. We can therefore assume that it is known. The modelling problem has been treated in the simple case of a Poisson traffic model (MMPP with only one state) in [2]. An extension to the general MMPP case as described here, has been done in [21]. This paper propose a formula relating the MMPP parameters to the parameters of a Hidden Markov Model as described in [18]. [21] shows also that the parameter K is not very sensitive for the purpose of interpretation.

The modelling problem, *i.e.* estimation of remaining parameters $\theta = (\Lambda, \Gamma)$, is solved by using the Expectation Maximization method that will be described further in section 25.4. The interpretation problems is solved using the viterbi algorithm or a Maximum *a posteriori* methodology that will be also describe 25.4.

25.3.2 Network tomography

Another example of application of the described interpretation framework is given by the so called “Network tomography” problem. This name was first coined by Vardi in [22] for the problem of traffic matrix estimation based on observed volume of traffic in network links. The name was later extended to a large class of problems consisting of inferring internal network characteristics based on end-to-end measurements [6]. Even if these two problems seem to be far apart, however the mathematical formulation of them remains the same, explaining the similarity of the name attributed to them. These two problem can be reduced to the solution of a highly under-dimensioned linear equation of the form $AX = Y$, where A is a kind of routing matrix, X is the vector of unknowns characteristics to infer and Y is the vector of observations.

For the sake of simplicity we will only described here the first network tomography problem as defined by Vardi, which consist of estimating the traffic matrix based on some partial observation of the traffic volume on internal network link. However the approach can be also applied to the second class of network tomography problems.

A Traffic Matrix (TM) represents the volume of traffic that flows between all source-destination pairs in a network. In a TM $\mathbf{X} = (X_{ij})$, rows and columns represent nodes in the network, and element x_{ij} represents the volume of traffic exchanged from node i to j. Traffic matrix is an important parameter for designing and dimensioning network topology and architecture as it gives the traffic demand. However, deriving directly this matrix is unfeasible. A lot of works have dealt with the estimation of traffic matrices using traffic count on network links that can be easily measured by SNMP MIBs [22, 7, 20, 11, 23].

However, traffic matrix have also another meaning that is relevant to the subject of this paper : the volume of traffic X_{ij} exchanged between nodes is the cause of the traffic observed in network links. If somebody wish to interpret traffic count measured on network links at time t, he will have to estimate the traffic matrix at time t. The presented framework might be applied to this problem. In fact proposed solutions to the matrix tomography problem follow unconsciously the framework that we have described.

For dealing with this interpretation problem, the *a priori* constructive model used is the following. Let c be the number of origin-destination (OD) pairs. If the network has n nodes, then $c = n * (n-1)$. Rather than represent the amount of data transmitted from node i to node j as X_{ij}, it is preferable to represent the list of OD pairs as a vector. We thus order the pairs and let X_k be the amount of data transmitted by OD pair k. Let $Y = (y_1, ...y_r)$ be the vector of link counts where y_l gives the link count for link l, and r denotes the number of links in the network. The link counts refer to the link load which is obtained via SNMP data. The vectors X (causes) and Y (observations) are related through a routing matrix A, which is an r by c matrix. Each routing matrix component $a_{ij} \in 0, 1$ where $a_{ij} = 1$ if link i belongs to the path used for OD pair j, and $a_{ij} = 0$ otherwise. The OD flows are thus related to the link counts according to the linear relation :

$$\mathbf{Y} = A\mathbf{X} \tag{25.1}$$

This means that the traffic matrix estimation problem is equivalent to solving the previous linear equation. However this linear equation is largely under-dimensioned as the number of link in a network is largely smaller than the number of OD pairs. Nevertheless, this equation can be used as the base of an *a priori* model for interpretation. We need to define a parametric structure for the distribution of the unobserved causes $\mathbf{X}$. Let's suppose that $\mathbf{X}$ follows a probability distribution $\mathcal{P}(\alpha)$ with unknows parameters α, *e.g.* a normal distribution with $\alpha = (\mu, \sigma)$ or a Poisson distribution with $\alpha = \lambda$, *etc.*.

The *a priori* model $\mathcal{M}(\theta, \mathbf{X})$ is governed by $\theta = (A, \alpha)$, where α is the unknown parameters of the probability distribution of $\mathbf{X}$. The routing matrix A of an IP networks can be obtained by gathering the OSPF or IS-IS links weights and computing the shortest-paths between all OD pairs. We need only to estimate α.

In this context the previously defined modelling and interpretation problem can be stated as :

Modelling problem : Observing a link traffic count vector $\mathbf{Y}$ what are the parameter $\theta = (\alpha)$ of the *a priori* model $\mathcal{M}$.

Interpretation problem : Knowing the parameter θ, *i.e.* parameters of the distribution of OD pairs, and observing a link traffic count vector $\mathbf{Y}$, what is the OD pairs vector $\mathbf{X}$ that have led to observation of $\mathbf{Y}$. In other terms the interpretation problem consists of solving the linear equation $A\mathbf{X} = \mathbf{Y}$ knowing the probability distribution of the hidden values $\mathbf{X}$.

As explained before several researches have dealt with these two problems. Some of these researches differ in the parametric hypothesis for the *a priori* probability distributions of OD pairs $\mathcal{P}(\alpha)$, *e.g.* [22, 20] assume a Poisson distribution, [7] assumes a Gaussian distribution where the mean and the variance are related by $\sigma_i^2 = \gamma\mu_i^c$ and [23] assume a MMPP model for the OD pairs. These researches differ also in the resolution method used for solving the modelling and interpretation problem, *e.g.* [22] use a method of moment for solving the modelling problem, [20] use a Bayesian estimation framework for solving the modelling and interpretation problem, [7] use an EM method for solving the modelling problem and an Iterative Proportional Fitting (IPF) for solving the interpretation problem. [11] make a comparison of these different approaches.

In this section, we have applied the proposed interpretation framework to two problems. We have demonstrated that even if these two problems seems far away, the resolution framework for the two is essentially the same. This conclusion might be reached over a large class of problem in Internet measurement analysis. In next section we will describe how to solve the interpretation and modelling problem and compare the different approach to the problem.

25.4 How to solve it

Previous sections have introduced the interpretation of measurement obtained over Internet as an inversion estimation problem. This section will give some guidelines for solving the modelling problems as described before. Three main approach are detailed here : the EM method based on the maximum likelihood paradigm, the Bayesian approach based on the Bayes theorem and the Maximum *a posteriori* approach and the regularization approach related to the maximum entropy paradigm. For the sake of simplicity, we will explain here the discrete variable framework where all variables, i.e. $\mathbf{X}, \mathbf{Y}, \theta$ *etc.*, are discrete random variables and it is meaningful to speak about $\mathbb{P}\text{rob}\{.\}$. The presentation can be easily extended to the continuous variable case by replacing the probability, density function and summations by integrals.

The interpretative model $\mathcal{M}(\mathbf{X}, \theta)$ defines two probability distribution function : an *a priori* distribution function for the hidden input of the model $\mathbb{P}\text{rob}\{\mathbf{X}; \theta\}$ and a conditional distribution function $\mathbb{P}\text{rob}\{\mathbf{Y}|\mathbf{X}; \theta\}$ giving the input-output relationship. These two distribution are the basis of the inverse statistical problem we have to solve for interpreting measurements.

25.4.1 EM method

The first approach for dealing with the modelling problem is based on the maximum-likelihood paradigm. Recall the definition of the maximum-likelihood paradigm. Let suppose that we have observed a realization $\mathbf{x}$ of a random vector $\mathbf{X}$, the log-likelihood function given the observed sequence $\mathbf{x}$ is defined as $\mathcal{L}(\theta|\mathbf{x}) = \log \mathbb{P}\text{rob}\{\mathbf{X} = \mathbf{x}; \theta\}$. One important point is that in the maximum likelihood paradigm, the parameter θ is not a random variable and it is rather an hidden fixed value that have to be computed. It is therefore meaningless to condition a probability by θ. This point is emphasized by using in notations a semi-colon to separate θ from other variables. The log-likelihood function might be thought of as a function of the parameters θ where the observed vector $\mathbf{x}$ is fixed. In the maximum likelihood paradigm, the value of θ is chosen such that it maximizes the log-likelihood

function.

$$\hat{\theta} = \text{Arg}\max_{\Theta} \mathcal{L}(\theta|\mathbf{X})$$

Depending on the form of the density function $\mathbb{P}\text{rob}\{\mathbf{X} = \mathbf{x}; \theta\}$ this problem can be easy or hard to solve. For some problem it is not possible to find simple analytical expressions for the log-likelihood function, and we must resort to other techniques, as the EM method that will be described.

The modelling problem could be rewritten in the maximum log-likelihood context as finding the value $\hat{\theta}$ that maximizes the complete-data log-likelihood function given the observation of $\mathbf{y}$ defined as $\mathcal{L}(\theta|\mathbf{X}, \mathbf{Y} = \mathbf{y}) = \log \mathbb{P}\text{rob}\{\mathbf{X}, \mathbf{Y} = \mathbf{y}; \theta\}$.

$$\hat{\theta} = \text{Arg}\max_{\Theta} \mathcal{L}(\theta|\mathbf{X}, \mathbf{Y} = \mathbf{y})$$

The complete-data log-likelihood function is obtained using $\mathbb{P}\text{rob}\{\mathbf{Y}, \mathbf{X}; \theta\} = \mathbb{P}\text{rob}\{\mathbf{Y}|\mathbf{X}; \theta\}\mathbb{P}\text{rob}\{\mathbf{X}; \theta\}$ that is computable as we have the interpretation model $\mathcal{M}$. However, as the vector $\mathbf{X}$ is hidden the previous optimization cannot be done easily. EM method is used to deal with situation where we have missing values, due to problems with or limitations of the observation process. This is exactly the situation we have to deal with.

As $\mathbf{X}$ is hidden, the complete-data log-likelihood function $\mathcal{L}(\theta|\mathbf{X}, \mathbf{Y} = \mathbf{y})$ can be assumed as a random variable. By fixing the value of the parameter $\theta = \hat{\theta}^{(i)}$, the marginal distribution of the unobserved input vector $\mathbf{X}$ can be obtained as $\gamma^{(i)}(\mathbf{X}) = \mathbb{P}\text{rob}\{\mathbf{X}|\mathbf{Y} = \mathbf{y}; \theta = \hat{\theta}^{(i)}\}$. The function $Q(\theta, \hat{\theta}^{(i)})$ can be defined as the expectation of the complete-data log-likelihood function over the marginal distribution $\gamma^{(i)}(.)$:

$$Q(\theta, \hat{\theta}^{(i)}) = \mathbb{E}\{\mathcal{L}(\theta|\mathbf{X}, \mathbf{Y} = \mathbf{y})|\mathbf{Y} = \mathbf{y}; \hat{\theta}^{(i)}\}$$

In some case the distribution $\gamma^{(i)}(.)$ is a simple analytic expression of the assumed parameter $\hat{\theta}^{(i)}$ and the observed data $\mathbf{y}$. This favourable case occurs when $\mathbf{X}$ and $\mathbf{Y}$ are jointly Gaussian or when we are in the Hidden Markov Model context where Baum-Welches backward and forward equations apply [4]. In worst case this marginal distribution might be difficult to compute and we have to use in place of it $\mathbb{P}\text{rob}\{\mathbf{X}, \mathbf{Y} = \mathbf{y}; \theta = \hat{\theta}^{(i)}\}$ which is more easy to derive. This will not affect the overall convergence of the EM methods, but it might slow down it.

The EM method consists of iterations of two steps : an Expectation step where the function $Q(\theta, \hat{\theta}^{(i)})$ is evaluated and a Maximization step where the maximum-likelihood paradigm is applied to previously define mean log-likelihood computed by $Q(\theta, \hat{\theta}^{(i)})$ which is maximized with respect to θ to derive the next value $\hat{\theta}^{(i+1)}$.

Expectation Step : In this step, we have to compute the expectation of the log-likelihood over the marginal distribution $\gamma^{(i)}(\mathbf{X})$ to derive $Q(\theta, \hat{\theta}^{(i)}$. In some case $Q(\theta, \hat{\theta}^{(i)})$ have a simple analytic form that simplify the maximization step.

Maximization Step : Maximize function $Q(\theta, \hat{\theta}^{(i)})$ with respect to θ and derive the new estimate $\hat{\theta}^{(i+1)}$:

$$\hat{\theta}^{(i+1)} = \text{Arg}\max_{\Theta} Q(\theta, \hat{\theta}^{(i)}).$$

If $Q(\theta, \hat{\theta}^{(i)})$ has analytic form, the maximization might be done by solving the following equation :

$$\frac{\partial Q(\theta, \hat{\theta}^{(i)})}{\partial \theta} = 0.$$

The maximization might lead to simple analytic formula in some case where $Q(\theta, \hat{\theta}^{(i)})$ have suitable form. However in some case, the maximization is too complex to be done completely. We might use the generalized EM method which differ from the EM method by the fact that $\hat{\theta}^{(i+1)}$ is set such that $Q(\theta, \hat{\theta}^{(i+1)}) > Q(\theta, \hat{\theta}^{(i)})$, *i.e.* $\hat{\theta}^{(i+1)}$ is not suppose to maximize $Q(\theta, \hat{\theta}^{(i)})$.

These two steps are repeated as necessary to be convinced of the convergence to a fixed point. Each iterations is guaranteed to increase the log-likelihood and the algorithm is guaranteed to converge to a local maximum of the log-likelihood function. However, the local minimum is not necessarily a global minimum. This problem have motivated some statistical variation of the EM method, as the SAEM (Simulated Annealing Expectation maximization) [8]. These variations introduce some controlled amount of noise into the process to push out the optimization from local minimum. But even with such methods, the EM method is highly sensitive to the initial value θ^0 used in the first iteration. The better this initial guess is, the lower the chance to get stick in a local minimum. It is therefore helpful to have an intelligent choice of this initial value. How to make this intelligent choice is a question of expertise and deep knowledge of the problem to analyze.

The next step after inferring the parameter $\hat{\theta}$ and solving the modelling problem is to deal with the interpretation problem. As explained before we need to derive the distribution $\gamma^{(i)}(\mathbf{X}) = \mathbb{P}\text{rob}\big\{\mathbf{X}|\mathbf{Y} = \mathbf{y}; \theta = \hat{\theta}^{(i)}\big\}$ in each step of the EM method. At the last step we have the *a posteriori* probability $\mathbb{P}\text{rob}\big\{\mathbf{X}|\mathbf{Y} = \mathbf{y}; \theta = \hat{\theta}\big\}$ as a side result of the EM estimation. This probability might be used to estimate $\hat{\mathbf{X}}$ and solve the interpretation problem :

$$\hat{\mathbf{X}} = \text{Arg}\max_{\mathcal{X}} \mathbb{P}\text{rob}\big\{\mathbf{X}|\mathbf{Y} = \mathbf{y}; \theta = \hat{\theta}\big\}$$

In some case, we might add some constraint in the previous optimization and reduce the search space, *e.g.* in the matrix tomography problem the constraint $A\mathbf{X} = \mathbf{Y}$ is added to validate the traffic constraint.

We have presented here the EM method in its most general form. The details of the steps required to compute the given quantities are very dependent on the particular application. [7] illustrate the application of EM method to the network tomography problem. [18, 21] presents the application of EM method to the interpretation of active measurement and [19] use SAEM method in the context of flow classification.

In next section we will describe the solution of the modelling problem in the context of Bayesian analysis.

25.4.2 Bayesian framework

The second approach for solving the modelling problem is based on the Bayesian framework. The main difference between this approach and the previously described Maximum likelihood approach is that in the Bayesian framework θ is a random variable that will have a probability distribution function $\pi(\theta)$ and it is meaningful to define a probability conditioned on θ, where in the previous approach θ is seen as an unknown but fixed parameter. This comment is not anecdotal, as it traduces profound methodological differences between the two approaches. The Bayesian approach is completely based on the definition of a prior distribution $\pi(\theta)$, where maximum likelihood do not need any prior distribution hypothesis. Use of prior is the most critical and most criticized point of Bayesian analysis.

Recall a description of the Bayesian approach. The framework is based on the Bayes theorem relating the conditional probability of two events A and B :

$$\mathbb{P}\text{rob}\big\{A|B\big\} = \frac{\mathbb{P}\text{rob}\big\{B|A\big\}\mathbb{P}\text{rob}\big\{A\big\}}{\sum_{\text{allevents}E}\mathbb{P}\text{rob}\big\{B|E\big\}\mathbb{P}\text{rob}\big\{E\big\}}$$

This theorem is very helpful for interpreting measurement. As described in previous section, the *a prior* model $\mathcal{M}$ defines a conditional probability distribution function $\mathbb{P}\text{rob}\big\{\mathbf{Y}|\mathbf{X},\theta\big\}$. However as we have observed $\mathbf{y}$ and $\mathbf{X}$ and θ are missing, we would rather be interest on knowing the inverse conditional probability $\mathbb{P}\text{rob}\big\{\mathbf{X},\theta|\mathbf{Y}=\mathbf{y}\big\}$. The Bayes theorem enable us to compute $\mathbb{P}\text{rob}\big\{\mathbf{X},\theta|\mathbf{Y}=\mathbf{y}\big\}$ by :

$$\mathbb{P}\text{rob}\big\{\mathbf{X},\theta|\mathbf{Y}=\mathbf{y}\big\} = \frac{\mathbb{P}\text{rob}\big\{\mathbf{Y}|\mathbf{X},\theta\big\}\mathbb{P}\text{rob}\big\{\mathbf{X}|\theta\big\}\pi(\theta)}{\sum_{\mathcal{Z},\Omega}\mathbb{P}\text{rob}\big\{\mathbf{Y}|\mathbf{Z},\theta\big\}\mathbb{P}\text{rob}\big\{\mathbf{Z}|\theta\big\}\pi(\omega)}$$

After obtaining the so called *a posteriori* probability distribution $\mathbb{P}\text{rob}\big\{\mathbf{X},\theta|\mathbf{Y}=\mathbf{y}\big\}$, we might compute marginal *a posteriori* probability defined as :

$$\begin{aligned}\mathbb{P}\text{rob}\big\{\theta|\mathbf{Y}=\mathbf{y}\big\} &= \sum_{\mathcal{X}}\mathbb{P}\text{rob}\big\{\mathbf{X},\theta|\mathbf{Y}=\mathbf{y}\big\}\\ \mathbb{P}\text{rob}\big\{\mathbf{X}|\mathbf{Y}=\mathbf{y}\big\} &= \sum_{\Theta}\mathbb{P}\text{rob}\big\{\mathbf{X},\theta|\mathbf{Y}=\mathbf{y}\big\}\end{aligned}$$

and apply a maximum *a posteriori* probability criteria to derive the needed parameter :

$$\begin{aligned}\hat{\theta} &= \text{Arg}\max_{\Theta}\mathbb{P}\text{rob}\big\{\theta|\mathbf{Y}=\mathbf{y}\big\}\\ \hat{\mathbf{X}} &= \text{Arg}\max_{\mathcal{X}}\mathbb{P}\text{rob}\big\{\mathbf{X}|\mathbf{Y}=\mathbf{y}\big\}\end{aligned}$$

One nice property of the Bayesian framework is that it might be extended to deal with cost functions. As explained before, we might define a suitable cost function $\mathcal{C}(Y,\mathcal{M}(\hat{X},\hat{\theta}))$ that will be used to evaluate the quality of the interpretation. To simplify notations, we will position ourself in the continuous situation where $\mathbf{X},\theta$ and $\mathbf{Y}$ are continuous random variables that are related by conditional defined by the *a priori* interpretive model $\mathcal{M}$ as $f(\mathbf{Y}|\mathbf{X},\theta)$ and $g(\mathbf{X}|\theta)$, moreover we will assume an distribution prior $\pi(\theta)$ for the parameters. We suppose that the inference is done by a function $delta(.)$ such that $\hat{\mathbf{X}},\hat{\theta} = \delta(\mathbf{y})$. The overall cost resulting from using the inference rule δ for solving the modelling and interpretation problems, can be derived as :

$$c(\delta) = \int_{\Theta}\int_{\mathcal{X}}\mathcal{C}(\mathbf{y},\mathcal{M}(\delta(\mathbf{y})))f(\mathbf{Y}=y|\mathbf{X},\theta)g(\mathbf{X}|\theta)\pi(\theta)d\mathbf{X}d\theta$$

The best inference rule for the interpretation problem is the rule that minimize this overall cost :

$$\delta^* = \text{Arg}\min_{\delta}c(\delta)$$

Such an estimator is called a Bayes estimator. The rich literature in Bayesian decision theory [17], give such Bayes estimator for specific cost function, *e. g.* for the popular mean square error cost function $\mathcal{C}(x,y) = |x-y|^2$, the Bayes estimator is given by :

$$\delta^*(\mathbf{y}) = \mathbb{E}_{\pi}\big\{\theta|\mathbf{Y}=\mathbf{y}\big\}$$

The previous discussion has shown that the main technical point in Bayesian analysis is how to calculate the involved integral. These integrals occurs for calculating the *a posteriori* probability as well as for deriving the total cost. Numerical integral evaluation have a complete and rich literature, however as we are in a statistical estimation context some specific methods have been designed to deal with integrals coming from Bayesian analysis. The Markov Chain Monte Carlo (MCMC) method is one of the most popular approach [17]. The idea here is to define a Markov chain that will converge to a stationary distribution that will follow the needed *a posteriori* distribution. The Bayesian inference problems might be addressed by MCMC approach.

MCMC approach have been applied to deal with the interpretation as defined in this paper. Examples of this application are [20] that use the Bayesian framework for solving the network tomography problem ; [23] refines this approach by integrating an MMPP traffic into the *a priori* model. [12] apply the Bayesian methodologoy and the MCMC method to another type of network tomography problem where the goal is to identifying lossy links in the interior of the Internet by passively observing the end-to-end performance of existing traffic between a server and its clients.

25.5 Conclusion and perspectives

In this paper we developed a new modelling methodology for analyzing and interpreting QoS as measured by active measurement with the help of an *a priori* constructive model. This approach is original as it starts with observed performance (or QoS) measure and find inputs that had lead to these observation.

This approach needs the introduction of the hidden variable statistical framework. We have described this framework and given some guidelines into the EM algorithm. This framework provides the mean to formalize the approach in the context of the well documented Hidden Markov Model.

Finally, we have illustrated the methodology in the context of the modelling of the loss observed in an internet path. This example shows that the proposed approach can be valuable in the context of interpretation of QoS measured by active measurements.

It is clear that this methodology needs more complete explanation and case studies. It enables to build a calibrated model that provides a good tradeoff between accuracy and complexity.

Bibliographie

[1] Network simulator- ns. http ://www.isi.edu/nsnam.

[2] Sara Alouf, Philippe Nain, and Donald F. Towsley. Inferring network characteristics via moment-based estimators. In *INFOCOM*, pages 1045–1054, 2001.

[3] J. Andren, M. Hilding, and D. Veitch. Understanding end-to-end internet traffic dynamics. In *Proceedings of SIGCOMM' 98*, 1998.

[4] Jeff Bilmes. A gentle tutorial on the EM algorithm including gaussian mixtures and baum-welch. Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, 1997.

[5] J.C. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive fec-based error control for internet telephony. In *Proceedings of IEEE INFOCOM*, pages 1453–1460, NY, March 1999.

[6] R. Caceres, N.G. duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network internal loss characteristics. *IEEE Transactions on Information Theory*, 45(7) :pp. 2462–2480, November 1999.

[7] J. Cao, D. Davis, S. Vander Weil, and B. Yu. Time-Varying Network Tomography. Journal of the American Statistical Association, 2000.

[8] Gilles Celeux, Didier Chauveau, and Jean Diebolt. On stochastic versions of the EM algorithm. Technical Report RR-2514, INRIA, 1995.

[9] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure ? In *INFOCOM*, pages 905–914, 2001.

[10] Fotis Georgatos, Florian Gruber, Daniel Karrenberg, Mark Santcroos, Ana Susanj, Henk Uijterwaal, and René Wilhelm. Providing active measurements as a regular service for isp's. In *Proceeding of PAM 2001 conference*.

[11] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and Diot C. Traffic Matrix Estimation : Existing Techniques and New Directions. In *ACM SIGCOMM*, Pittsburgh, USA, August 2002.

[12] Venkata N. Padmanabhan, Lili Qiu, and Helen J. Wang. Passive network tomography using bayesian inference. In ACM, editor, *Proceedings of IMW 2002*, Marseille, France, 2002.

[13] V. Paxson. *Measurements and Analysis of End-to-End Internet Traffic*. PhD thesis, UC Berkeley, February 1997.

[14] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP performance metrics. *RFC-2330*, May 1998.

[15] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale internet measurement. *IEEE Communications*, 36(8) :48–54, August 1998.

[16] Plato. *The republic*. IndyPublish.com, 2002.

[17] Christian P. Robert. *The Bayesian Choice : From Decision-Theoretic Foundations to Computational Implementation*. Springer, 2nd edition, Janvier 2001.

[18] Kavé Salamatian and Sandrine Vaton. Hidden markov modeling for network communication channels. In *SIGMETRICS 2001*, July 2001.

[19] Augustin Soule, Kavé Salamatian, Richard Emilion, Nina Taft, and Konstantina Papaginiaki. Flow classification by histogram or how to go to safari over interneton stochastic versions of the EM algorithm. Technical report, LIP6-UPMC, 2003.

[20] C. Tebaldi and M. West. Bayesian Inference of Network Traffic Using Link Count Data. Journal of the American Statistical Association, 1998.

[21] Kavé Salamatian Thomas Bugnazet and Bruno Baynat. Cross traffic estimation by loss process analysis. In *Proceedings of 15th ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, July 2002.

[22] Y. Vardi. Estimating Source-Destination Traffic Intensities from Link Data. *Journal of the American Statistical Association*, pages 365–377, 1996.

[23] Sandrine Vaton and Annie Gravey. Iterative bayesian estimation of network traffic matrices in the case of bursty flows. In *Proceedings of IMW 2002*, 2002.

[24] Yannis Viniotis. *Probability and Random Processes*. Mc Graw Hill, 1998.

[25] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the mbone multicast network. In *IEEE Global Internet Conf.,London,UK*, 1996.

[26] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modelling of the temporal dependence in packet loss. In *infocom*, New York, March 1999.

Cinquième partie

Sous Projet 6

Mesure de SLA et tarification

METROPOLIS

Métrologie POur L'Internet et les Services

Projet RNRT Exploratoire

Sous Projet 6
Tarification et SLA

Livrable
Rapport d'avancement

Coordinateur du sous projet : François-Xavier Andreu (GIP Renater).

Auteurs : GIP Renater : François-Xavier Andreu, Emmanuel Da Costa, Gilles Yonnet

LAAS : Philippe Owezarski, Nicolas Larrieu

Groupement d'intérêt public RENATER
Laboratoire d'Analyse et d'Architecture des Systèmes LAAS

Chapitre 26

Introduction

L'Internet est en train de devenir le réseau universel pour tous les types d'informations, du transfert simple de fichiers binaires jusqu'à la transmission de la voix, de la vidéo ou d'informations interactives en temps-réel. L'Internet se doit donc de fournir de nouveaux services adaptés aux applications Internet et aux données qu'elles transmettent. L'Internet doit en fait évoluer d'une offre de service "best effort" unique vers une offre multi-services.

Cela se traduit en particulier par le besoin fort d'offrir aux utilisateurs des qualités de services (QdS) garanties, et pour cela d'être en mesure pour l'opérateur de mesurer en permanence la qualité (ou plutôt les paramètres de qualité) du service qu'il fournit. Cela revient en fait à mesurer les paramètres de QdS ou de trafic qui forment le SLA[1] entre l'opérateur et ses clients, chaque client pouvant naturellement avoir négocié un SLA spécifique. D'autre part, il va de soi que ces mesures peuvent également être exploitées pour la tarification des services fournis par les opérateurs aux utilisateurs. En effet, l'introduction de mécanismes de garantie de QdS différenciés est un des arguments commerciaux pour les opérateurs. Jusqu'à présent il s'avère que les utilisateurs ne sont pas prêts à payer plus que le coût de raccordement pour une connexion "best effort". Par contre, nombre d'opérateurs pensent que des services garantis de niveaux supérieurs pourraient intéresser des clients, même si une surfacturation leur est appliquée. C'est dans cette mouvance que le travail mené dans le sous-projet 6 "SLA et tarification" a été mené sous la conduite de RENATER.

Naturellement, les outils de métrologie qui ont été conçus et mis en place [30] [5] constituent des éléments de choix pour effectuer les travaux de mesure des SLA et de supervision du trafic à des fins de tarification et facturation. Ce rapport ce compose donc de deux parties principales :

- Une première partie qui étudie une solution pour une mesure en continue des paramètres de SLA. Cette partie est essentiellement basée sur des sondes de mesures actives à même de contrôler en permanence les paramètres de QoS des SLA entre les différents routeurs d'un réseau d'opérateur – celui de RENATER en l'occurrence.
- Une seconde partie qui se basera sur les études réalisées dans le cadre du sous-projet 3 et en relation avec l'activité de caractérisation du trafic pour proposer un nouveau modèle de tarification. Cette étude repose sur l'utilisation des mesures passives (avec les sondes DAG

[1] *Service Level Agreement*

notamment), et les conclusions obtenues jusqu'à présent sur les caractéristiques du trafic permettent de dépoussiérer le modèle "smart market" et de l'appliquer au trafic Internet actuel.

Enfin, la conclusion présentera la prospective de ce sous-projet pour sa 3e et dernière année sur les deux aspects que sont la mesure et le contrôle des SLA et les principes de tarification. En particulier, il est apparu que les problèmes de quantités d'informations obtenues par les mesures passives est rédhibitoire pour de la tarification et de la facturation en temps réel. Un des objectif de la dernière année du projet sera donc d'exploiter également les résultats d'échantillonnage du projet, et d'étudier comment les adapter au cas particulier des SLA et de la tarification.

Chapitre 27

Contrôle des SLA

27.1 Introduction

Le GIP RENATER souhaitait proposer à ses utilisateurs différents niveaux de services avec une politique tarifaire modulable en fonction des services fournis. Afin de contrôler ce service, il est nécessaire de déployer sur le réseau une solution de sondes de mesures actives. Les solutions doivent cependant être à même de mesurer correctement les métriques définies. Ce chapitre spécifie dans un premier temps les besoins d'un tel système, puis présente les problèmatiques auxquelles il est confronté pour finir avec une description des solutions testées.

27.2 Spécification des besoins

Cette partie présente les caractéristiques du réseau RENATER et les contraintes auxquelles devront répondre les futurs systèmes de métrologie active.

27.2.1 Caractéristiques du réseau RENATER

Le réseau RENATER est constitué de liaisons haut débit (2,5 Gb/s) reliant différents points de présence auxquels sont reliés les réseaux de collecte régionaux. Plusieurs caractéristiques découlant de la conception de ce réseau doivent être prises en compte dans le cadre du contrôle des SLAs :

- Les temps de trajet dans RENATER sont très faibles (de l'ordre de quelques ms - figure 27.1) comparés aux temps de trajet de bout en bout.
- Le réseau est peu chargé en moyenne.
- Le réseau est très stable, la gigue excédant rarement la milliseconde.

Ce réseau étant utilisé par la communauté de l'éducation et de la recherche, les applications y circulant sont diverses et couvrent de nombreuses catégories comme le courrier électronique, l'Internet, mais aussi la visioconférence ou le calcul distribué.

La figure 27.1 présente les temps d'aller retour depuis Paris, obtenu par Ping.

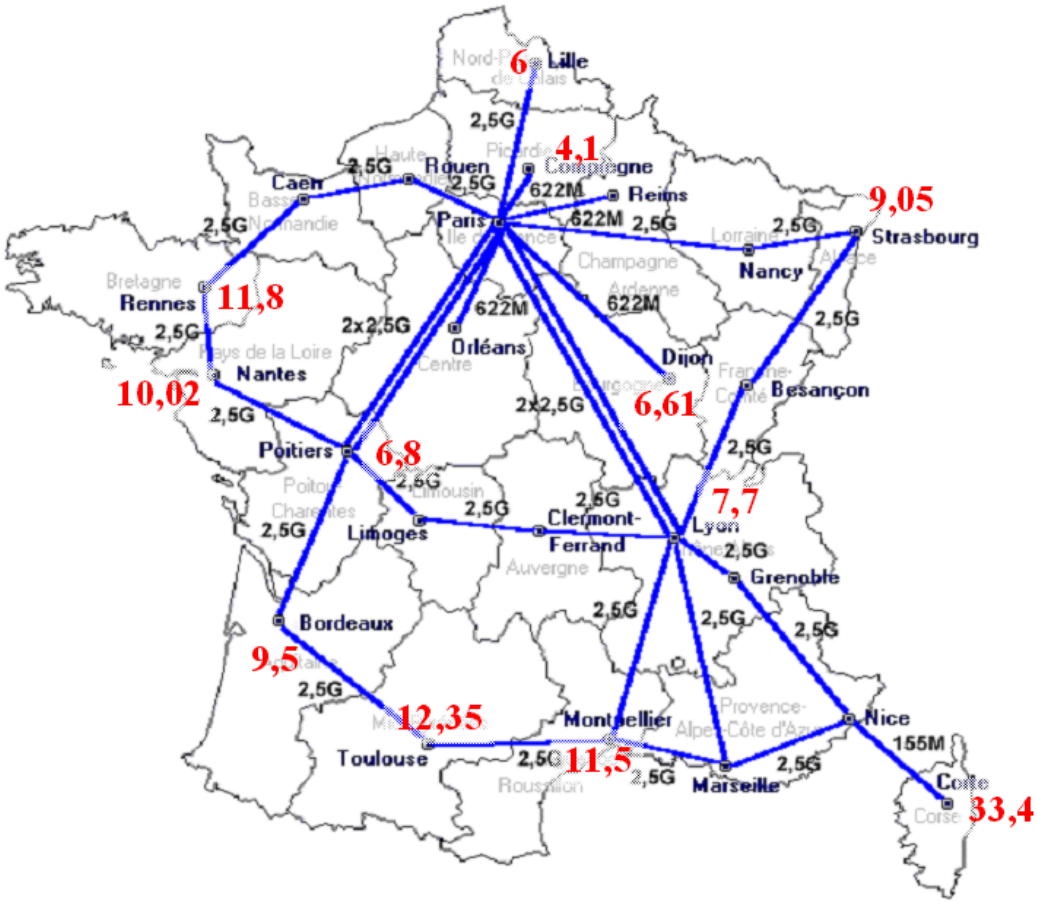


FIG. 27.1 – Temps d'aller retour depuis Paris en ms

Une autre caractéristique importante pour la métrologie active est la présence de classes de service. Pour l'heure le réseau ne différencie pas les paquets IP, mais le service est opérationel est sera actif au début de l'année 2004 sur RENATER [44]. Tous les paquets IP traversant le backbone subissent donc actuellement les mêmes conditions indépendamment du protocole de transport utilisé, des ports de l'application, etc. ... L'injection d'un quelconque trafic de test simulera donc exactement les conditions subies par les utilisateurs, à condition qu'il s'agisse de paquets IP. Les seuls paramètres pouvant influer sur le temps de trajet sont donc la taille des paquets et, à l'avenir, la classe de service.

On soulignera cependant que ceci est bien différent dans les réseaux des sites utilisateurs où la mise en place de filtrage (firewall) ou de proxies peut induire des différences de traitement et donc de délais entre les différents types de trafics.

Le réseau RENATER transportant également de l'IPV6 et du trafic multicast, l'adaptation de la solution adoptée à ces spécificités pourra être envisagée afin de caractériser l'intégralité des applications du réseau.

27.2.2 Fonctionnalités d'un système de métrologie active

Plusieurs niveaux d'exigences peuvent être définis, selon les moyens qui sont mis à disposition et la politique souhaitée de la part de l'opérateur.

Le plus élémentaire consiste à produire des ordres de grandeur des temps de trajets sur le réseau alors que le plus élaboré doit permettre la surveillance proactive du réseau à partir de mesures très précises et en temps réels sur le réseau. Dans tous les cas on souhaite déterminer une matrice, plus ou moins complète, des temps de trajet entre les points de présence.

La mesure de l'impact de Diffserv peut également être envisagée. On notera cependant que les différences de performance entre les trafics de chacune des classes ne sont sensibles que lorsque le réseau est très fortement chargé [43], ce qui n'est pas le cas sur le réseau RENATER.

Plusieurs paramètres tels que la synchronisation, les métriques considérées, la localisation des points de mesure ou le trafic généré peuvent être ajustés. Les différentes possibilités pour chacun de ses paramètres sont décrites dans les parties suivantes.

Métriques considérées

Trois classes de mesures peuvent être mises en
oeuvre sur le réseau :

- Les mesures bidirectionnelles. A partir d'un outil simple tel que Ping, plusieurs caractéristiques du réseau peuvent être déterminées : délai, gigue et pertes lors d'aller-retours de paquets. Le principal inconvénient de cette classe est qu'elle ne permet pas de distinguer les trajets aller et retour. Elle ne caractérise donc pas véritablement la qualité vue par l'utilisateur. Elle présente cependant le grand avantage de ne pas nécessiter de synchronisation particulière des différentes stations de mesure.
- Les mesures unidirectionnelles. La mise en place de stations disposant d'une synchronisation fiable et précise permet la collecte de mesures unidirectionnelles du délai, de la gigue, des taux de pertes et du déséquencement, métriques représentatives de la qualité perçue par l'utilisateur.

– Les mesures “mixtes”. Cette solution constitue un compromis entre les deux classes décrites ci-dessus. La synchronisation précise des équipements pouvant constituer un problème, on peut s’affranchir de celle-ci en abandonnant la mesure du délai unidirectionnel. On peut en effet aisément et même sans synchronisation, à condition de disposer d’une horloge stable à court terme sur les stations de mesure, mesurer gigue, pertes et déséquencement unidirectionnels. En adjoignant la mesure du délai bidirectionnel on peut ainsi obtenir un compromis quant aux données collectées.

L’élément déterminant dans le choix entre ces trois possibilités est la synchronisation qui sera établie.

Synchronisation et précision

Comme décrite précédemment, la mise en place de mesures unidirectionnelles nécessite la mise en place d’une synchronisation des stations. Dans le cas de mesures bidirectionnelles la synchronisation sert juste à dater les mesures et ne requiert donc pas une grande précision.

Trois modes de synchronisation sont possibles :

– A partir d’un GPS. La synchronisation de chacune des stations de mesures sur un boîtier GPS permet d’obtenir une précision de l’ordre de 10 microsecondes. Malgré le prix modéré d’un tel dispositif (de l’ordre de 1500 euros par point de mesure), le déploiement à grande échelle peut néanmoins constituer un réel problème. Les antennes doivent en effet disposer d’une bonne visibilité et donc être placées à l’extérieur et de préférence en hauteur.
– A partir de serveurs NTP. La mise en place de serveurs NTP de “Strate 1” en un certain nombre de points ou l’utilisation de serveurs NTP de “Strate 1 ou 2” existants peut également être envisagée. La précision atteinte est nécessairement plus faible que celle obtenue par GPS.
– A partir d’une solution mixte GPS + NTP. Le placement de stations synchronisées sur GPS en un certain nombre de nœuds du réseau et la distribution de cette synchronisation par NTP aux nœuds adjacents est un compromis intéressant.

Afin de déterminer l’efficacité des différentes solutions, des tests préliminaires ont été réalisés, les résultats sont présentés dans la section 27.4.2. Ces tests ont permis à la fois de déterminer la précision relative des trois méthodes et l’ordre de grandeur des temps de traversée du réseau RENATER et de comparer les outils de métrologie utilisés.

La réalisation de tests préliminaires permettra de plus d’estimer la facilité d’installation, de configuration et d’usage ainsi que la charge induite sur le réseau et les équipements de mesure.

Un autre point capital est qu’aucun facteur extérieur (charge de la plateforme, fonctionnement du système d’exploitation...) ne doit perturber la mesure. Dans une optique de surveillance du réseau, il ne faut en effet pas qu’une alerte sur dépassement de seuil soit levée à cause de l’outil de mesure ou qu’au contraire une dégradation du réseau soit négligée.

Il n’est donc pas possible non plus, dans un tel cas, de négliger des mesures jugées aberrantes en les attribuant à un quelconque facteur extérieur.

27.2.3 Localisation des points de mesure

La localisation des points de mesure peut être envisagée de plusieurs façons, en fonction des moyens disponibles :

– Une station de mesure dans chaque point de présence. Si l'on considère que l'objectif est de garantir la qualité de service offerte par le backbone RENATER dans tous les cas de figures, les mesures devront avoir lieu entre tous les points d'entrée et de sortie du backbone. Ceci pourra être réalisé uniquement en implantant dans chaque point de présence une station de mesure.
– Une station de mesure en des points particuliers. Cette alternative consiste à exploiter la structure du réseau RENATER en implantant des stations dans les endroits les plus intéressants, c'est à dire voyant le plus de trafic (Paris, Lyon, Poitiers...). On perd cependant la possibilité de fournir à l'utilisateur des statistiques pleinement représentatives, certains chemins n'étant alors pas mesurés.
– Dans les deux cas précédents on peut envisager en plus la présence de stations de mesures au plus près des utilisateurs.

27.2.4 Sécurité

En théorie, un système de mesure déployé dans de nombreux sites distants doit respecter des règles de sécurité, par exemple :

– Le système devant être constitué de plusieurs plateformes, une communication chiffrée est souhaitable.
– L'accès à chacune des plateformes doit requérir une authentification (mot de passe, clé RSA ...).
– Les paquets sondes doivent pouvoir être authentifiés afin d'éviter leur contrefaçon en vue de fausser les mesures.
– Des mécanismes de protection face à un déni de service sur les plateformes de mesure et de présentation sont necessaires.

Comme indiqué dans la suite du document, les différentes solutions testés sont très disparates du point de vue des fonctionnalités relevant de la sécurité.

27.2.5 Trafic généré

Le trafic généré par les mesures et l'éventuelle exportation des données peuvent modifier le comportement du réseau s'il n'est pas contrôlé.

L'émission des paquets sondes doit de plus être représentative du trafic réel présent sur le réseau afin de simuler au plus juste les contraintes subies par ce trafic. Les tailles des paquets sondes et les intervalles d'émission doivent donc prendre en compte cette exigence. Il semble par exemple souhaitable d'émettre des paquets de différentes tailles, ceux-ci ne subissant pas les mêmes conditions de transports.

Des paquets sondes doivent être envoyés dans chacune des classes de service définies pour permettre le contrôle de la différenciation de service et produire les données nécessaires à la vérification du SLA.

27.2.6 Récupération et présentation des données

La récupération des données doit se faire le plus fréquemment possible, idéalement après chaque mesure, afin de permettre une présentation et une éventuelle réaction en temps réel.

Ces données doivent par la suite pouvoir être traitées et présentées à l'utilisateur en temps réel à travers une interface graphique, web par exemple.

Il est également intéressant de disposer d'un système d'alarme, envoyant par exemple automatiquement un mail lorsque certains seuils sont dépassés.

27.3 Etude des problématiques liées aux mesures

Ce chapitre détaille les études et expérimentations menées sur les différents éléments nécessaires à la mise en place d'un système de mesures actives répondant aux besoins énoncés. Les différents modes de synchronisation sont étudiés, ainsi que le fonctionnement et la fiabilité des différents outils de mesure et l'intérêt de systèmes temps réels dans le contexte de ce projet.

27.3.1 Synchronisation

La synchronisation étant une problématique centrale de notre projet, de nombreux tests ont été menés sur le sujet. Il s'agissait de quantifier la précision produite par différents modes de synchronisation dans un contexte proche de celui du réseau RENATER. On a ainsi cherché à caractériser la stabilité de l'horloge logicielle, c'est-à-dire la stabilité et la précision de NTP en environnement LAN et WAN. Au cours de l'année 2003 la fonctionnalité synchronisation avec GPS n'a pas pu être testée. Les observations sur son efficacité se sont faites à partir des expériences acquises en la matière par divers projets de métrologie. Toutefois des tests sont prévus lors de la dernière année du projet (cf. chapitre 29).

Cette partie décrit les différentes solutions existantes, les expérimentations réalisées et les résultats obtenus.

Description des solutions de synchronisation

Horloge logicielle La référence de temps couramment utilisée dans les ordinateurs est l'horloge de l'unité centrale. Il convient donc de connaître son fonctionnement et ses limitations afin d'appréhender les différents problèmes liés à la synchronisation.

L'horloge logicielle du PC, sur les systèmes de type UNIX, est mise à jour par une interruption déclenchée sur réception d'un top dérivé de l'oscillateur à quartz de la carte mère, intervenant toutes les 10 ms par défaut [27]. La fréquence d'interruption peut être modifiée (constante HZ du noyau), afin d'obtenir une meilleure résolution. En accédant au nombre de cycles CPU intervenus depuis la dernière interruption, stocké dans le registre TSC (Time Stamp Counter), les programmes peuvent interpoler l'heure entre deux interruptions [37]. On accroît alors de manière artificielle la résolution de l'horloge logicielle. La structure standard de temps UNIX contient ainsi des valeurs en microsecondes.

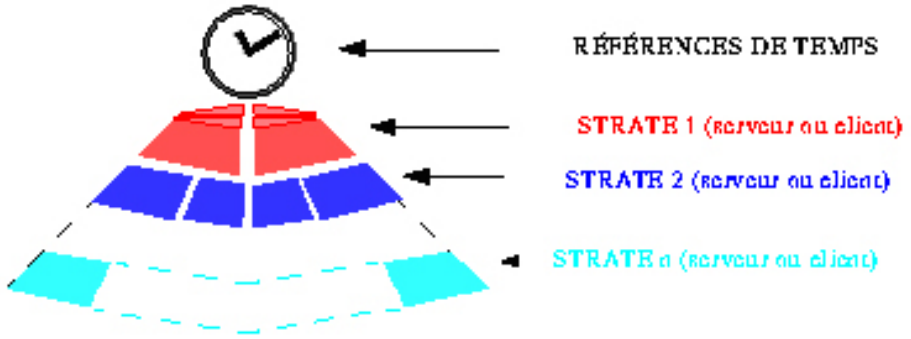


FIG. 27.2 – Stucture arborescente de NTP (source : Comité Réseau des Universités)

La fréquence à laquelle s'incrémente le temps n'est cependant pas parfaitement stable. Une erreur de quelques PPM (Parties Par Million) sur la fréquence, peut ainsi mener à un décalage temporel allant de quelques millisecondes jusqu'à quelques secondes par jour. Les expérimentations sur la stabilité de l'horloge logicielle seront présentées dans la deuxième partie de cette section (27.3.1).

Afin de garantir une stabilité à long terme, on doit donc se synchroniser à partir d'une horloge de référence fiable, éventuellement à travers un réseau : c'est le rôle du protocole NTP.

NTP Le protocole NTP, développé par David H. Mills, permet la synchronisation de machines à travers un réseau. Ce protocole, dans sa version 3, est décrit dans la RFC 1305.

Ce protocole, à l'algorithme complexe, estime le décalage de l'heure de la machine avec l'heure absolue (UTC : Universal Time Coordinated), à partir d'une référence et des temps estimés de trajet sur le réseau, et le corrige en ajustant progressivement la fréquence.

Pour calculer ce décalage (offset), le serveur NTP va périodiquement interroger, selon un modèle requête-réponse, le(s) serveur(s) sur lesquels il se synchronise[1]. La période entre les interrogations est comprise entre 64 et 1024 secondes et s'ajuste selon le décalage observé. Plus ce décalage est grand, plus l'interrogation est fréquente.

NTP est fondé sur une structure arborescente, chaque serveur de Strate n (n>1) se synchronisant sur un ou plusieurs serveurs de Strate n-1, les serveurs de strate 1 se synchronisant directement à partir d'une horloge de référence (horloge atomique ou GPS) (figure 27.2). La précision est donc de moins en moins bonne lorsque le numéro de Strate augmente.

La principale limitation du protocole NTP réside dans le fait qu'il ne peut pas connaître les temps unidirectionnels de trajet dans le réseau, mais qu'il les estime à partir des RTT. L'estimation du décalage avec l'heure absolue est donc légèrement faussée, d'autant plus lorsque l'aller-retour est asymétrique.

[1] NTP utilise le port 123 en UDP et TCP

La précision obtenue dépend donc énormément du réseau. Nos expérimentations, exposées dans la partie 27.4.2, ont donc eu pour but de vérifier les précisions possibles en environnement LAN et WAN et de les comparer aux valeurs présentées dans la littérature (1 ms en LAN, 50 ms en WAN). Il est en effet important de connaître les possibilités de NTP afin de savoir si un GPS s'avère indispensable.

GPS La mesure des temps sur un réseau requerrant une grande précision, nombre d'infrastructures de métrologie ont recours au GPS pour la synchronisation. Le positionnement précis fourni par le système GPS nécessite en effet une synchronisation temporelle très précise. Les signaux obtenus à partir d'un GPS peuvent être convertis par un boîtier intermédiaire pour être utilisés comme horloge de référence NTP autorisant une précision de 10 microsecondes. Le modèle Acutime 2000 Synchronisation Kit de Trimble est utilisé dans l'expérience décrite dans 27.4.2 et coûte environ 1000 euros HT.

Certains boîtiers proposent également un signal PPS (Pulse Per Second), qui utilisé par l'intermédiaire d'une API particulière, permet une synchronisation plus efficace.

Expérimentations

Les machines utilisées dans les tests décrits dans cette partie sont des PC ayant un processeur pentium II ou supérieur, une carte d'interface Ethernet 100 Mbit/s et fonctionnant sous freeBSD. Un minimum de services sont présents sur les machines afin de ne pas perturber les mesures.

Les mesures ont été réalisées avec Saturne (cf. 27.4.2), envoyant un paquet UDP de 500 octets toutes les minutes, et avec Ping puis présentées en temps réel sur un serveur web Apache au moyen de l'outil MRTG. A noter que ce dernier présente les statistiques moyennées des mesures. Dans chacune des expériences présentées la mesure du RTT par Ping permet de mettre en évidence l'instabilité du réseau. On peut ainsi attribuer l'instabilité du délai observé soit au réseau soit au protocole NTP.

Enfin, dans les cas où la synchronisation d'une station a lieu sur des serveurs NTP strate 2, ceux ci sont choisis proches géographiquement de la station afin d'avoir la meilleure précision possible.

Pour chacun des graphes montrés sur les pages suivantes l'unité est la microseconde.

Stabilité de l'horloge logicielle Afin de mesurer la stabilité de l'horloge logicielle nous avons mesuré le délai unidirectionnel et le RTT entre deux machines séparées uniquement par un switch.

La machine émettrice était synchronisée sur 5 serveurs NTP de Strate 2, ce qui garantit une bonne stabilité à long terme. La synchronisation de la station réceptrice est réalisée sur les mêmes cinq serveurs NTP durant les neuf premiers jours représentés sur le graphe puis sur sa propre horloge locale lors de la fin du test.

Les résultats sont représentés sur la figure 27.3, le délai unidirectionnel étant en bleu clair, le Ping en bleu foncé. La courbe rose représente le maximum de ping enregistré, la courbe verte représente le maximum de délai unidirectionnel.

A noter que les mesures de délai unidirectionnel ont commencé le vendredi et que la synchronisation de la machine réceptrice est

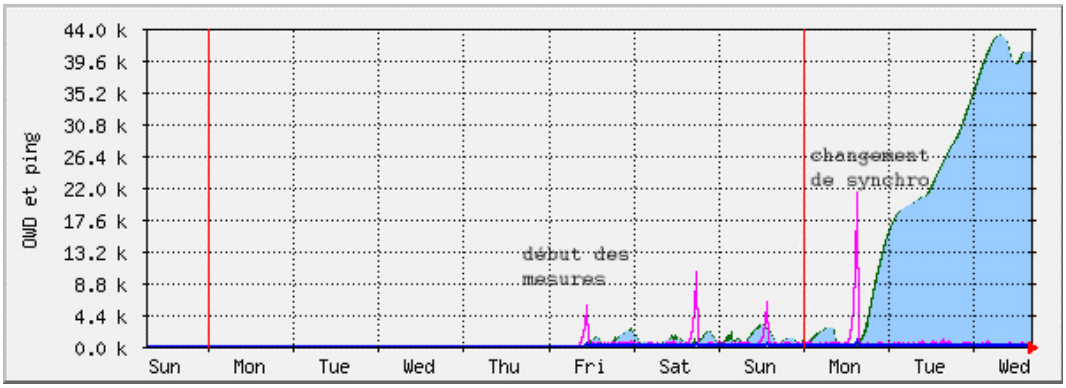


FIG. 27.3 – Stabilité de l'horloge logicielle. Le "OWD" est représenté en bleu clair (mesuré par Saturne 27.4.2, le bleu foncé représente le "RTT" mesuré par Ping et la courbe rose représente le maximum du Ping. L'échelle est en microsecondes.

On observe une importante dérive de l'horloge locale à partir du moment où la synchronisation sur les serveurs NTP est abandonnée au profit de l'horloge locale. Le réseau étant très stable, cette dérive ne peut être attribuée qu'à l'instabilité de l'horloge.

Cette dérive est faible, de l'ordre de quelques millisecondes par jour, mais irrégulière. On observe en effet des ruptures de pentes selon les heures de la journée. Ceci peut s'expliquer par la sensibilité de l'oscillateur à la température de la pièce, relativement variable au cours d'une journée[2].

La dérive de l'horloge peut cependant être beaucoup plus importante. Il est ainsi communément admis que celle-ci peut atteindre plusieurs secondes par jour et nous avons par exemple constaté une dérive de 600ms en 15h à l'occasion d'une autre expérience.

Cette dérive, correspondant à environ une milliseconde par minute, impose par conséquent une correction périodique de la fréquence. C'est ce que réalise NTP.

Précision de NTP en LAN L'expérience consistait ici à mesurer à l'aide des outil de mesure Saturne (cf. 27.4.2) et ping le délai unidirectionnel entre deux machines uniquement séparées par un switch, la machine émettrice se synchronisant sur la machine réceptrice. La configuration des machines est la même que lors du test précédent (27.3.1). L'offset, c'est à dire le décalage avec le temps de référence estimé par l'algorithme NTP, était également collecté afin de connaître l'estimation par NTP de la précision.

La figure 27.4 représente le délai unidirectionnel (en bleu clair) et le Ping (en bleu foncé) dans le cas de la synchronisation de l'émetteur sur sa propre horloge locale. Le palier final correspond à l'arrêt des mesures.

On observe d'importantes variations du délai mesuré, ces variations ne pouvant être attribuées au réseau puisque le RTT (Ping) est stable. Outre les pics, dont nous traiterons dans la partie 27.4, on constate en effet des fluctuations lentes de l'ordre de 200 microsecondes s'expliquant par les cor-

[2]Précision d'une horloge en fonction de la température : http ://www.beaglesoft.com/clcaspecs.htm et http ://pdfserv.maxim-ic.com/arpdf/Design/RTC.pdf

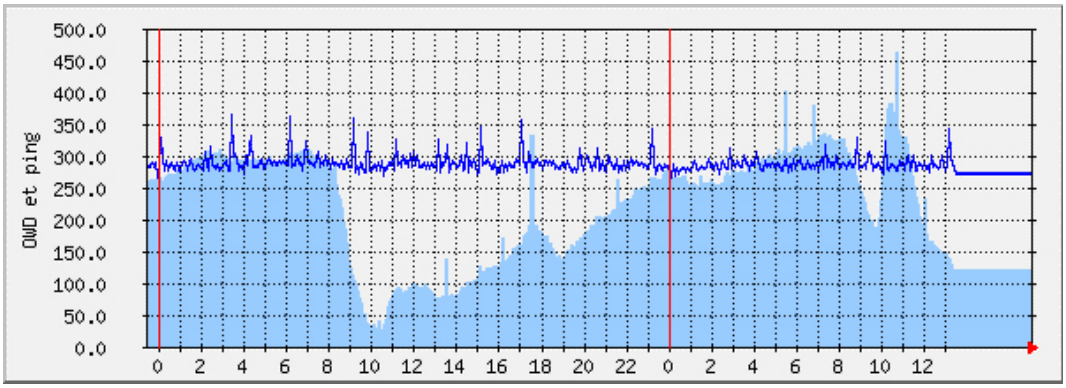


FIG. 27.4 – Mesures du délai en LAN : “OWD” en bleu clair et “RTT” en bleu foncé (échelle en microsecondes)

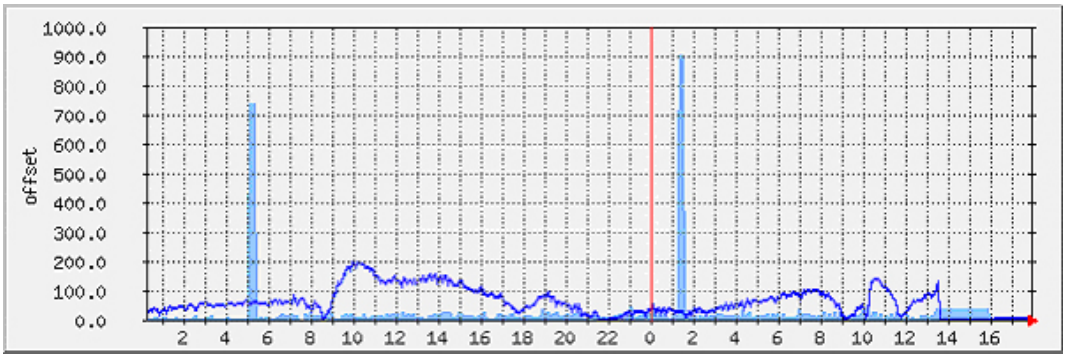


FIG. 27.5 – Offset (bleu foncé) et gigue (bleu clair) en LAN, échelle en microsecondes

rections faites par NTP. Ceci se trouve confirmé par la mesure de l’offset (en bleu foncé), représenté sur la figure 27.5. La courbe bleu clair décrit la gigue estimée par NTP.

L’analyse des mesures (non présentée ici) faites lorsque l’émetteur est synchronisé sur 5 serveurs fait apparaître des fluctuations légèrement plus importantes. Ceci peut s’expliquer par le fait que l’émetteur subit alors des corrections induisant sur le récepteur, qui se synchronise sur lui, une perturbation. Cette sensibilité de NTP à la stabilité de sa référence et du réseau est illustrée sur la figure 27.6 (la gigue étant représentée en bleu foncé et l’offset en bleu clair). A l’origine de cette expérience, les deux machines A et B sont reliées directement par un câble croisé. Durant toute l’expérience A est synchronisée sur sa propre horloge locale et B est synchronisée sur A via NTP. Outre l’important temps de convergence de l’algorithme au début de la synchronisation, on constate que chaque changement de condition a une forte incidence sur la synchronisation de B par rapport à A. Ainsi :

– Une variation brusque de température (due à l’ouverture ou à la fermeture d’une fenêtre dans le local contenant A) change le comportement de son horloge locale (cf. 27.3.1). Ce changement induit une forte augmentation de l’offset, traduisant la désynchronisation de B par rapport à A.

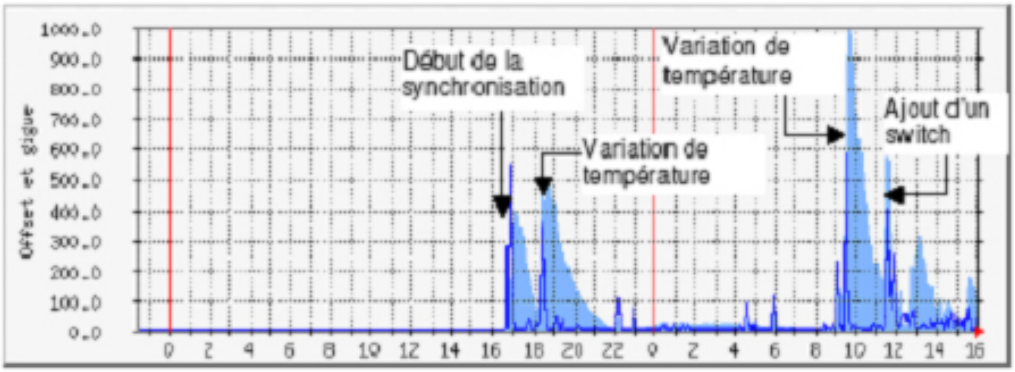


FIG. 27.6 – Offset (bleu clair) et gigue (bleu foncé) en LAN avec changements de conditions, échelle en microsecondes

– L'ajout d'un switch entre les deux machines, ajoutant brutalement 100 microsecondes de délais, donne lieu à une augmentation de l'offset.

Etant donné le décalage entre la valeur réelle du délai (approximativement la moitié du RTT) et sa valeur observée sur la figure 27.4, la précision atteignable avec NTP dans un contexte optimal peut être finalement estimée à environ 250 microsecondes, la référence devant évidemment être elle même précise et stable et le réseau stable.

Précision de NTP en WAN Les mesures ont ici été faites entre une station située dans les locaux de l'ENST Bretagne à Rennes et synchronisée directement sur GPS et une station située dans les locaux du GIP RENATER et synchronisée de différentes manières. Une partie non négligeable du trajet est donc faite sur le réseau de collecte et le réseau local de l'ENST Bretagne.

Les résultats obtenus (cf. Figure 27.7) lorsque la synchronisation de la station réceptrice se fait sur 5 serveurs NTP de Strate 2 donnent une idée de la précision de ce mode de synchronisation (à noter que des redémarrages de la station ont perturbé les mesures le dernier jour). On constate en effet d'importantes fluctuations (de l'ordre de 3 ms) non corrélées avec le Ping. Une variation du temps de trajet retour compensant celle du trajet aller étant peu probable, ces variations sont dues à NTP.

Il est également possible que s'ajoute à ces variations un décalage constant par rapport à la valeur réelle. Par conséquent on peut évaluer la précision dans ce contexte à environ 4 ms.

Les résultats obtenus lorsque la machine réceptrice se synchronise sur la machine émettrice présentent des variations de plus fortes amplitudes (de l'ordre de 4 ms), et ce malgré la stabilité et la précision de l'horloge sur laquelle on se synchronise. Ceci s'explique par des variations de grande amplitude de l'offset NTP calculé sur la machine réceptrice (cf. 27.8), due à une relative instabilité du réseau.

Il apparaît donc que le réseau traversé à une grande importance dans la qualité de la synchronisation obtenue.

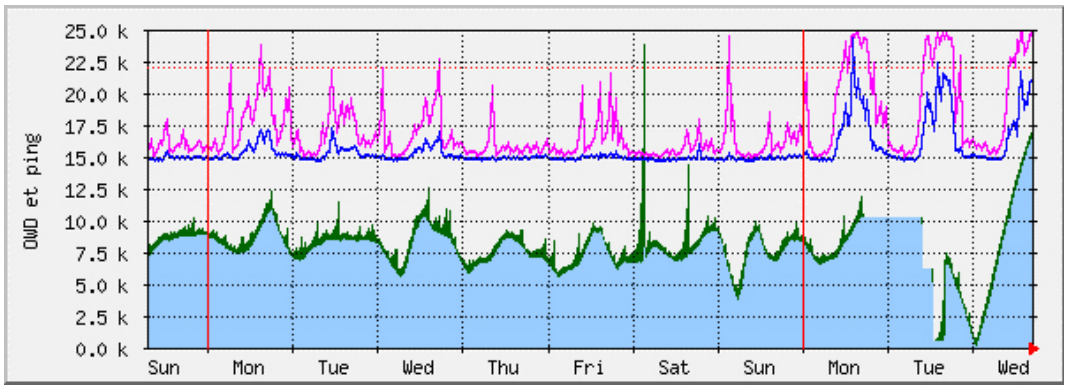


FIG. 27.7 – “OWD” (courbe pleine) et Ping (en bleu foncé, maximum en rose) entre Rennes et Paris avec synchronisation NTP Strate 2 (microsecondes)

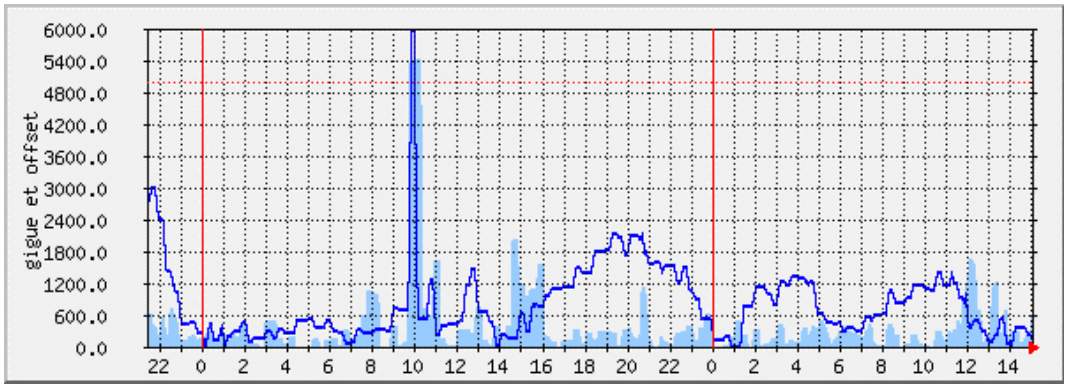


FIG. 27.8 – Offset (courbe bleu foncée) et gigue (histogramme bleu clair), de la machine réceptrice, en microsecondes

La précision obtenue par NTP dans ce contexte est finalement, au vu des graphes précédents, de l'ordre de 3 à 4 ms.

Conclusion

Le réseau RENATER étant un WAN très performant, on peut s'attendre à une précision intermédiaire entre celle obtenue en LAN et celle obtenue en WAN, c'est à dire de l'ordre de la milliseconde. Certaines études se montrent cependant plus optimistes, évoquant une précision de 250 microsecondes. C'est par exemple l'estimation faite lors des mesures par Internet2 sur le réseau Abilene aux Etats Unis [20] et dans l'étude décrite dans [42].

La précision de NTP dépend cependant énormément de la stabilité de la source et du réseau. Si l'on peut effectivement obtenir une précision très bonne lorsque les chemins sont symétriques et très stables, une variation quelconque du délai dégrade durablement la précision, de l'ordre de grandeur de la variation. Cette dégradation est de plus d'autant plus forte et durable que la variation est brusque. Si l'objectif est justement de mesurer des variations pouvant être fortes on ne peut donc pas envisager de se synchroniser par NTP à travers un réseau.

Le GPS apparaît donc comme la seule solution permettant la mesure des délais de manière suffisamment fiable et précise. Bien que son coût soit raisonnable (de l'ordre de 1000 euros HT l'unité), l'installation et la maintenance sur un grand nombre de sites constituent cependant un problème important. Elles sont en effet très dépendantes des lieux dans lequel le GPS doit être implanté. Ainsi il faudra localement prendre en compte la structure des bâtiments.

27.4 Outils de mesure

Différents outils et infrastructures peuvent répondre aux besoins décrits dans la partie 27.2. Cette partie expose les différentes observations résultant de l'utilisation et de l'analyse des outils SAA, Saturne, Rude/Crude, RIPE TTM, NIMI et Ping. Les résultats des diverses expérimentations visant à évaluer leur précision sont ensuite commentés.

27.4.1 Analyse des outils

La principale différence entre ces outils réside dans la façon dont les envois et réceptions des paquets sondes sont faits.

Pour l'émission on distingue en effet trois techniques : estampillage avant passage des données au tampon d'émission du système ("Socket") (Rude, RIPE TTM), estampillage au niveau kernel juste avant le départ sur le réseau (Saturne) (après passage des données du tampon d'émission système à la couche inférieure), envoi et estampillage réalisé de manière matérielle (solution QOSmetrix).

Pour la réception des paquets sondes trois méthodes existent : réception des données au niveau du tampon entre la couche transport et applicative et récupération de l'estampille (Crude), capture des paquets entrant sans utiliser la pile réseau traditionnelle et récupération de l'estampille (RIPE TTM, Saturne), traitement matériel (QOSmetrix 27.4.2).

	SAA	RIPE TTM	Saturne	Rude/Crude	NIMI	QOSMetrix
Matériel	routeur	boîtier (PC/FreeBSD)	PC/FreeBSD	PC(Linux)	PC(divers OS)	boîtier
GPS intégré	non	oui	non	non	non	Selon modèle
Sécurité	Authentification sondes par MD5 possible	Aucune	Aucune	Paquets numérotés	Chiffrement	Intégrée
Réception des paquets sondes	Non précisée	Capture (pcap) + socket	Capture (bpf)	Socket	Dépend de l'outil utilisé	matérielle
Estampillage	Non précisé	Avant envoi sur socket	Kernel	Avant envoi sur socket	Dépend de l'outil utilisé	matériel
Paramétrage du DSCP des paquets sondes	oui	non	oui	oui	Dépend de l'outil utilisé	oui
Collecte	Distante, par snmp	Logs locaux envoyés au site central quotidiennement	Logs locaux, envoi au site central par rpc de chaque mesure	Logs locaux	Logs locaux	Vers site central en temps réel
Présentation des résultats	Non intégrée	Très complète, sur serveur web local et central (à Amsterdam)	Non intégrée	Non intégrée	Non intégrée	Site web central
Modification/ adaptation de l'outil	Impossible	Possible (mais nécessite demande à RIPE)	Possible	Possible	Possible (mais code source imposant)	Impossible
Licence	commerciale	commerciale	non définie actuellement	GPL	GPL	commerciale

FIG. 27.9 – Caractéristiques des outils utilisés

L'objectif étant de mesurer le "temps fil"[3], la précision que l'on peut attendre de systèmes fonctionnant selon le principe de RIPE TTM est donc plus faible que celle de systèmes type Saturne, elle-même plus faible que celle de systèmes comme celui proposé par QOSmetrix.

Les systèmes d'exploitation type Unix n'étant pas de véritable systèmes temps réel, il faut mesurer l'impact éventuel de ce comportement sur les mesures. Des retards artificiels dus à des interruptions entre les instructions peuvent en effet avoir lieu. Si cet impact s'avère important l'utilisation de solutions temps réel devra être envisagée. Les caractéristiques des différents outils sont résumées dans le Tableau 4-1. Les outils pour lesquels la présentation des résultats est dite « non intégrée » nécessitent la mise en place d'un frontal de présentation des données, par exemple un serveur web avec MRTG ou rrdtool. RIPE TTM et QOSMetrix proposent quant à eux des frontaux de présentation très complets qui présentent cependant l'inconvénient de ne pas pouvoir être fondamentalement modifiés.

27.4.2 Expérimentations

Cette partie décrit la mise en œuvre des mesures et les résultats obtenus avec les différents outils.

[3]Le délai unidirectionnel (One Way Delay - RFC 2679) est le temps mis par un paquet pour aller d'une machine à une autre. Ce temps ne doit pas prendre en compte le temps passé dans les machines émettrices et réceptrices mais uniquement le trajet effectué. Le RFC 2679 introduit en cette fin le terme de "temps-fil" (wire time).

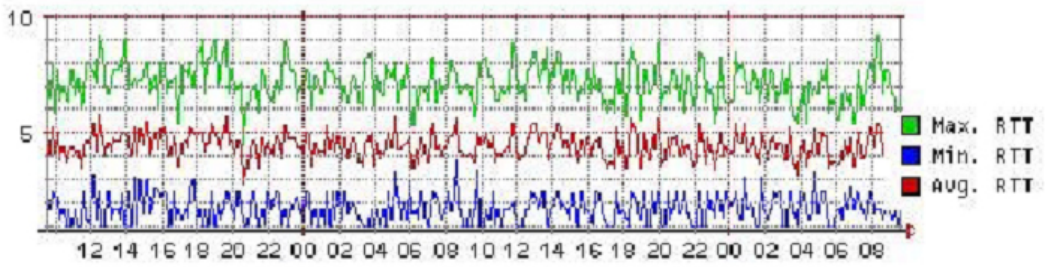


FIG. 27.10 – RTT Compiègne-Nancy en ms

Cisco SAA

Cisco IOS[4] Service Assurance Agent [6] est un outil de mesure active intégré au système d'exploitation IOS (à partir de sa version 11.2). La configuration s'effectue à partir de IOS ou par SNMP. Les résultats sont disponibles par SNMP et des alertes de violation des SLA peuvent être envoyées par "Trap SNMP".

Outre les mesures par UDP de RTT, délai unidirectionnel et gigue, SAA mesure les temps de réponse ICMP saut par saut et la gigue associée, les performances des services (comme la résolution des noms, les délais de connexion TCP et de transaction http), les pertes de paquets, etc. Les mesures des RTT, délais, gigue et pertes sont réalisées par l'envoi de trains de paquets UDP de 10 octets espacés de 10 ms.

SAA permet également de définir des alarmes sur dépassement des seuils et des traps SNMP. Plusieurs campagnes de mesures ont été menées par le Centre Inter-Universitaire de Ressources Informatiques de Lorraine, entre différents routeurs du réseau RENATER. Les figures 27.10 et 27.11 présentent respectivement les RTT et gigue obtenus sur une journée (les deux graphes correspondent à la même période).

On constate un très forte disparité des résultats obtenus. Celle-ci ne peut pas s'expliquer par d'éventuelles variations sur le réseau. Leur régularité tend à prouver qu'il s'agit d'une erreur systématique de SAA.

Cette hypothèse est confirmée par la courbe de gigue au cours de la journée (cf. figure 27.11). On observe en effet des valeurs de gigue très importantes et réparties de manière régulière.

SAA n'est donc pas apte à effectuer des mesures dans le contexte du réseau RENATER, sa précision étant de l'ordre de 10ms.

Saturne

Saturne est une plateforme développée à l'ENST Bretagne [7]. Celle-ci utilise des stations Free BSD. Elle permet l'estampillage des paquets selon un comportement simulant un routeur de périphé-

[4]Internetworking Operating System est le système d'exploitation propriétaire cisco, utilisé dans ses équipements

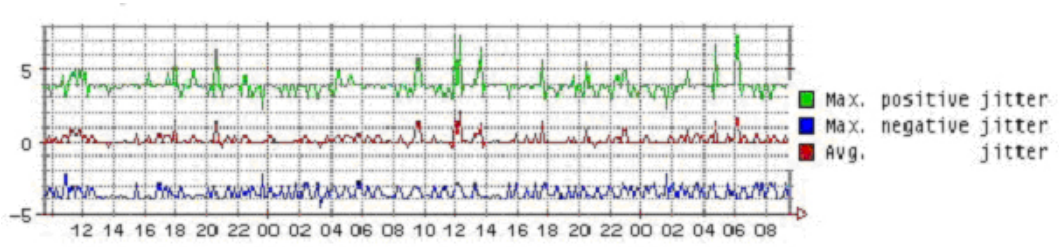


FIG. 27.11 – Gigue Compiègne-Nancy en ms

rie de réseau implémentant DiffServ. Les résultats sont envoyés par la station réceptrice par RPC[5] à un serveur et stockés dans une base de données rrdtool [46].

L'installation de Saturne nécessite deux recompilations et installations du noyau. Il faut en effet ajouter au noyau le module Alt-q puis Adserv. La mise en place des mesures par la suite est relativement simple. Elle implique la modification d'un fichier de configuration et du programme de capture puis le lancement des processus d'envoi de paquet et d'estampillage sur la machine émettrice et de capture sur la machine réceptrice.

Le programme de capture peut être adapté, pour envoyer les résultats à un serveur central par exemple. Dans notre cas la présentation des résultats se faisait par MRTG, invoquant un script allant chercher les dernières mesures dans le fichier local de résultat. A noter qu'un remaniement du code et la création d'une interface java facilitant l'usage de Saturne sont en cours à l'ENST Bretagne.

Comme on a pu le remarquer dans 27.3.1, les courbes de délais mesurés à l'aide de Saturne comportent des pics de quelques centaines de microsecondes. Ces pics sont très courts dans le temps et ne peuvent être attribués ni à NTP, les offset observés en ces moments n'atteignant pas de telles valeurs, ni au réseau, les machines étant reliées directement l'une à l'autre.

La figure 27.12 présentant la moyenne glissante sur cinq valeurs des mesures prises toutes les minutes, il convient d'examiner dans le détail les valeurs mesurées pour déterminer l'amplitude des pics obtenus.

Ces pics sont d'amplitudes variables et dépassent parfois la milliseconde. Ces pics ne durent par ailleurs que le temps d'une mesure, puisque les précédentes et suivantes retrouvent la valeur « normale ». On dénombre ainsi 13 pics de plus de 100 microsecondes sur les 2880 mesures représentées, représentant donc 0,45% des mesures. Des pics de même amplitude ont été observés dans une proportion similaire lors des mesures entre Paris et Rennes. On peut donc penser que ce problème provient du système d'exploitation (cf. 27.4.1) le programme d'estampillage utilisant une grande partie des ressources (de l'ordre de 99%).

Rude / Crude

RUDE et CRUDE [8] sont deux logiciels distribués sous General Public License [18]. RUDE réalise l'envoi de paquet UDP, incluant une estampille temporelle, selon les paramètres spécifiés dans un fichier de configuration. Un grand nombre de paramètres peuvent être utilisés dont le TOS,

[5] Remote Procedure Call

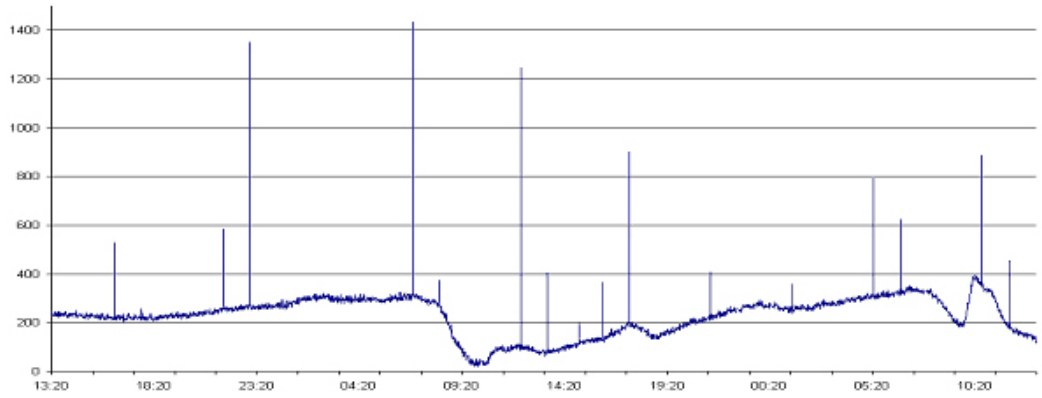


FIG. 27.12 – Délais unidirectionnels mesurés en LAN par Saturne

le temps d'émission ou la taille des paquets. Il est également possible de générer un trafic selon un modèle particulier à partir d'un fichier de trace. CRUDE se charge de collecter les paquets émis par RUDE et de stocker les résultats dans un fichier. La présentation des résultats peut par la suite être effectuée par l'analyse des données contenues dans le fichier, à l'aide d'outils tel que MRTG (Multi Router Traffic Grapher [45]) ou Cricket, comme il est suggéré dans le projet SEQUIN [9]. Le principal avantage d'une telle solution est lié à son statut logiciel libre, la rendant aisément adaptable aux besoins.

L'installation de Rude/Crude se fait à l'aide du traditionnel configure/make/make install sur chacune des plateformes. Un fichier de configuration permet de définir de nombreux paramètres pour l'émission (taille des paquets, durée d'émission, fréquence d'émission, ...). La collecte des résultats s'effectue par Crude directement dans un fichier texte ou dans un fichier en format binaire pour décodage ultérieur. Il est également possible de recueillir directement les statistiques (nombre de paquets perdus, délais moyens...) après chaque campagne de mesure. Dans notre cas, les résultats étaient stockés dans un fichier binaire et extraits par un script invoqué périodiquement par mrtg.

Les mesures sont effectuées selon le principe décrit par la Figure 2-4. Les paquets ont une taille de 500 octets et sont envoyés toutes les minutes.

A noter que les modifications du code source s'avèrent plus difficiles que pour Saturne, le programme étant moins modulaire (estampillage, gestion de la configuration et émission sont confondus).

Les graphes obtenus par mrtg sont analogues à ceux obtenus avec Saturne. Cependant, une analyse similaire à celle effectuée dans la partie précédente (test avec Saturne) et effectuée dans le même contexte montre que cet outil ne subit pas de pics significatifs contrairement à Saturne (cf. figure 27.13 et 27.12). On observe cependant des variations du délai dans un intervalle de 200 microsecondes.

L'absence de pics peut s'expliquer par le fait que Rude et Crude ont été paramétrés pour être prioritaires face aux autres processus du système.

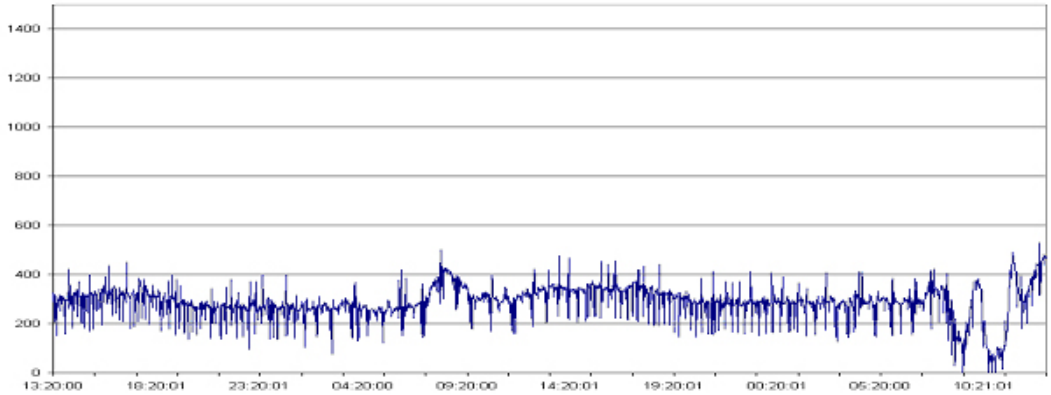


FIG. 27.13 – Délais unidirectionnels mesurés en LAN par Rude / Crude

A noter que lors d'une autre campagne de mesures avec Rude et Crude un pic a été obtenu. Si Rude et Crude limitent donc les erreurs dues aux système d'exploitation, ils ne peuvent les supprimer totalement.

RIPE NCC Test Traffic Measurement Service

Développé par le RIPE NCC[6], cette solution ([41] et [5]) dédiée à ses membres permet de mesurer les métriques citées précédemment ainsi que la bande passante. Ceci est effectué au moyen de boîtiers, de format 1U ou 4U, qui envoient à chacun des autres boîtiers deux paquets sonde de cent octets par minute, contenant chacun une estampille temporelle obtenue par GPS. Les données collectées sont ensuite envoyées au serveur central hébergé par RIPE NCC puis présentées au moyen d'une interface web. Le principal avantage de cette solution est qu'elle ne nécessite pas de paramétrage particulier, l'ensemble de la chaîne de mesure étant configurée par RIPE. Ceci peut cependant constituer un inconvénient puisqu'il n'est pas possible de réaliser d'autres types de tests ou de présentations des données autrement qu'en attendant les mises à jour.

Les plateformes RIPE TTM sont livrées et installées prêtes à l'emploi. Au format 1U leur prix est de 2500 euros. L'installation coûte 250 euros et les frais de fonctionnement s'élève à 3000 euros par an. A noter que ces tarifs sont dégressifs en fonction du nombre de boîtiers utilisées. La configuration des mesures et leur présentation s'effectuent simplement via un serveur web intégré. Les formulaires permettent de spécifier simplement un grand nombre de paramètres tant pour la configuration (taille, fréquence d'émission) que pour la présentation des résultats (durée, paire émetteur-récepteur...). Les graphes présentés sont très nombreux (aperçu et résumé des délais, des gigues, traceroutes, nombre de satellites vus, ...) et il est possible de tracer des graphes à la demande. Les résultats collectés par la plateforme durant les quinze derniers jours sont disponibles à partir du serveur web local. Les résultats sur de plus longues périodes sont envoyés quotidiennement au serveur central à Amsterdam pour être stockés et présentés sur un serveur web. RIPE utilise la plateforme d'analyse de données ROOT [40] développée par le CERN[7] pour le stockage, l'analyse et la présentation des données. La Figure 27.14 présente les délais mesurés entre le LIP6 et l'ENST.

[6] Réseaux IP européens Network Coordination Center
[7] Centre d'Etudes et de Recherches Nucléaires

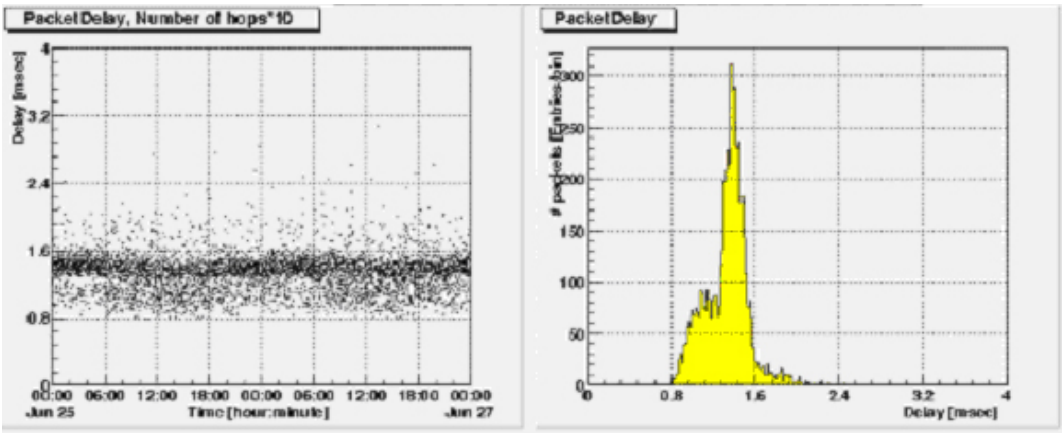


FIG. 27.14 – Délais mesurés par RIPE TTM entre le LIP6 et l'ENST

On constate une légère dispersion autour de la valeur moyenne pouvant avoir les mêmes causes que celle observée avec Saturne. Elle peut cependant être due au réseau (phénomène de "re-routage"). On constate toutefois une dispersion identique sur les autres graphes consultés lors de notre étude. A la lumière de cette constatation et de la conception des stations RIPE, on peut supposer que ces stations sont également sujettes à des erreurs causées par le système d'exploitation.

NIMI

NIMI (National Internet Measurement Infrastructure [25] [5]) est une architecture complexe visant à déployer des mesures sur un réseau de grande ampleur. L'infrastructure NIMI est constituée de sondes et de plateformes de configuration et de contrôle de ses sondes. Ces dernières permettent de planifier les mesures et de contrôler l'accès aux mesures. Les mesures sont effectuées par des modules dans les sondes. Cette conception modulaire permet ainsi de définir d'autres types de mesure, étendant ainsi les possibilités du dispositif.

L'architecture NIMI est relativement complexe. Elle se compose de 4 modules :

- Le Probe est chargé d'effectuer les mesures. Il est lui-même divisé en deux parties (Nimid et Scheduled), la première communiquant avec les autres modules et la seconde exécutant les mesures. Les mesures sont effectuées par des outils invoqués selon une API par le Scheduled.
- Le CPOC (Configuration Point Of Contact) administre un ensemble de probes.
- Le MC (Measurement Client) est la partie cliente à l'origine des tests sur le réseau.
- Le DAC (Data Analysis Client) stocke les résultats pour une exploitation future. Par défaut il est joint au MC.

Le principe de l'architecture NIMI est illustré Figure 27.15. Les différents échanges, à l'exception des mesures, sont chiffrés et chaque probe contient une table ACL (Access Control List) distribuée par le CPOC indiquant quelles mesures il autorise pour quels MC.

Quelques mesures ont été effectuées avec Zing, un des modules de mesure intégrés à NIMI, afin de vérifier le fonctionnement du système. L'objectif était plus de comprendre et d'évaluer l'archi-

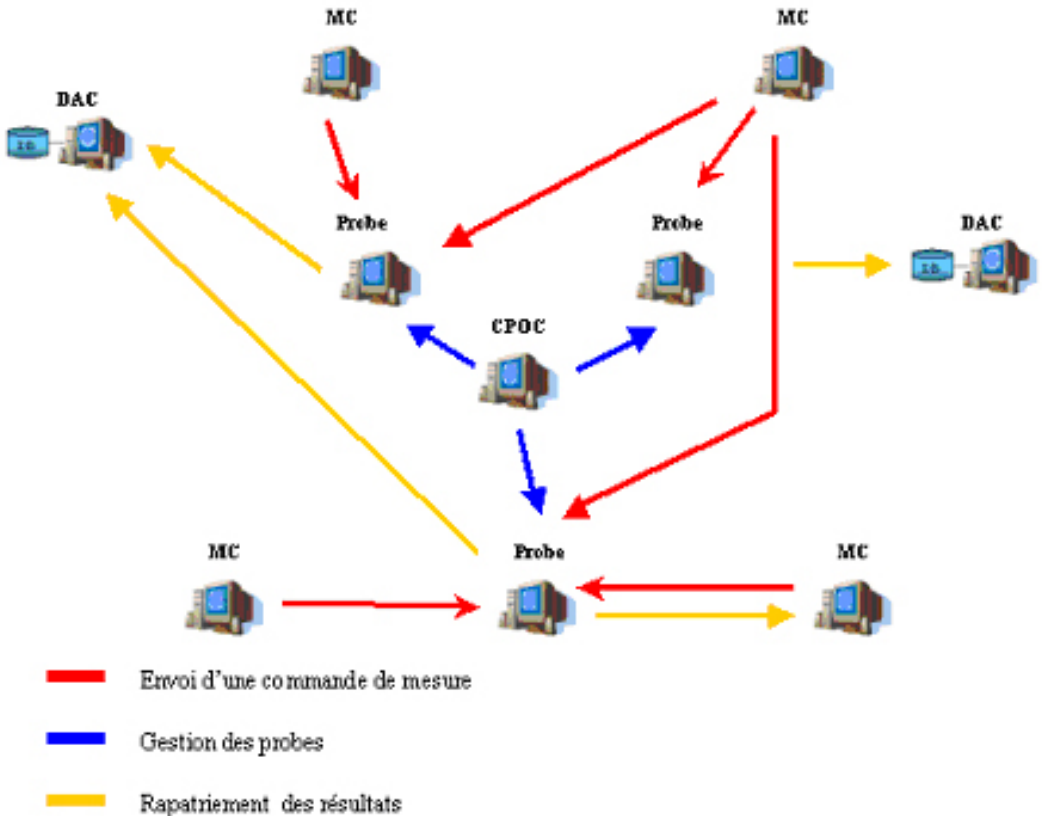


FIG. 27.15 – Architecture NIMI

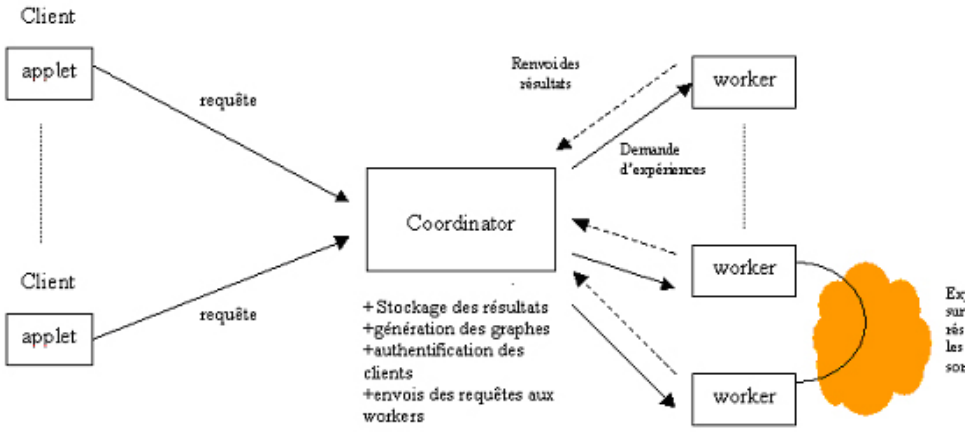


FIG. 27.16 – Architecture ANEMOS

tecture, en vue de l'utiliser conjointement avec un outil de mesure unidirectionnel, que de mesurer l'efficacité des outils.

La mise en œuvre de NIMI s'est avérée difficile, la documentation existante étant insuffisante et le code très imposant (plusieurs milliers de ligne). Son usage tel quel ou son adaptation apparaissent donc relativement difficiles et peu souhaitables dans notre contexte. Ses caractéristiques principales (architecture, contrôle, sécurité) peuvent cependant servir de base à l'élaboration d'un outil plus simple et adapté au contexte du réseau RENATER.

ANEMOS

ANEMOS (Autonomous Network Monitoring System [11]) permet de planifier, d'effectuer et d'analyser des mesures actives à partir d'une interface web. Développé en java, il est comme NIMI composé de différents types de plateforme et effectue ses mesures par l'intermédiaire de modules. Il est actuellement en version beta et permet pour l'heure la mesure de la bande passante et celle du RTT par une déclinaison UDP du Ping.

L'architecture d'ANEMOS s'appuie sur trois composants :

– Le client permet, à travers une applet java, de planifier les mesures. On peut se connecter à partir de ce client en utilisateur ou administrateur, les différents utilisateurs et administrateurs étant déclarés dans une table MySQL. L'utilisateur peut lancer des mesures entre deux Workers, accéder aux résultats et ajouter des seuils au-delà desquels une alarme sera déclenchée. L'administrateur peut quant à lui consulter la liste des Workers installés et des expériences en cours, ajouter ou retirer des Workers et arrêter des expériences.
– Le Worker effectue les mesures. Pour ce faire il invoque le module responsable de la mesure et communique avec le Worker de l'autre extrémité pour coordonner la mesure. Pour l'heure, deux modules sont disponibles, l'un réalisant la mesure du RTT par l'équivalent UDP de Ping et l'autre la mesure de la bande passante. L'existence de modules chargés des mesures doit à

terme, comme pour NIMI, permettre d'ajouter de nouveaux types de mesures de façon simple. Dans l'immédiat cet ajout passe par certaines modifications du code source.
- Le Coordinateur reçoit les demandes de mesures des clients et les relaie aux Workers. Il se décompose en trois modules : le "clienthandler" communiquant avec le client, le "workerlistener" récupérant les résultats et les stockant dans une base MySQL, le "surveyor" gérant la file d'attente des tâches et commandant aux workers le début et l'arrêt des mesures.

Les communications entre ces différents composants sont détaillées sur la Figure 27.16.

La sécurité est assurée par l'authentification et le chiffrement des données. L'authentification par login et mot de passe a lieu lors des échanges client-coordinator. Le chiffrement symétrique intervient lors de toutes les communications à l'exception des mesures.

L'installation d'ANEMOS s'effectue soit à partir des sources soit à partir des binaires. Il faut ensuite modifier le script de création de la base de données et le fichier de configuration du Coordinateur. Dans les faits plusieurs problèmes peuvent intervenir. Ayant opté pour l'installation à partir des binaires nous avons par exemple dû reconstruire certains exécutables à partir des sources après quelques modifications. La nécessité d'installer et de configurer MySQL peut par ailleurs être une difficulté supplémentaire.

L'utilisation d'ANEMOS est relativement simple. En mode utilisateur, l'interface de l'applet propose un onglet de planification des mesures, un onglet de définition des alarmes et un onglet de consultation des graphes de résultats.

Les essais réalisés au GIP RENATER n'ont cependant pas donné de résultat. Il n'a en effet pas été possible de tracer des graphes et les résultats bruts produits par les outils fournis se sont avérés aberrants. ANEMOS propose donc un concept intéressant mais n'est pas encore tout à fait mature (version Bêta).

Ping

Ping -commande Unix standard- est l'outil de métrologie active utilisé le plus couramment. Il mesure le RTT à l'aide de l'envoi d'un paquet ICMP "echo request" auquel la machine réceptrice répond par un ICMP "echo reply". Il est intégré à tous les systèmes d'exploitation et ne nécessite aucune configuration particulière.

La Figure 27.17 montre la grande stabilité du ping. Il est également a priori relativement précis. Une meilleure appréciation de sa précision nécessiterait cependant des mesures fiables des délais unidirectionnels avec une synchronisation par GPS, ce qui n'a pu être réalisé dans notre contexte.

On soulignera enfin que d'autres mesures effectuées en LAN ont fait apparaître des pics non corrélés à l'éventuelle charge du réseau.

QOSMetrix

QOSMetrix [39] propose une architecture constituée de sondes sous formes de boîtiers ("Netwarrior") dont les résultats sont collectés et présentés par la plateforme "Netadvisor". Cette dernière peut également contrôler les sondes. Les métriques collectées sont nombreuses ; outre celles standardisées par l'IPPM, l'architecture caractérise également les performances des protocoles de plus

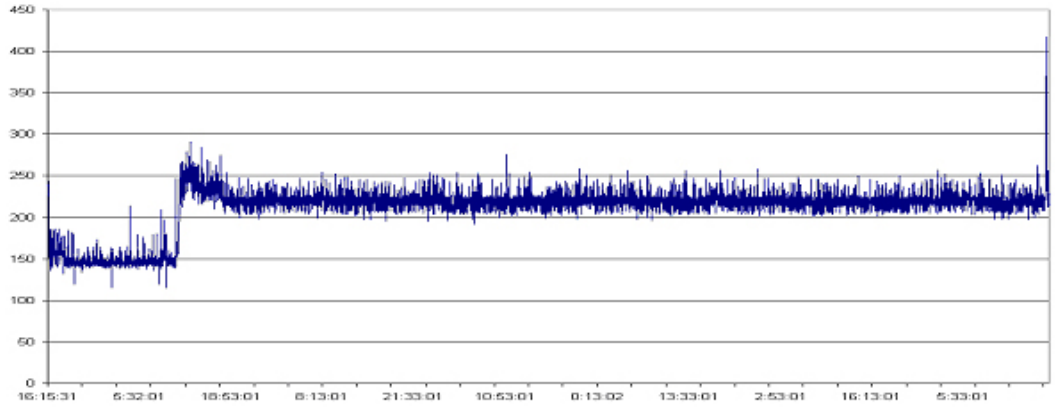


FIG. 27.17 – Mesures du ping en LAN

haut niveau (RTP/RTCP par exemple) pour les trafics unicast ou multicast. Cette solution présente l'avantage de pouvoir être déployée rapidement dans de grands réseaux, "Netadvisor" pouvant gérer plusieurs centaines de sondes. La question de la synchronisation a été étudiée de près : référence GPS ou NTP ou une carte PCI avec une horloge intégrée. De plus le hardware et le système d'exploitation sont customisés : pas de disque dur (donc moins d'interruption système) et un noyau Linux spécifique. Malheureusement cette solution n'a pas pu être testée en 2003. Cette action est prévue au cours de l'année 2004.

27.4.3 Conclusion

Parmi les outils étudiés, seul Cisco SAA ne peut être envisagé du fait de sa trop faible précision. L'architecture et les fonctionnalités de NIMI pourront ainsi servir de base à l'architecture développée. Son adaptation à un outil de mesures unidirectionnelles n'est cependant pas envisagée, la modification du code source s'avérant trop importante.

Pour ce qui est de la réalisation des mesures en elles-mêmes, les différents outils testés présentent de bonnes performances (à l'exception de Cisco SAA) et peuvent donc être utilisés pour une surveillance du réseau. Il faut néanmoins souligner deux points :

- Les valeurs obtenues ne sont pas absolument fiables. On observe en effet plus ou moins fréquemment des pics non dûs aux réseaux. L'amélioration de ce comportement passe à priori par l'utilisation d'un système d'exploitation temps réel. Ces pics sont cependant très peu fréquents et pourront, selon le niveau d'exigence, être tolérés.
- Si la mesure par Ping est fiable, les résultats obtenus ne sont pas forcément représentatifs du trafic émis par l'utilisateur. Ping étant en effet basé sur des messages ICMP, ces paquets peuvent être traités différemment des autres paquets IP.

Les études menées dans ce chapitre amènent donc à considérer, entre autres possibilités, la réalisation d'un outil simple, mesurant RTT, délais unidirectionnels, gigues, pertes et déséquencement par l'envoi de paquets UDP, fonctionnant sur une plateforme temps réel afin de proposer une mesure fiable. La mesure du RTT ne nécessite en effet pas de synchronisation particulière et peut donc être extrêmement précise quelle que soit la synchronisation choisie. La détection d'anomalies par ce

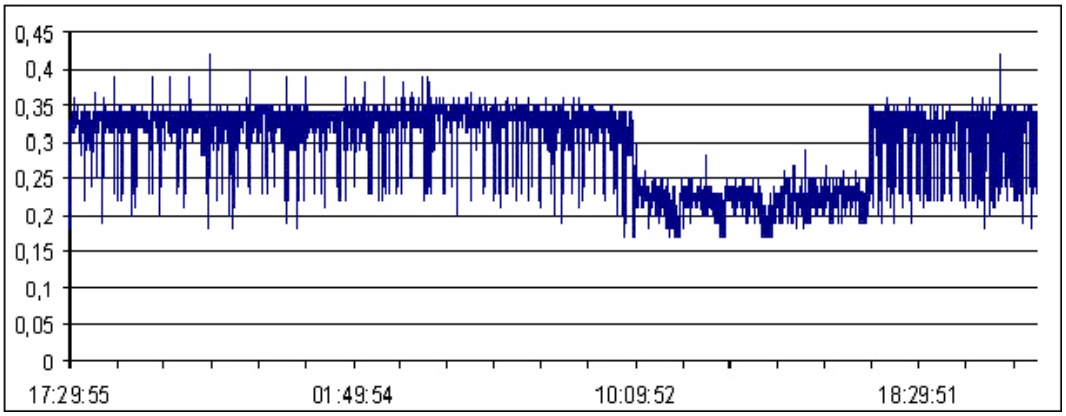


FIG. 27.18 – Mesure de l'influence de la charge

RTT pourrait alors donner lieu à une analyse plus approfondie des délais unidirectionnels, ceux-ci donnant alors une information qualitativement intéressante même si pas nécessairement précise.

27.5 Système d'exploitation et temps réel

Les mesures effectuées précédemment font apparaître des pics, plus ou moins fréquents et importants, dont le système d'exploitation semble être en partie responsable. En effet, si d'autres mécanismes "bas niveau", ARP[8] par exemple, peuvent expliquer en partie ces pics, l'ampleur ne peut qu'être due au mécanisme mis en œuvre dans le noyau. Cette hypothèse se trouve par ailleurs confirmée par différentes études [32]. Afin de quantifier pleinement ce phénomène, nous avons mesuré le RTT entre deux machines à l'aide d'une version légèrement modifiée du ping UDP fournie par ANEMOS. Un paquet de 100 octets était envoyé chaque seconde. Afin d'évaluer l'influence de la charge on exécutait durant une certaine période le programme de craquage de mot de passe John the Ripper. La figure 27.18 présente les résultats obtenus.

Cette figure fait clairement apparaître la complexité de la problématique temps réel. On remarque en effet une baisse du délai mesuré a priori illogique lorsque la charge augmente (jusqu'à plus de 95% de CPU). La cause de ce phénomène n'a pas été établie, mais on peut supposer qu'il est dû à une fonctionnalité du noyau linux. Les variations sont sur cette figure de l'ordre de 200 microsecondes. On a également constaté la présence de pics (non représentés sur cette figure pour des raisons de clarté) de plusieurs dizaines de millisecondes. Ces pics sont néanmoins très peu fréquents (de l'ordre de un pour dix mille mesures à un pour cent mille mesures selon les campagnes).

La réalisation d'un système de métrologie apte à superviser finement le réseau RENATER en temps réel nécessite une précision de l'ordre de 100 microsecondes. Il apparaît donc nécessaire de réduire autant que possible l'apparition et l'amplitude de ces pics. Le problème étant dû à la nature non temps réel des systèmes d'exploitation mis en œuvre, plusieurs solutions peuvent être envisagées [32] :

[8] Adress Resolution Protocol. Protocole de résolution d'adresse utilisé pour déterminer l'adresse physique correspondant à une adresse logique

– les systèmes temps réel propriétaires (VxWorks [47], QNX [38] ...).
– les patchs préemptifs (“preempt kernel” [19] ou “low latency” [1], ...) diminuant le temps de latence du noyau linux
– les solutions ajoutant au système linux standard des capacités temps réel par l’ajout d’un mécanisme d’ordonnancement temps réel (RTLinux, RTLinux/Free [17], RTAI [10]) ....

Les premières solutions nécessitent un important investissement, tant financier qu’humain. Les licences d’utilisation sont bien souvent coûteuses et le développement d’application nécessite un apprentissage particulier. Pour ces raisons leur implémentation dans le contexte de ce projet n’est pas souhaitable.

Le principal avantage de la seconde catégorie est sa simplicité de mise en œuvre. Les applications n’ont en effet pas à être réécrites. La contrepartie est que l’on obtient ainsi un temps réel “soft”, c’est à dire que les temps de traitement des données ne peuvent être garantis qu’en moyenne. On peut donc avoir ponctuellement une certaine latence, même si celle-ci est plus faible que pour un système non patché.

La dernière catégorie propose quant à elle un temps réel “dur”. Les processus sont dans ce cas certains d’accéder au bout d’un temps borné, et faible, au processeur. Les applications doivent cependant être réécrites pour tirer profit du support temps réel.

Nous nous intéressons donc par la suite au patch “low latency”, plus efficace que son homologue “preempt kernel” d’après [32], et à RTAI, produit libre concurrent du commercial RTLinux et mieux maintenu que la version libre RTLinux/Free.

27.5.1 Patch low latency

Le principe des patchs préemptifs est de réduire les temps passés dans des boucles d’attente trop longues et d’attribuer ainsi à chaque tâche des quanta de temps presque égaux. Le processeur étant moins susceptible d’être monopolisé par un processus, on réduit ainsi sensiblement les temps de latence du noyau.

Les mesures présentées dans [32] font état d’une spectaculaire amélioration de latence. On obtient en effet une latence maximale de 300 microsecondes dans un contexte où un linux standard donne des pics jusqu’à 66 millisecondes. Ces résultats dépendent cependant des configurations utilisées et du contexte, une machine rapide et peu chargée étant naturellement moins sujette à des latences importantes. Les tests effectués au GIP RENATER entre un ordinateur portable et un pc, tout deux à 1 GHz et “patchés” par “low latency”, font ainsi apparaître des pics (1 pour 10000 mesures) pouvant atteindre 12ms. Les mesures effectuées dans le même contexte que celles réalisées dans 4.3 mais avec la machine émettrice modifiée par “low latency” montrent des variations de l’ordre de 150 microsecondes et, surtout, une plus grande stabilité lors de la charge. Des pics analogues à ceux de l’expérience du **??** sont cependant intervenus lors de ces mesures. Ils peuvent néanmoins être attribués à la machine réceptrice, non modifiée avec “low latency”. Une vérification sera donc nécessaire.

27.5.2 RTAI

RTAI (Real Time Applications Interface) a été développé à l’école polytechnique de Milan afin d’avoir une alternative aux coûteux systèmes propriétaires. Inspiré de RTLinux, RTAI en reprend

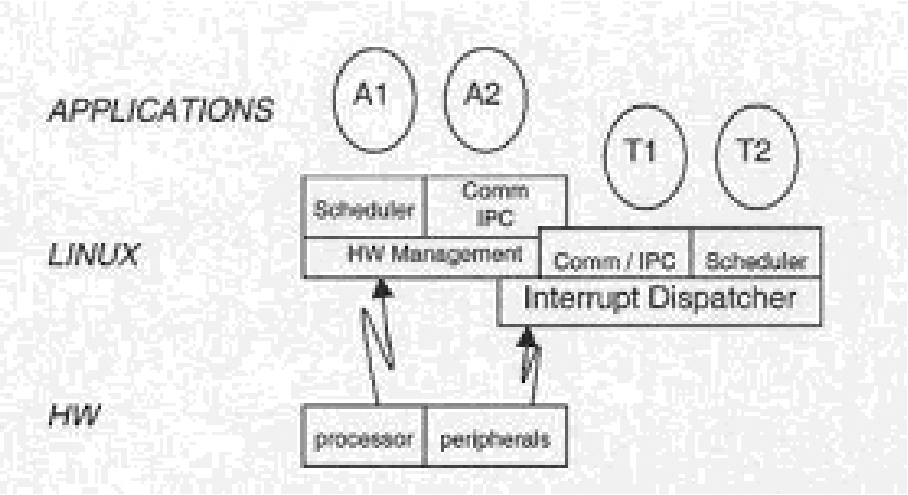


FIG. 27.19 – Architecture de RTAI (Source : [33])

le concept (ajout d'un noyau auxiliaire au noyau standard), en améliorant certains aspects. Mis au point depuis plus de trois ans, il est aujourd'hui largement utilisé.

RTAI intercepte les interruptions matérielles et, si nécessaire, les reroute vers le noyau Linux. RTAI va donc traiter directement les tâches temps réels et déléguer les autres tâches au noyau linux, ce dernier fonctionnant comme tâche de fond de plus faible priorité. Pour ce faire, une couche d'abstraction matérielle (Hardware Abstraction Layer : HAL) est ajoutée au noyau sous la forme d'un "patch". Deux HAL sont supportées par RTAI : RTHAL et, plus récemment, ADEOS.

RTAI est constitué de plusieurs modules ("rtai, rtai_sched, rtai_fifos, lxrt, rtai_pqueue, rtai_pthread, rtai_utils"). Un ou plusieurs de ces modules, selon les capacités utilisées, doivent être chargés afin de faire fonctionner une tâche temps réel.

Les applications temps réel sont normalement programmées, en utilisant une API particulière, sous la forme de tâches fonctionnant dans l'espace noyau. Ces tâches sont lancées en tant que module du noyau (via la commande modprobe par exemple) et peuvent communiquer avec les applications non temps réel par divers mécanismes (FIFO, mémoire partagée ...).

Cependant à la différence de RTLinux, RTAI permet au moyen de l'extension LXRT de développer des tâches temps réel dans l'espace utilisateur, au prix d'une dégradation très faible des performances. Outre la plus grande facilité de développement, ceci autorise les utilisateurs autres que "root" à lancer des tâches temps réels.

L'installation de RTAI s'effectue en deux temps : construction du noyau patché puis installation des modules RTAI. La première étape est analogue à l'installation de low latency. La seconde consiste à configurer et compiler les modules composant RTAI, ceci s'effectue simplement par les commandes suivantes, exécutées depuis le répertoire contenant RTAI : "make menuconfig, make dep, make ; make install, make dev". Afin de rendre pleinement temps réel les applications utilisant le réseau il convient également de rendre temps réel les piles TCP/IP. RTnet [15] propose donc une pile TCP/IP temps réel "dur" ainsi qu'un protocole de niveau liaison de donnée destiné à éviter les collisions afin de rendre complètement déterministe les échanges réseaux. RTnet est cependant rela-

tivement jeune (version 0.2.10), comme ont pu nous le confirmer ses développeurs, et souffre d'un manque de documentation. De plus, il n'est pour l'heure pas compatible, avec ADEOS. Nous ne sommes donc pas parvenus à le faire fonctionner de manière satisfaisante lors de nos tests.

27.5.3 Synthèse

Si l'utilisation de RTnet et RTAI semble pouvoir garantir un fonctionnement temps réel de l'outil de métrologie, son implémentation s'avère pour l'heure relativement difficile. La solution "patch" préemptif est ainsi beaucoup plus simple à mettre en œuvre dans l'immédiat. Des "patchs" préemptifs sont par ailleurs déjà intégrés au noyau de développement 2.5 et devraient être intégrées dans le prochain noyau stable 2.6, devant voir le jour fin 2003.

Si l'application d'un tel "patch" améliore considérablement la latence du système d'exploitation, la ramenant à des valeurs acceptables, elle n'est cependant pas conceptuellement aussi efficace qu'un système d'exploitation temps réel natif (QNX) ou émulé (RTLinux, RTAI). On peut donc s'attendre à des aberrations ponctuelles, en particulier lorsque la machine est très chargée.

La problématique temps réel, nécessaire à la fiabilité des mesures, s'avère donc extrêmement complexe. L'amélioration des performances passe en effet non seulement par l'amélioration du système d'exploitation mais également par celle des autres couches du système, éventuellement jusqu'au dispositif physique (carte réseau par exemple). Le développement des outils de mesure est également capital. On a en effet pu constater des différences notables entre les différents outils.

En résumé, un soin particulier doit être porté à l'optimisation de l'outil (prioritisation ...) et de son environnement (système d'exploitation, réseau ...) lors de l'implémentation du système en contexte opérationnel, en particulier si ce système doit s'avérer chargé. Du fait de la fréquence faible d'apparition des phénomènes, on peut enfin envisager de les éliminer lors d'un post traitement ou de laisser le soin à l'administrateur de les écarter.

27.6 Conclusion

La spécification des besoins pour un contrôle des SLA sur un réseau à haut débit a montré la difficulté d'effectuer des mesures fiables et précises. De nombreux domaines techniques sont abordés (synchronisation, architecture logicielle et matérielle, systèmes d'exploitation, ...). Les offres sont très variées et encore souvent en cours d'élaboration ou de maturation, le choix d'une solution reste donc délicat. Cependant les connaissances acquises lors de cette année 2003 vont permettre la mise en place d'un système de contrôle des SLA sur le réseau RENATER au cours de la dernière année du projet. Les offres QOSMetrix et DFN[9] seront ainsi testées puis une solution déployée sur le backbone RENATER.

[9] *Deutsche Forschungsnetz : homologue alemand de RENATER qui a developpé et déployé sa propre solution de mesures actives au cours de l'année 2003 - http ://www-win.rrze.uni-erlangen.de/cgi-bin/ipqos_disp.pl*

Chapitre 28

Une extension du modèle de tarification “smart market” pour l’Internet basé sur le contrôle de congestion

28.1 Introduction

La tarification de services de communication dans l’Internet représente un enjeu considérable, mais pose de gros problèmes aux opérateurs Internet actuels et aux ISP[1]. A l’heure actuelle, ils n’existe pas de solution répondant à la fois à l’attente des utilisateurs et des opérateurs. En effet, les principales offres de tarification disponibles dans les réseaux commerciaux opérationnels sont principalement de deux sortes :

– Une connexion à la durée vers le réseau de l’ISP : cette approche, issue du réseau téléphonique, est encore utilisée pour les utilisateurs accédant à l’Internet par l’intermédiaire d’un modem téléphonique. Ce modèle de tarification est défavorable aux utilisateurs qui doivent payer leur connexion même lorsqu’ils ne transmettent pas d’information. Ceci est particulièrement vrai dans les régions où les appels téléphoniques locaux ne sont pas forfaitaires.
– Un abonnement : ce principe de facturation est celui qui s’est imposé avec l’avènement des accès à l’Internet par câble ou ADSL[2]. La plupart du temps, les utilisateurs souscrivent une connexion permanente à l’Internet, dont le prix est basé sur sa capacité, et disposent ainsi d’un accès illimité à l’Internet. Ce principe de facturation est très favorable aux utilisateurs, et après plusieurs années de déploiement, apparaît comme très dangereux pour les opérateurs et les ISP. En effet, avec un accès illimité, les utilisateurs génèrent une très grande quantité de trafic, par exemple des fichiers musicaux ou vidéos, parfois volumineux, échangés par l’intermédiaire

[1] ISP : Internet Service Provider
[2] ADSL : Asymmetric Digital Subscriber Line

d'applications P2P[3]. A cause de l'importance grandissante de ce nouveau type de trafic, les réseaux des opérateurs ou des FAI commencent à être congestionnés (ou du moins plus assez sur-dimensionnés) ce qui provoque des diminutions de QdS[4]. Ainsi, ils doivent faire évoluer leurs réseaux pour pouvoir fournir des services Internet de qualité ce qui représente pour les opérateurs, un surcout important.

A l'heure actuelle, réaliser une tarification d'un réseau à service unique n'est pas chose aisée. Mettre en place un système de tarification dans un réseau multi-services devient encore plus complexe. Ainsi, pour permettre une tarification plus adaptée aux services Internet il est important de cibler les enjeux d'un tel réseau.

- Tout d'abord, il faut pouvoir homogénéiser les divers domaines ou AS[5] de l'Internet qui n'offrent pas la même QdS et intégrer dans ce schéma la vision de bout en bout des utilisateurs. Ceci constitue l'aspect "horizontal" de cette problématique. En effet, l'Internet est une interconnexion de domaines indépendants, gérés de façon autonome, qui possèdent leurs propres capacités et proposent leurs propres CdS[6]. C'est particulièrement vrai pour les services DiffServ [3]. Dans ce cas, les services peuvent être différents d'un domaine à l'autre, et les utilisateurs vont obtenir le niveau de QdS du domaine le moins performant même s'ils sont facturés pour un service de niveau supérieur. Ainsi, il apparaît qu'il est nécessaire de prendre en compte le point de vue de bout en bout des utilisateurs pour proposer des mécanismes de différenciation et de facturation.
- Ensuite, il est nécessaire de fusionner les comportements des utilisateurs et des opérateurs. En effet, les seconds facturent un service de niveau réseau (couche 3) pendant que les premiers disposent d'un point de vue applicatif (couche 7). Par exemple, les opérateurs font payer les retransmissions alors que les utilisateurs voudraient uniquement payer pour le trafic utile dont ils disposent. Cela représente l'aspect "vertical" de cette problématique.

Dans ce chapitre, nous proposons une approche de tarification qui combine ces points de vue et établit un pont entre les différents aspects de cette problématique. Cette approche, qui résulte des travaux de caractérisation du trafic Internet effectués dans le SP3 [31] s'inspire de l'esprit du modèle "smart market" (décrit dans la section suivante) où les utilisateurs paient des prix élevés pour le trafic qu'ils émettent en période de congestion durable, et qui gêne donc la résorption de cette congestion. Bien sûr, à l'heure actuelle, les phénomènes de congestion à long terme sont plutôt rares dans l'Internet et de fait, le modèle "smart market" doit être rénové par rapport à cette problématique. Ainsi, notre approche de différenciation et de tarification de service s'appuie sur les différents niveaux d'agressivité des mécanismes de contrôle de congestion (de TCP[7] notamment) employés par les différents services mis en œuvre. L'agressivité des mécanismes utilisés permet à un flux d'utiliser plus de ressources, et donc de maximiser sa QdS (ainsi que celle de tous les flux qui vont utiliser le même mécanisme de contrôle de congestion), mais elle engendre aussi des oscillations néfastes pour les performances globales du réseau [35]. Les oscillations sont en grande partie dûes aux mécanismes d'autorégulation des protocoles de transmission comme ceux de TCP (section 28.2). Ainsi, dans notre approche, des trafics oscillant fortement doivent être facturés plus cher, à cause à la fois des perturbations qu'ils introduisent dans le réseau global et de la dégradation des performances qui en résulte. Cette nouvelle approche rénove le modèle "smart market" pour l'Internet nouvelle génération, en considérant plutôt l'aspect dynamique du trafic que les comportements de congestion

[3] P2P : Peer to Peer
[4] Qualité de Service
[5] AS : Autonomous Systems
[6] CdS : Classe de Service
[7] TCP : Transmission Control Protocol

à long terme. De plus, ce modèle s'applique de façon continue, et pas seulement au moment des congestions.

Le plan de ce chapitre est le suivant : tout d'abord dans la section 28.2, nous exposons les principes de notre approche de tarification et nous montrons comment nous nous inspirons de l'approche du modèle "smart market" que nous adaptons aux besoins de l'Internet actuel. Ensuite, dans la section 28.3, nous présentons une série d'expériences nous permettant de valider notre modèle de tarification. Nous démontrons que cette approche est cohérente à la fois avec les points de vue utilisateurs et opérateurs, ainsi qu'avec la structure multi-domaines de l'Internet. Ces expériences ont été réalisées avec le simulateur NS-2[8] et nous permettent de proposer une évaluation quantitative de notre modèle de tarification. La section 28.4 propose ensuite plusieurs principes pour des mécanismes de facturation des utilisateurs en accord avec notre modèle de tarification. Enfin, la section **??** conclut ce chapitre.

28.2 Une nouvelle approche de différenciation de service et de tarification pour l'Internet

28.2.1 Oscillations et contrôle de congestion

Les mesures actuelles de métrologie sur les liens de l'Internet révèlent la présence d'oscillations dans le trafic Internet [28]. Un exemple de trafic observé sur un lien Internet est donné dans la figure 28.1. Celle-ci compare le trafic Internet actuel avec un modèle simple de trafic : le modèle de Poisson qui était il y a quelques années l'un des modèles supposés de l'Internet. En fait, les courbes de trafic doivent se lisser lorsque la granularité de l'observation augmente. C'est ce qui est représenté dans la figure 28.1 où pour chaque trafic (Internet actuel et Poissonien) l'amplitude des oscillations décroît lorsque la granularité d'observation est plus importante. Sur cette figure, on note aussi la différence entre les deux types de trafic : pour une granularité d'observation importante (1 seconde par exemple), l'amplitude des oscillations du trafic Internet est plus importante et se lisse moins vite que pour celles du trafic Poissonien.

Certaines analyses du trafic Internet réalisées dans le cadre de récents projets de métrologie comme IPMON [13] ou METROPOLIS ont montré que ces oscillations étaient le résultat de la présence de LRD[9] et / ou d'auto-similarités dans le trafic [36]. Ces phénomènes ont plusieurs causes, notamment les mécanismes de contrôle de congestion, et tout particulièrement ceux de TCP qui est le protocole dominant dans l'Internet [34]. Parmi tous les mécanismes de TCP, il est clair que son système de contrôle en boucle fermée introduit de la dépendance à court terme dans le trafic étant donné que les acquittements dépendent de la réception d'un paquet, et que tous les autres paquets de ce flux dépendent de cet acquittement. De la même façon, les deux mécanismes de TCP ("slow-start" et "congestion avoidance"), introduisent de la dépendance entre les paquets de différentes fenêtres de contrôle de congestion. Et naturellement, la notion d'émission en rafale des sources TCP ajoutée à la LRD permettent d'expliquer la présence d'oscillations qui se répètent à toutes les échelles de temps dans le trafic global. En généralisant ces observations, nous pouvons affirmer que tous les paquets d'un flux sont dépendants les uns des autres. De plus, avec l'augmentation des capacités de l'Internet permettant aux utilisateurs d'échanger des fichiers de plus en plus volumineux, comme

[8] NS-2 : Network Simulator version 2
[9] LRD : Long Range Dependence

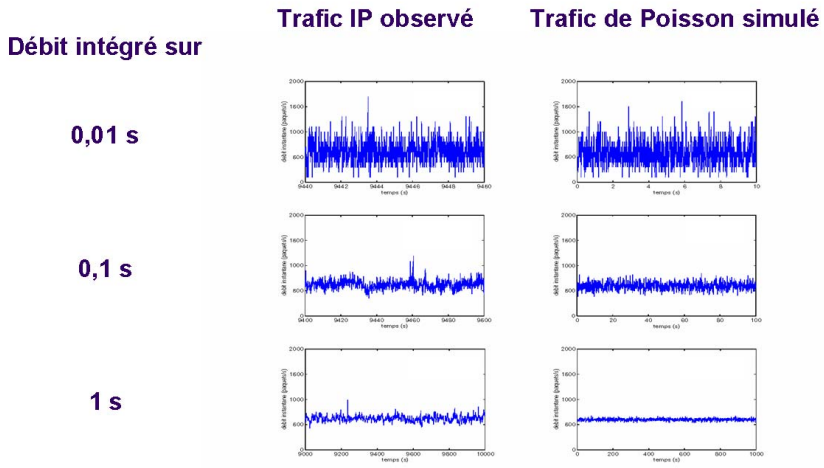


FIG. 28.1 – Comparaison entre les oscillations d'un trafic Internet et celles d'un trafic de type Poissonnien

des données musicales ou vidéo par exemple, il est clair que l'échelle du phénomène de LRD augmente, ce qui explique pourquoi l'amplitude des oscillations mesurées dans l'Internet, même avec une granularité d'observation élevée, est si importante. En effet, les phénomènes de dépendance de TCP se propageant dans le trafic par l'intermédiaire des flux, l'allongement des flux observable avec l'arrivée des applications P2P, augmente aussi la portée des dépendances qui se retrouvent à très long terme. Une oscillation à la date t provoque ainsi d'autres oscillations à des dates pouvant être éloignées de t. Une congestion (sporadique) induite par une forte oscillation sur un flux peut ainsi ne pas être complètement résorbée plusieurs heures après (dans le cas du téléchargement d'un film par exemple), c'est à dire que ce flux qui continuera à proposer au réseau des pics de trafic dépendants de cette première oscillation, engendrera de nouvelles congestions sporadiques[10]. De plus, il est clair que les longs flux, à cause de leur longue existence dans le réseau, et par l'importance des capacités des réseaux la plupart du temps très surdimensionnés, ont le temps d'atteindre des valeurs élevées de la fenêtre de contrôle de congestion (CWND). Aussi, une perte va entraîner une forte baisse, suivie d'une forte hausse du débit du flux. Les nouveaux usages de l'Internet (P2P notamment) qui entraînent la transmission de fichiers de plus en plus gros favorisent l'émergence d'oscillations de très fortes amplitudes et dépendantes sur de très longues périodes[11]. Bien sûr, les oscillations sont très

[10] Naturellement, avec les nouvelles capacités hauts débits des réseaux et le choix des opérateurs de sur-dimensionner les capacités du réseau pour améliorer la qualité de service, les congestions sur de longues durées n'existent plus. Par contre, des congestions sporadiques continuent à exister dans le réseau, notamment dans le réseau d'accès, à cause des fortes oscillations du trafic.

[11] En effet, les résultats métrologiques sur la caractérisation du trafic depuis l'an 2000 ont montré que l'Internet qui était alors presque exclusivement utilisé pour de la navigation web, est aujourd'hui de plus en plus utilisé par des applications P2P pour des échanges de fichiers, souvent de tailles importantes (fichier audio, films, etc.). Aussi, le trafic Internet en 2000 se caractérisait par la transmission de flux courts, avec seulement un très faible pourcentage de flux de tailles plus importantes (environ 2 % des flux faisaient plus 100 Ko). Avec l'arrivée des applications P2P, la proportion de gros flux a considérablement augmenté, de même que la taille moyenne des flux transmis. De fait, le trafic Internet présente aujourd'hui une double caractéristique issue des applications dominantes générant ce trafic, avec de très nombreux petits flux (souris) et un nombre grandissant de gros flux (éléphants) [29].

dangereuses pour l'utilisation globale des ressources du réseau étant donné que la capacité consommée par un flux après une perte, par exemple, ne peut pas être immédiatement utilisée par les autres flux : ceci correspond à un gaspillage de ressources, et évidemment entraîne une diminution de la QdS globale du trafic et du réseau. En effet, plus l'amplitude des oscillations est importante, plus les performances globales dans le réseau sont faibles [35].

Il apparaît ainsi que, la couche transport, par la mise en œuvre de ces mécanismes de contrôle de congestion, agit directement sur le profil du trafic réseau et sur la gestion de la QdS. De plus, elle rend un service perturbé par rapport à celui de la couche réseau. En ce sens, une approche de tarification dans l'Internet ne peut se faire uniquement au niveau 3 et doit prendre en compte, au moins, les spécificités de comportement de la couche 4. Le point de vue des opérateurs qui est de considérer le trafic et la QdS uniquement au niveau réseau est donc insuffisant. En effet, il est indispensable de facturer les utilisateurs en étudiant le niveau d'oscillation qu'ils engendrent en particulier au niveau transport. L'approche que nous présentons dans ce chapitre s'appuie sur les remarques précédentes pour adapter l'approche du modèle de tarification "smart market".

28.2.2 Les principes du modèle "smart market"

Ce modèle a été proposé par Mackie-Mason et Varian [22]. Il s'agit d'un mode de facturation proposé, à l'origine, pour juguler le comportement "agressif" des utilisateurs dans l'Internet qui n'utilisaient pas (ou peu) de contrôle de congestion lors de leurs transferts d'informations (e-mail, ftp ou audio). Un tel comportement pouvant entraîner des phénomènes de congestion généralisée sur le réseau, il leur est apparu nécessaire de mieux facturer les paquets à l'origine de cette congestion.

Ainsi, [22] a proposé d'ajouter à chaque paquet de données prêt à être émis sur le réseau un indicateur reflétant le prix que l'utilisateur est prêt à payer pour émettre son paquet. En parallèle, le réseau dispose d'une borne inférieure[12] pour laquelle les paquets qui possèdent un indicateur dont la valeur est supérieure à cette borne sont véhiculés au mieux (selon les capacités du réseau pendant cette période).

Dès lors, en période d'utilisation normale du réseau (cf. pas de congestion), cet indicateur n'est pas utilisé : il n'y a pas de facturation supplémentaire[13]. Par contre, lorsque des phénomènes de congestion se produisent, les paquets ayant l'indicateur le plus élevé sont véhiculés en priorité car facturés les plus chers. Les autres, dont l'indicateur est inférieur à la borne mesurée par le réseau, sont soit temporairement bufferisés, soit reroutés sur un réseau moins rapide, soit supprimés en fonction des possibilités offertes par les équipements de routage et le réseau.

Ce mode de facturation laisse donc la possibilité à l'utilisateur de choisir le service qu'il souhaite utiliser pour véhiculer ses paquets de données étant donné que plus ce service est performant plus le tarif est élevé (un tel schéma ne s'appliquant qu'en période de congestion du réseau). D'autre part, l'utilisateur se voit facturer un prix de transfert par paquet qui varie de minute en minute en fonction de l'état de congestion du réseau. Un avantage important d'un tel fonctionnement est de ne faire payer que les utilisateurs responsables de la congestion du réseau.

De la même façon, d'autres travaux ont été menés par Kelly et al. pour proposer d'autres méthodes de tarification elles aussi basées sur les phénomènes de congestion. Le lecteur pourra se

[12] Cette borne représente un indicateur de l'état de congestion du réseau.

[13] Néanmoins, une facturation de base est toujours appliquée en fonction des coûts de connexion des différents utilisateurs au réseau de l'opérateur.

référer à deux articles qui ont servi de base aux travaux présentés dans cet article : [16] et [14]. Il est à noter l'existence d'un projet européen ambitieux dans ce domaine proposant des nouvelles solutions de tarification pour les services Internet : le projet "Market Managed Multiservice Internet" [21].

Il existe naturellement de nombreuses autres approches, non basées sur les congestions, qui ont été proposées. Parmi elles se trouvent des approches reposant sur la facturation en fonction de la quantité de données transmises, d'autres se basant sur des mécanismes de tarification par abonnement, etc. Ces propositions – y compris celles basées sur les congestions – connaissent des fortunes diverses et, de plus, leur perception peut changer au cours du temps. Néanmoins, la solution n'a pas encore été trouvée et mise en œuvre. En tant que chercheurs en réseau, et plus particulièrement dans le domaine de la métrologie de l'Internet, il nous est apparu que des caractéristiques du trafic, dommageables pour la QdS, sont imputables aux oscillations dues à TCP et à la dépendance qui existe entre tous les phénomènes oscillatoires des flux. De ce fait, l'esprit du modèle "smart market" qui consiste à facturer plus les utilisateurs qui perturbent le plus la qualité du service réseau, nous apparaît très séduisante en incitant financièrement les utilisateurs à plus de rigueur et permettant d'aider à l'amélioration de la QdS de l'Internet.

28.2.3 Un nouveau modèle de tarification des services Internet

Le modèle "smart market" permet une tarification et une facturation adaptées uniquement au niveau de congestion du réseau induit par l'utilisateur. Evidemment, à cette époque, les congestions étaient les seuls dysfonctionnements importants qui pouvaient pénaliser de façon significative le comportement des applications utilisatrices. Avec les besoins de l'Internet actuel, notamment en termes de garanties temporelles, l'ensemble des dysfonctionnements pouvant affecter les utilisateurs et le réseau est plus important. Dans cette perspective, il nous semble réaliste de mettre en relation le modèle de tarification et les phénomènes d'oscillation dus aux mécanismes de contrôle de congestion. En effet, plus le trafic oscille, plus l'utilisation globale du réseau est faible. Il est donc important de faire payer plus cher les utilisateurs responsables du caractère oscillant du trafic. Pour cela, il faut disposer d'un indicateur permettant de mesurer facilement ce caractère oscillatoire. Etant donné que la notion d'oscillation protocolaire est fortement liée au concept d'agressivité protocolaire (comme nous le verrons dans l'étude expérimentale de la section 28.3), l'agressivité des mécanismes utilisés nous semble représenter le paramètre adéquat pour un tel mode de tarification. En effet, les utilisateurs paieraient pour différentes CdS, chacune représentant un protocole de transport différent et son contrôle de congestion associé. Plus le mécanisme serait agressif, plus il entraînerait d'oscillations sur le réseau, plus les performances globales seraient susceptibles d'être dégradées et plus sa facturation devrait être élevée.

En résumé, notre contribution pour la définition d'un nouveau modèle de tarification des services Internet s'inspire de l'esprit qui est à l'origine du mode de facturation smart market, à savoir la pénalisation des fauteurs de troubles dans l'Internet, responsables des baisses de performances et du faible niveau de QdS du réseau. Par contre, les modalités techniques de tarification sont différentes. En effet, dans notre approche nous ne proposons pas de nous baser sur le niveau de congestion induit par les utilisateurs dans l'Internet mais sur une métrique différente – l'agressivité protocolaire des mécanismes de congestion de niveau transport (cf. définition de ce concept dans la section 28.3) – qui traduit à la fois le niveau d'oscillation qu'engendre les utilisateurs du réseau, la baisse des

performances induites par leur comportement et l'impact sur le niveau de QdS offerte au niveau du réseau.

Les modalités d'applications pratiques concernant ce principe de tarification pour l'Internet seront abordées dans la section 28.4. La section qui suit propose une évaluation expérimentale de cette approche.

28.3 Evaluation

Pour valider notre approche de tarification basée sur des niveaux d'agressivité différents, il nous faut disposer d'un moyen de classifier les flux en fonction de leur niveau d'oscillation (ou encore de leur agressivité). Intuitivement, on peut définir une relation directe entre le niveau d'oscillation d'un mécanisme (ou son agressivité) et les ressources qu'il occupe. En conséquence, plus un protocole oscille, plus il est capable d'utiliser des ressources en quantité importante. Dès lors, nous allons étudier dans cette partie un moyen de différencier les protocoles par niveaux d'agressivité en mesurant leur capacité à consommer des ressources dans le réseau.

28.3.1 Principes des expériences réalisées

L'évaluation de l'approche de tarification décrite dans la section 28.2 a été réalisée en s'appuyant sur un ensemble de simulations NS-2 [26]. La topologie réseau simulée est constituée d'un ensemble de sources de trafic interconnectées par des liens d'accès et concentrées sur un même lien de cœur. Cette topologie va nous permettre de confronter l'agressivité mise en œuvre par les différents mécanismes de niveau transport et la QdS fournie aux applications utilisant ces protocoles. Pour cela, les mesures sont réalisées sur le lien de cœur. Il faut préciser que les RTT[14] sont les mêmes pour tous les protocoles de façon à ce qu'ils fonctionnent dans des conditions similaires et permettent ainsi de tirer des conclusions seulement sur les mécanismes de niveau transport à l'exclusion de tous les autres caractéristiques externes. L'impact du RTT sur le niveau d'agressivité d'un flux sera vu plus avant.

Les différents protocoles analysés dans les simulations sont les suivants :
- TCP Tahoe [2] : la plus ancienne version du protocole TCP implémentant le mécanisme Slow-Start ;
- TCP New Reno [2] : la version de TCP la plus utilisée aujourd'hui ;
- TCP SACK [23] : une nouvelle version de TCP introduisant un mécanisme d'acquittement sélectif qui devrait très prochainement supplanter TCP New Reno ;
- TCP Vegas [4] : une version de TCP non déployée qui propose un mécanisme de contrôle de congestion dont l'évaluation de la congestion se fait par l'analyse de l'évolution des RTT ;
- TFRC [12] : un nouveau mécanisme de contrôle de congestion "TCP-friendly", qui est plus régulier que les précédents mécanismes (en terme de débit émis) et s'adresse donc plus spécifiquement à des applications orientées flux.

Historiquement, ces versions de TCP (excepté TCP Vegas) avaient pour objectif d'être capables d'utiliser le plus efficacement possible les nouveaux réseaux qui offrent toujours plus de bande passante. Leur agressivité devrait donc croître en fonction de leur année d'introduction dans l'Internet.

[14] RTT : Round Trip Time

TFRC ne répond pas à ce schéma étant donné que son objectif est de rendre un service le plus stable possible et non pas le plus agressif possible.

28.3.2 Validation du principe de tarification basé sur l'agressivité des mécanismes de niveau transport

Les simulations ont été réalisées en mettant en concurrence 5 flux de chacun de ces protocoles. Tous les flux démarrent au même moment. Les paquets échangés ont la même taille, d'un flux et d'une version des protocoles à l'autre. Enfin, le nombre de paquets échangés est identique pour tous les flux. Les résultats sont décrits dans les figures 28.2 et 28.3.

La figure 28.2 représente le débit par protocole. Il apparaît clairement ici que le débit, i.e. la QdS, offert par chaque protocole respecte l'ordre suivant :

QdS(TCP SACK) > QdS(TCP New Reno) > QdS(TCP Tahoe) > Qds(TCP Vegas) > QdS(TFRC)

Cet ordre est en accord avec nos prévisions qui se basaient sur le principe que les versions successives de TCP ont été développées pour être de plus en plus agressives et être capables de tirer partie de l'augmentation continue de la capacité des liens de l'Internet. L'agressivité est ici définie comme la capacité d'un protocole à consommer des ressources rapidement. L'agressivité est donc calculée comme la fonction dérivée du débit. Dans un environnement discret, l'agressivité d'un protocole est alors :

$$Ag_{Prot}(t) = \frac{d_{Prot}(t + \Delta t) - d_{Prot}(t)}{\Delta t} \tag{28.1}$$

où :

- Ag_{Prot} représente la fonction instantanée d'agressivité pour un protocole donné $Prot$;
- $d_{Prot}(t)$ représente le débit mesuré pour un protocole ;
- Δt représente la granularité des mesures.

Les résultats de l'agressivité sont représentés sur la figure 28.3. Il faut noter que seules les valeurs positives apparaissent car elles correspondent à une augmentation de la consommation de ressource. La figure 28.3 met ainsi en évidence le fait que l'agressivité protocolaire varie beaucoup pendant la transmission d'un flux (cf. l'évolution de l'ensemble des 5 connexions au cours du temps pour un protocole donné). Il est important de noter ici que ce mode de calcul de l'agressivité protocolaire intègre implicitement l'impact de la valeur RTT établi entre l'émetteur et le récepteur du message. En effet, plus le RTT sera court, plus les acquittements TCP seront rapidement acheminés au travers du réseau et plus la fenêtre de congestion pourra croître rapidement, augmentant ainsi le risque d'oscillation si des pertes venaient à apparaître sur le réseau. L'agressivité peut donc varier au cours du temps pour chacun des flux, notamment en fonction des variations des RTT[15]. De fait, ce qu'il est intéressant de quantifier dans cette situation, c'est la capacité maximale d'un flux ou d'un protocole à

[15] Même si ce n'est pas le cas dans cette simulation pour laquelle, nous le rappelons, les RTT sont maintenus constants.

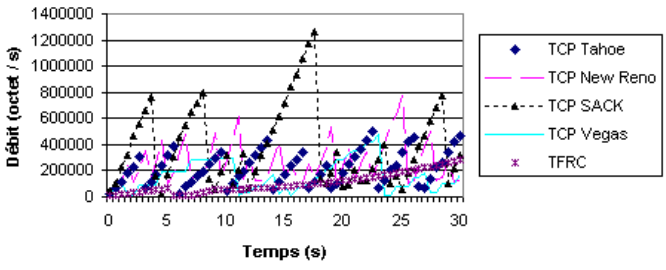


FIG. 28.2 – Débit des différents protocoles de transport

consommer des ressources. Ainsi, l'agressivité globale $Ag(Prot)$ pour un protocole $Prot$ est définie par :

$$Ag(Prot) = max(Ag_{Prot}(t)) \tag{28.2}$$

Ainsi, la figure 28.4 représente sur le même graphique à la fois le débit moyen obtenu par chaque protocole de transport et son agressivité globale définie dans l'équation 28.2. Il apparaît clairement que l'ordre des niveaux de QdS par protocole est le même que l'ordre de leur niveau d'agressivité (excepté pour TCP New Reno) :

Ag(TCP New Reno) > Ag(TCP SACK) > Ag(TCP Tahoe) > Ag(TCP Vegas) > Ag(TFRC)

Ce résultat démontre que notre modèle de tarification est cohérent par rapport à la vision opérateur mais aussi par rapport au point de vue utilisateur. En effet, nous avons expliqué dans la section 28.2.1 comment les oscillations, c'est à dire l'agressivité, des protocoles de niveau transport perturbent le profil du trafic et réduisent les performances du réseau. Dans la présente section, nous avons démontré que l'agressivité, même si elle réduit les performances globales du réseau, aide à l'obtention de meilleures performances de façon individuelle. La seule exception concerne TCP New Reno, mais ces résultats de simulations ont l'intérêt de montrer que TCP New Reno n'est pas un protocole intéressant pour les utilisateurs et les opérateurs. En effet, il consomme une grande quantité de ressources pour un niveau final de QdS qui est plutôt limité. TCP New Reno apparaît donc comme un protocole peu efficace qui est responsable de beaucoup de problèmes rencontrés actuellement dans l'Internet : il contribue fortement à l'augmentation de la LRD et de l'auto-similarité dans le trafic, ces deux propriétés étant la cause des difficultés actuelles pour augmenter la QdS de l'Internet.

28.4 Mécanismes de tarification

Comme nous venons de le voir, facturer les services Internet en fonction du niveau d'agressivité des mécanismes de contrôle de congestion est une solution pour les opérateurs qui devraient recevoir également l'aval des utilisateurs. Néanmoins, il reste à définir les principes de mise en œuvre d'un tel mode de tarification. Pour le moment, résoudre ce problème constitue un travail à venir. Cependant, cette section propose plusieurs ébauches de techniques pour mettre en place une telle approche de tarification. Evidemment, seul les principes généraux seront donnés afin d'expliquer comment les différents services peuvent être facturés en fonction de l'agressivité des protocoles de transport. Les autres paramètres qui pourraient être nécessaires pour calculer le montant de la facture par utilisateur (quantité de données, débit, etc.) ne seront pas pris en compte dans ce qui va suivre.

28.4.1 Mécanisme de tarification par CdS

Cette première approche de tarification de service s'inspire de l'architecture DiffServ qui intègre un identifiant dans l'en-tête de chaque paquet de données véhiculé sur le réseau et qui indique à quelle CdS ce paquet appartient. La différenciation de service et bien sûr la tarification est ainsi réalisée en fonction de la valeur de cet identifiant.

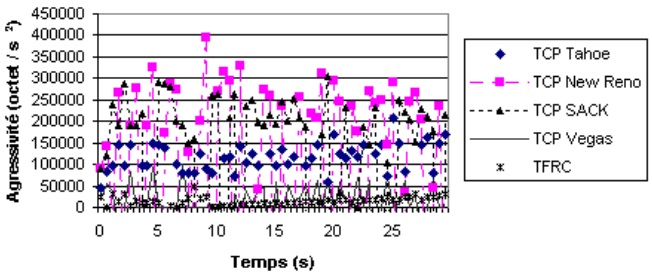


FIG. 28.3 – Agressivité des différents protocoles ($Ag_{Prot}(t)$)

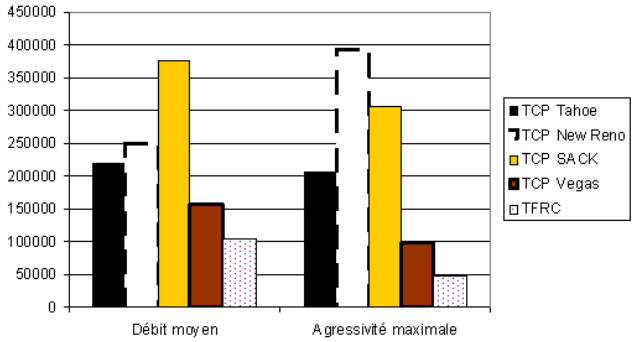


FIG. 28.4 – Comparaison entre débit (QdS) et agressivité maximale ($Ag_{Prot}(t)$) par protocole

Une approche similaire est possible dans notre cas. En fait, la solution serait de mettre dans l'en-tête de chaque paquet, un identifiant dont la valeur serait fonction du mécanisme de contrôle de congestion utilisé. Pour cela, nous pourrions utiliser le champ "protocol" de l'en-tête IP[16]. Ce champ identifie, à l'heure actuelle, le protocole qui a généré ce paquet : UDP[17], TCP, ICMP[18], etc. Une solution serait d'assigner un numéro de protocole particulier pour chaque version de TCP afin de représenter, bien sûr le protocole, mais surtout le mécanisme de contrôle de congestion utilisé. Ensuite, en réalisant des calculs statistiques sur l'agressivité des protocoles de transport les plus connus, il serait possible de définir le montant à appliquer pour n'importe quel paquet de n'importe quelle CdS pré-définie.

28.4.2 Tarification par flux

Cependant, la solution précédente fonctionne seulement pour des CdS pré-définies et tous les flux appartenant à une même CdS seront facturés au même prix, bien que l'agressivité d'un flux puisse être augmentée par des RTT courts. Dès lors, il semble intéressant de pouvoir réaliser une facturation de façon plus précise, par flux voire même par connexion TCP pour prendre en compte toute la dynamique du trafic Internet. Pour cela, il est nécessaire de pouvoir mesurer le niveau d'oscillation de chaque flux ou de chaque émetteur d'une connexion TCP. Les outils de métrologie, utilisés pour réaliser l'analyse de trafic présentée dans la section 28.2.1 nous semblent tout à fait adéquat. En effet, ces outils [24, 18] sont de plus en plus déployés dans l'Internet car ils représentent une nouvelle approche pour la gestion et la supervision des réseaux, l'ingénierie des trafics, etc. Nous sommes convaincus que ces outils de métrologie peuvent être tout aussi utiles pour réaliser une facturation des services Internet. De plus, ils peuvent représenter un moyen de contrôle du comportement des utilisateurs, et ainsi détecter les fraudeurs qui n'utiliseraient pas le mécanisme de contrôle de congestion pour lequel ils sont facturés.

Le principe de ces outils serait de calculer l'agressivité de chaque flux ou connexion en analysant la quantité maximale d'octets qui a été transmise dans un intervalle de temps donné correspondant

[16] IP : Internet Protocol
[17] UDP : User Datagram Protocol
[18] ICMP : Internet Control Message Protocol

à la granularité des mesures (comme nous l'avons détaillé dans la section 28.3.2). Cette granularité devra être choisie de façon suffisamment fine pour permettre de capturer toutes les oscillations, particulièrement celles présentes dans les hautes fréquences. L'agressivité pourra ainsi être mesurée individuellement pour un flux.

Chapitre 29

Conclusions

Le bilan des travaux des deux premières années dans le cadre du SP6 est tout a fait intéressant puisqu'il a permis de définir les principaux aspects d'un système de mesure des SLA pour RENATER (en liaison étroite avec les travaux faits dans les sous-projets 4 et 7), et qu'il a permis, en liaison avec les sous-projets 3 et 7, d'exploiter les résultats de caractérisation du trafic Internet pour proposer une évolution du modèle de tarification "smart market" pour l'Internet actuel.

La suite du projet va naturellement poursuivre les travaux sur les deux aspects mesure des SLA et tarification :

- Pour les SLA, la dernière année du projet sera utilisée pour déployer de façon opérationnelle des sondes de mesure active permettant de mesurer en continu la QdS offerte entre tous les noeuds du réseau. Le travail à réaliser consistera donc à sélectionner sur le marché la solution la mieux adaptée. En particulier, une contrainte qui restait secondaire pour le SP4 – la facilité de pouvoir exploiter les résultats de mesure en temps réel, en générant des alarmes par exemple – sera ici un des éléments déterminants du choix, parmi les solutions qui offrent les capacités de mesure requises.
- Les travaux autour de la nouvelle approche seront approfondis, notamment sur les aspects de la mise en œuvre. En effet, aujourd'hui, seuls les principes de cette mise en œuvre ont été abordés, et ils doivent être davantage développés dans l'optique d'être exploitables dans un réseau réel. En particulier, un des problèmes qui se pose, dès lors que l'on considère un réseau à haut débit comme le réseau RENATER, concerne la capacité du système de mesure à traiter en temps réel tout le trafic. Aussi, pour répondre à ce problème, nous allons étudier des techniques d'échantillonnage afin de limiter la quantité de données à traiter. L'objectif de cet échantillonnage est naturellement de limiter de façon drastique la quantité de données à analyser sans perdre d'information quant à la tarification. En particulier, et comme cela a été analysé dans le SP3 et exploité ici pour proposer une nouvelle approche de tarification, les caractéristiques problématiques pour l'écoulement du trafic apparaissent avec les flux éléphants (alors que les flux souris s'écoulent sans difficulté). Les techniques d'échantillonnage à développer devront donc permettre de distinguer tous les flux longs (éléphants), même si de l'information sur les flux souris est perdue, puisque la politique de tarification de RENATER (et des opérateurs Internet en général) se base sur un débit maximum d'utilisation. Les utilisateur qui

dépassent leur contrat[1], causent du tort au réseau et à la QdS qu'il rend. Ils doivent donc être repéré malgré l'échantillonnage et si besoin sanctionné (par exemple par une surfacturation). Les recherches sur les techniques d'échantillonnage composeront le thème central de l'activité de recherche au sein du SP6 au cours de la dernière année du projet. Toutefois, les mécanismes de tarification autour du "smart market" proposés dans ce rapport seront adaptés en fonction des différents échantillonnages utilisés. De plus, le principe sera affiné pour pouvoir continuer à fonctionner de façon juste et cohérente avec d'éventuelles pertes d'information au niveau des flux échantillonnés.

[1] défini par les agréments dans le cas de RENATER

Bibliographie

[1] Morton A. Linux scheduling latency. http ://www.zip.com.au/ akpm/linux/schedlat.html.

[2] M. Allman, V. Paxson, and S. Stevens. Tcp congestion control. *RFC*, (2581), April 1999.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *RFC*, (2475), December 1998.

[4] L.S. Brakmo, S.W. O'Malley, and L. Peterson. Tcp vegas : New technique for congestion detection and avoidance. In *ACM SIGCOMM*, London, October 1994.

[5] R. Casellas, T. Friedman, F. Roueff, K. Salamatian, G. Texier, T. Chahed, and F. Guillemin. Metropolis – sous-projet 4 – echantillonnage et mesures actives. Technical report, Contrat RNRT METROPOLIS, Janvier 2004.

[6] CISCO. Service assurance agent (saa). http ://www.cisco.com/warp/public/732/Tech/nmp/saa/.

[7] J. Corral, M. Ourraou, G. Texier, and L. Toutain. Saturne : Une architecture pour mesurer les performances des classes de service ip, 2002.

[8] RUDE & CRUDE. Rude & crude. http ://rude.sourceforge.net/.

[9] DANTE. Sequin. http ://www.dante.net/sequin/.

[10] Dipartimento di Ingegneria Aerospaziale Politecnico di Milano. Diapm rtai - realtime application interface. http ://www.rtai.org.

[11] C. Dovrolis and A. Danalis. Anemos : An autonomous network monitoring system. In *PAM*, 2003.

[12] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *ACM SIGCOMM*, 2000.

[13] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, K. Papagiannaki, and F. Tobagi. Design and Deployment of a Passive Monitoring Infrastructure. In *Passive and Active Measurement Workshop*, Amsterdam, April 2001.

[14] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, (35 pp. 1969-1985), 1999.

[15] Kiszka J. Rtnet. http ://www.rts.uni-hannover.de/rtnet/.

[16] F. P. Kelly, A. Maulloo, and D. Tan. Control for communication networks : Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, (49 pp. 237-252), 1998.

[17] FSM Labs. Fsmlabs - the rtlinux company. http ://www.rtlinux.org/.

[18] M. Luckie, A McGregor, and H. Braun. Towards improving packet probing techniques. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, California, USA, November 1st/2nd 2001.

[19] Love R. M. Linux kernel patches | rml. http ://www.tech9.net/rml/linux/.

[20] Zekauskas M. Measurements on internet2. measurement.internet2.edu/presentations/2001-04-19-NRW-Meas.pdf, Avril 2001.

[21] M3I. Projet «market managed multiservice internet» - site web : http ://www.m3i.org.

[22] J. Mackie-Mason and H. Varian. *Pricing the Internet.* in public access to the Internet, B. Kahin and J. Keller eds, Englewood Cliffs, NJ ; Prentice-Hall, 1995.

[23] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov. Tcp selective acknowledgement options. *RFC*, (2018), October 1999.

[24] J. Michel, I. Graham, and Donnelly S. Precision timestamping of network packets. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, California, USA, November 1st/2nd 2001.

[25] NLANR. Nlanr es : Nimi. http ://www.ncne.nlanr.net/nimi/.

[26] NS-2. Site web : http ://www.isi.edu/nsnam.

[27] NTP.org. http ://www.ntp.org/ntpfaq/NTP-s-sw-clocks.htm.

[28] P. Olivier and N. Benameur. Flow level ip traffic characterization. *Proc. of ITC'17 Moreira de Souza, Fonseca and de Souza e Silva (eds.)*, December 2001.

[29] P. Owezarski, FX. Andreu, C. Fricker, K. Salamatian, C. Chekroun, N. Benameur, P. Olivier, J. Roberts, P. Robert, and F. Guillemin. Projet metropolis. sous-projet 1 : Rapport d'état de l'art. Technical report, Contrat RNRT METROPOLIS, Janvier 2003.

[30] P. Owezarski and Larrieu N. Metropolis – sous-projet 7 – conception et mise en place de la plate-forme de mesures passives. Technical report, Contrat RNRT METROPOLIS, Janvier 2004.

[31] P. Owezarski, Larrieu N., and Soule A. Metropolis – sous-projet 3 – analyse du réseau. Technical report, Contrat RNRT METROPOLIS, Janvier 2004.

[32] Ficheux P. and Kadionik P. Linux temps réel. *Linux magazine*, Juillet-Août 2003.

[33] Mourot P. Rtai internal presentation. http ://www.aero.polimi.it/ rtai/documentation/articles/patric_mourot-rtai_internal_presentation.html.

[34] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *IEEE ICNP*, 1996.

[35] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. In *SPIE International Conference on Performance and Control of Network Systems*, November 1997.

[36] K. Park and W. Willinger. *Self-similar network traffic : an overview*. In Self-similar network traffic and performance evaluation, J.Wiley & Sons, 2000.

[37] A. Pasztor and D. Veitch. Pc based precision timing without gps. http ://www.cubinlab.ee.mu.oz.au/ darryl/tscclock_final.pdf, Juin 2002.

[38] QNX. Qnx realtime operating system (rtos) software, development tools, and services for embedded applications. http ://www.qnx.com.

[39] QOSMetrix. http ://www.qosmetrix.com/.

[40] Brun R. and Rademakers F. The root system home page. http ://root.cern.ch.

[41] RIPE. Ripe-ncc/test traffic measurements. http ://www.ripe.net/ttm/.

[42] Ubik S. and Smotlacha V. Precise measurement of one-way delay and analysis of synchronization issues. http ://staff.cesnet.cz/ ubik/publications/2002/qosplot.pdf, 2002.

[43] N. Simar. Less than best effort. http ://archive.dante.net/geant/presentations/LBE - QoS IP 2003.ppt, Februar 24 2003.

[44] F. Simon. Implémentations des classes de services dans renater-3. *Actes du congrès JRES2003*, 2003.

[45] Oetiker T. Mrtg : The multi router traffic grapher. http ://people.ee.ethz.ch/ oetiker/webtools/mrtg/.

[46] Oetiker T. Rrd tool. http ://people.ee.ethz.ch/ oetiker/webtools/rrdtool/.

[47] Windriver. Wind river - vxworks ae. http ://www.windriver.com/products/vxworks_ae/index.html.

Sixième partie

Sous Projet 7

Architecture de mesure

METROPOLIS

Métrologie Pour L'Internet

Projet RNRT Exploratoire

Sous Projet 7
Architecture de mesure

Livrable
Rapport d'avancement

Coordinateur du sous-projet : Philippe Owezarski (LAAS).
Auteurs : LAAS : Philippe Owezarski et Nicolas Larrieu
LIP6 : Timur Friedman, Augustin Soule
GET : Tijani Chahed

2003/10/16

Laboratoire d'Analyse et d'Architecture des Systèmes LAAS
Laboratoire d'Informatique de Paris 6 LIP6
Groupe des Ecoles de Télécommunications GET

Chapitre 30

Architecture de mesure active

Chapitre 31

La Plate-forme NIMI / MINC

31.1 Introduction

Le projet NIMI a pour objectif le déploiement d'une infrastructure nationale (au niveau des Etats-Unis) de mesures actives. Cette infrastructure est flexible et permet le recueil de diverses mesures actives. Cette infrastructure a été utilisée durant les deux ou trois années passées pour plusieurs campagnes de mesures, dont la détermination d'une matrice de distances dans Internet. L'infrastructure NIMI s'est aussi étendue en Europe.

Le projet MINC utilise la diffusion de sondes actives par le biais du multicast pour inférer sur la structure interne du réseau. Certaines des idées et méthodes développées dans le cadre de ce projet sont des bases pour la problématique de l'échantillonnage que nous souhaitons développer dans le cadre de METROPOLIS

Chapitre 32

La Plate-Forme METROPOLIS-RIPE

32.1 Introduction

RIPE NCC (Réseaux IP Européens - Network Coordinaton Center) est un des quatre RIR (Regional Internet Registries) qui existe dans le monde, les autres étant ARIN (American Registry for Internet Numbers), APNIC (Asia Pacific Network Information Centre) et LACNIC (Latin American and Caribbean IP address Regional Registry). Il a pour mission de fournir les services nécessaires à l'opérabilité de l'Internet, principalement en Europe, au Moyen Orient et en Afrique.

Le service TTM (Test Traffic Measurements) de RIPE NCC a été initié en Octobre 2000 pour offrir à la communauté Internet un service de métrologie des liens Européens et autres. Cette activité se déroule selon la méthode active, en accord avec les standards du groupe de travail IPPM (IP Performance Metrics) [1] de l'IETF.

32.2 RIPE TTM

32.2.1 Les sites membres

Une centaine de sites participent à cette activité de métrologie active. La majorité de ces sites se trouvent en Europe. Quelques sites existent néanmoins en Amérique du Nord, un site au Moyen Orient (Israel), un site en Asie (Japon), un site en Australie et un site en Nouvelle Zélande. Ces sites sont typiquement des universités, des laboratoires de recherche ainsi que des opérateurs et fournisseurs de service Internet (ISP).

Cela étant, les résultats des mesures sont aussi fournis à des régulateurs des télécommunications et à diverses corps gouvernementaux qui veillent sur l'intérêt public dans ce domaine.

32.2.2 Les boîtiers de mesures

Les boîtiers de mesures sont des PCs industriels dédiés, relié à un équipement GPS (Global Positionning System) qui sert à synchroniser les horloges. Le GPS est un système de navigation de satellite développé par le US Department of Defence (DoD). Les récepteurs reçoivent un signal de temps et de positionnement de la part de 24 satellites. Le temps ainsi obtenu est précis jusqu'à 200 μs.

Les boîtiers sont installés sur la DMZ (Demilitarized Zone), à l'extérieur des réseaux locaux des sites membres ; le but étant de mesurer l'état des liens entre les différents partenaires et non pas leur activité interne.

32.2.3 Le trafic test

Chaque boîtier envoie du trafic test en continu vers tous les autres sites. Le trafic test consiste en des paquets UDP, qui peuvent être de trois tailles différentes, petite (56 bytes, comme les outils ping et autres), moyenne (576 bytes, la taille minimale qu'un routeur doit traiter en tant que paquet individuel) et grande (2048 bytes par exemple afin d'étudier les effets d'une possible fragmentation des paquets). Les paquets de petite et moyenne taille sont envoyée toutes les minutes, les grands paquets sont envoyés chaque dix minutes. Ceci génère en moyenne un trafic de 14 octets par seconde pour chaque mesure, sans compter les overheads et les données de traceroute. Ceci est un compromis entre la précision des mesures (qui nécessite l'envoi d'un grand nombre de paquets) et la non altération de l'état réel du réseau.

Chaque paquet test contient entre autres une estampille qui indique le moment d'envoi et de réception du paquet, essentiel pour le calcul des paramètres temporels, ainsi que des informations sur les horloges permettant une quantification de l'erreur liée à l'horloge. Il contient aussi un *reference number* afin de garder un suivi des paquets envoyés et éventuellement perdus. Il contient enfin le nombre de sauts entre l'émetteur et le récepteur.

32.2.4 Les paramètres mesurés

RIPE TTM mesures les paramètres relatifs à la connectivité [8] entre chaque pair de sites, le délai [4], la perte [5], la gigue [7] ainsi que les vecteurs de routage.

D'autres paramètres, généralement inspirés du groupe de travail IPPM sont en cours de développement. Chaque RFC est dédié à un paramètres donné et contient une définition du paramètre, une ou des méthodologies de mesures ainsi que des discussions sur les horloges.

32.2.5 Les résultats des mesures

Les résultats des tests sont consultable uniquement par les membres de TTM et sont disponibles au lien suivant :

http ://www.ripe.net/ttm/Plots/

Les données sont sous formes de courbes représentant les délais, la gigue et les pertes entre chaque deux sites sur les dernières 24 heures, semaine et mois. Les membres de RIPE TTM peuvent aussi spécifier des périodes arbitrairement choisies pour l'étude des données. Des résumés indiquant des quantiles et des médianes sur les délais et taux de perte sont aussi disponibles. Des informations concernant l'état des horloges au moment où les mesures sont effectuées sont aussi disponibles.

32.2.6 Les nouvelles activités

Parmi les nouveaux services offerts par RIPE TTM, nous pouvons citer les mesures configurées par les usagers ainsi que le suivi d'incident majeurs sur Internet.

La première activité doit permettre aux membres de TTM de spécifier leurs mesures en terme de débit d'échantillonage et de taille de paquets. Dans le futur, des champs relatifs à la qualité de service peuvent même être activés.

Le suivi d'incident majeur sur l'Internet, par exemple le Sapphire/Slammer Worm [3] du 22 Janvier 2003, permet d'analyser et de quantifier son impact sur l'Internet.

32.3 METROPOLIS-RIPE

32.3.1 Sites participants

Parmi les membres du projet Metropolis, les sites suivants hébergent un boîtier de mesure RIPE : INT (Institut National des Télécommunications, Evry), LIP6 (Laboratoire Informatique Paris 6, Paris), ENST (Ecole Nationale Supérieure des Télécommunications, Paris), LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse) et Eurecom (Sophia Antipolis). Renater est en train d'étudier un éventuel achat de boîtier RIPE. Nous appellerons cette plate-forme METROPOLIS-RIPE.

32.3.2 Etat présent de METROPOLIS-RIPE

Dans un premier temps, nous avons opté pour un *outsourcing* de notre plate-forme pour RIPE TTM. Cela veut dire que chaque membre achète un boîtier RIPE (environ 2500 euros) et un service de maintenance, d'administration des boîtiers ainsi qu'une collecte et mise en forme des résultats par RIPE TTM (environ 2500 euros par an).

Chaque membre a le choix de limiter l'envoi et la réception du trafic test entre les membre du projet Metropolis ou pas. Les résultats des tests entre les membres du projet sont publiés dans une section privée, visible uniquement par les membres du projet Metropolis, au lien suivant :

http ://www.metropolis.ripe.net/

L'avantage de cette première configuration est que RIPE TTM se charge de tout et que les expérimentations ont pu démarrer très tôt. Les résultats des mesures sont aussi immédiatement disponibles. L'inconvénient est que les résultats vont avoir le même format standard RIPE. En plus, dans ce cas, nous devons payer des frais annuel de maintenance.

32.3.3 Etat futur de METROPOLIS-RIPE

L'idéal pour notre projet est de construire une plate-forme de mesures actives autonome, c'est à dire indépendante de RIPE TTM. La plus grande difficulté de cette solution réside dans l'installation et la maintenance d'un serveur central de collecte et d'analyse de données. Le portage du code de chez RIPE ainsi que sa personalisation pour nos propres besoins peut être très compliquée et peut prendre beaucoup de temps et de ressources avant d'être opérationnel. L'avantage de cette perspective est que nous pouvons tout personaliser à souhait mais l'inconvénient est que nous devons tout faire nous même, y compris la maintenance.

32.4 Autres directions

32.4.1 NIMI

En attendant, nous avons installé un autre logiciel de mesures actives, à savoir NIMI (National Internet Measurement Infrastructure) [10], sur les boîtiers RIPE. Ceci permet un contrôle avancé des mesures ainsi qu'un accès à beaucoup de sites de par le monde.

32.4.2 Pandora

Nous envisageons pour la suite d'installer Pandora [2], un outil qui nous semble prometteur pour nos besoins de métrologie active.

Chapitre 33

Architecture de mesure passive

33.1 Introduction

La métrologie passive qui a été précisément décrite dans [9] consiste à capturer des informations sur le trafic qui passe en un ou plusieurs points d'un réseau afin de pouvoir le ou les analyser. Cette approche pour réaliser des mesures correspond à une vision opérateur de supervision du réseau et des services qu'il fournit. Naturellement, le type des informations qui sont collectées peut être de natures très diverses, allant de quelques statistiques sur la quantité de trafic qui passe en ce point (vision macroscopique du trafic) jusqu'à la collecte d'une trace d'information de tous les événements se produisant sur le réseau, c'est-à-dire garder une trace du passage de tous les paquets individuellement (vision microscopique du trafic). Le types d'informations collectées a une influence capitale sur le mode d'analyse qui peut être réalisé. Dans le premier cas, les analyses sur quelques paramètres précis peuvent se faire en temps-réel et permettre des réactions très rapides aux événements observés. Dans le second cas, en revanche, l'analyse ne pourra se faire qu'a posteriori, et permettra des analyses en temps différé, en théorie sans limite de complexité.

Dans le cadre de METROPOLIS qui initie cette thématique de recherche dans le secteur académique en France, avec des objectifs ambitieux notamment dans le domaine de la caractérisation et la modélisation du trafic, il est apparu judicieux d'observer le trafic avec ces deux angles de vue, micro et macroscopique. Pour l'analyse microscopique, la solution retenue, et qui fait l'objet de ce rapport, est basée sur les cartes DAG [6]. Pour l'analyse macroscopique la solution retenue est basée sur les sondes QoSMOS qui sont décrites dans [rapport QoSMOS ?].

La suite de ce rapport va donc présenter les besoins de METROPOLIS en matière de métrologie passive (partie 2) et décrire la solution DAG retenue (partie 3). La partie 4 présentera ensuite la carte de déploiement des sondes à l'heure actuelle, alors que la partie 5 évoquera les problèmes rencontrés ainsi que les nouvelles contraintes qui sont apparues et les solutions que nous y avons apportées. Enfin la partie 6 conclura ce rapport.

33.2 Premières contraintes et besoins

La principale contrainte qui se pose pour l'installation de sondes de mesure est due au fait que le réseau dont nous souhaitons analyser le trafic est un réseau opérationnel, et que malgré la présence de la sonde, ce réseau doit continuer à fonctionner sans aucune dégradation du service qu'il offre. Le premier besoin pour le système de mesure à mettre en place est donc une transparence totale pour le réseau et son trafic. Cela signifie que pour être non intrusif, cet équipement ne devra pas provoquer de pannes, d'erreurs de transmission et ne pas introduire de délais pour ne pas modifier le profil du trafic.

Le second besoin lors du choix des sondes de mesure passive concerne sa précision et la validité des traces qu'elle produira. Ainsi, il est essentiel de ne pas « manquer » de paquets transitant sur le réseau, et d'avoir des informations précises sur le passage de ces paquets, notamment au niveau temporel, qui représente aujourd'hui une des difficultés majeures avec les systèmes actuels. Le système devra donc être bien dimensionné et offrir une horloge précise qui ne dérive pas.

Enfin, le troisième besoin qui apparaît concerne la possibilité de corréler des événements de plusieurs traces, par exemple de suivre un paquet en plusieurs points du réseau, ou d'analyser de façon croisée le passage des paquets et de leurs acquittements, etc. Pour pouvoir finement analyser de tels événements se produisant en des points géographiquement distants et à des instants distincts mais faiblement éloignés temporellement, il est nécessaire de disposer d'une base temporelle commune et universelle pour toutes les sondes.

33.3 La solution DAG

Pour répondre à ces besoins (transparence, précision temporelle, temps universel), la solution existante la mieux adaptée est indéniablement une solution basée sur les cartes DAG conçues et développées à l'université de Waikato en Nouvelle-Zélande et, à l'heure actuelle, commercialisées, maintenues et améliorées par la société ENDACE. Le principe de fonctionnement des sondes DAG est décrit sur la figure 33.1. Le premier avantage de cette carte est de pouvoir travailler en dérivation du lien à analyser. Ainsi, dans le cadre de réseaux sur fibres optiques, le principe de branchement de la sonde consiste à insérer un « splitter » optique qui laisse passer 80 % de la puissance optique sur la fibre originelle (chemin normal), et récupère 20

De son côté, la carte DAG est une carte dédiée qui réalise, en temps-réel, l'extraction des entêtes de tous les paquets qui passent sur le lien. La taille de l'entête est précisée au moment de la configuration de la carte pour la capture. Dans notre cas, nous souhaitons pouvoir capturer les entêtes IP et TCP. Enfin, pour chaque paquet capturé, la carte ajoute une estampille codée sur 64 bits à l'entête capturée. Le tout est ensuite stocké sur disque. Il est à noter que les traces ainsi constituées deviennent rapidement très volumineuses, surtout sur les réseaux à hauts débits, et nécessitent donc d'utiliser des disques de grandes capacités et en nombres suffisants. Toutes les machines sont donc équipées de 3 disques durs de 73 Go.

Pour la même raison, le trafic qui transite entre la carte DAG et le disque dur de la station hôte est très élevé, et pour les réseaux aux capacités les plus fortes, les bus PCI classiques des ordinateurs habituels ne suffisent pas. Il est nécessaire dans ce cas d'utiliser la dernière génération de bus PCI, à savoir les bus 64 bits à 66 MHz qui offrent des bandes passantes bien supérieures aux bus PCI

traditionnels de 32 bits cadencés à 33 MHz. Tous ces éléments (carte dédiée temps-réel, bus haute capacité, mémoire importante et disques durs de grandes capacités) sont les éléments indispensables pour garantir un système bien dimensionné capable de capturer une trace de tous les paquets ayant transité sur le lien mesuré.

En ce qui concerne l'estampille de passage de chaque paquet, stockée avec l'entête du paquet, une référence GPS est utilisée. La carte est en effet directement reliée à une antenne GPS. Ainsi, l'horloge de la station qui héberge la carte DAG est resynchronisée chaque seconde sur un signal GPS qui transporte le temps universel venant des horloges atomiques de référence. Ainsi, la dérive de l'horloge est quasiment inexistante, garantissant une grande précision des mesures temporelles, ainsi que le temps universel, car les sondes seront effectivement synchronisées sur le temps de référence universel.

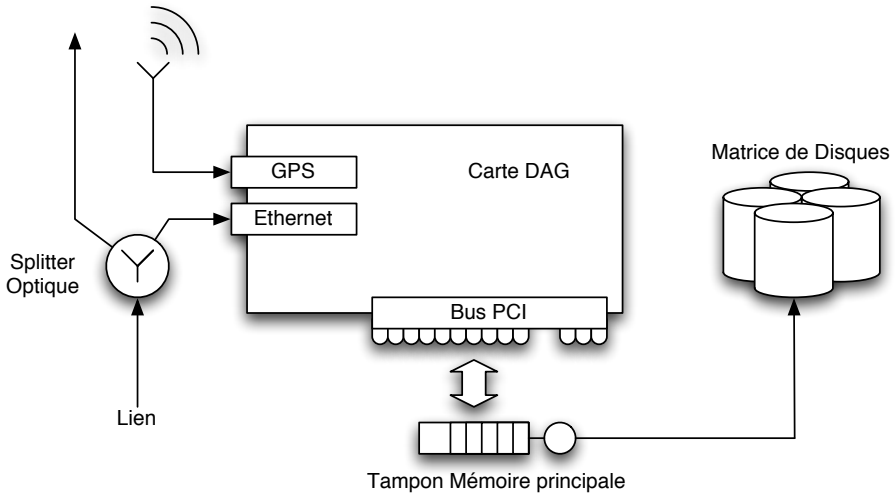


FIG. 33.1 – Principe opératoire des cartes DAG

Respectant ces principes de conception, la sonde conçue pour capturer le trafic sur un lien Giga-Ethernet est détaillée sur la figure 33.2.

De la même façon, la sonde pour capturer le trafic sur des réseaux Fast-Ethernet est détaillée sur la figure 33.3. A noter que dans ce cas, le support physique est de la paire torsadée. Il n'y a donc pas de « splitter », inutile avec les propriétés naturelles de propagation de l'électricité, et un système de « bypass » le remplace pour garantir un bon fonctionnement du réseau même si la sonde s'arrête.

Il est à noter que cette solution DAG a déjà été mise en place en de nombreux points sur le réseau commercial de SPRINT, sur des liens SONET. Et ces équipements se sont avérés très performants autant en termes de transparence que de précision. C'est pour nous un gage de sécurité et de satisfaction, même si nous devons déployer dans notre cas des équipements beaucoup plus récents fonctionnant sur Fast et Giga-Ethernet.

Enfin, pour analyser les traces capturées par les sondes, une plate-forme de stockage des traces et d'analyse a été conçue et mise en place. Cette plate-forme est hébergée au LAAS à Toulouse et ouverte aux partenaires de METROPOLIS. Les besoins de cette plate-forme sont donc essentiellement

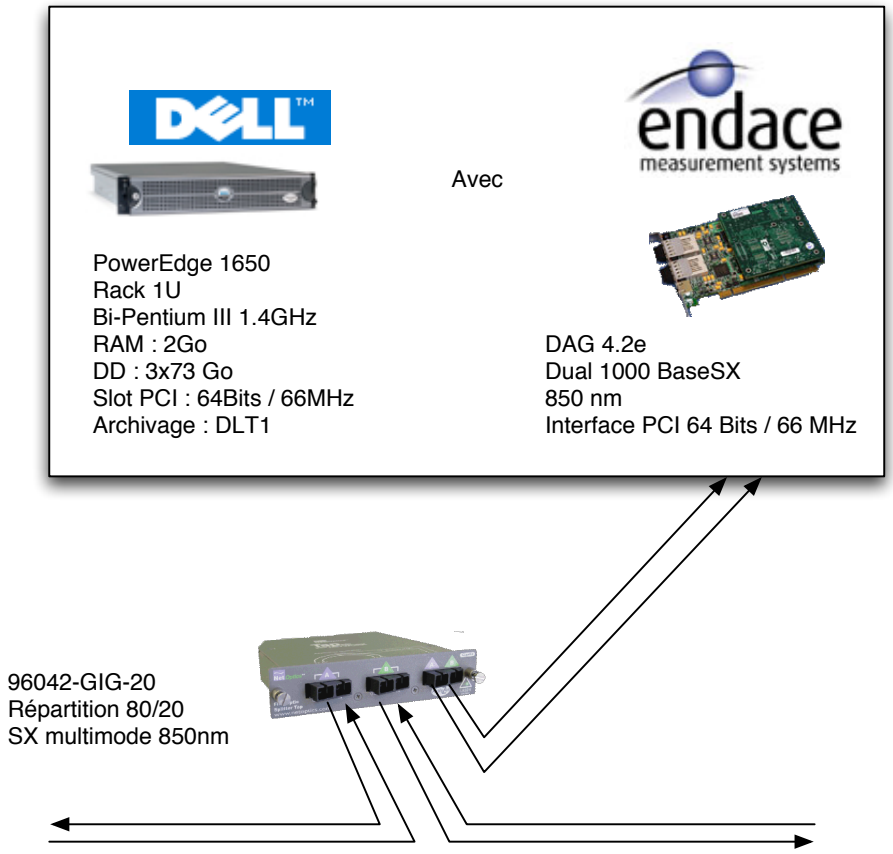


FIG. 33.2 – Conception de la sonde de métrologie passive pour un lien Giga-Ethernet

une grande capacité de stockage, et une grande capacité de traitement (processeurs et mémoire essentiellement). Cette plate-forme est donc composée de 2 PowerEdge 4600 dont les caractéristiques sont :

- Poweredge 4600
- Rack 6U
- Bi-Pentium Xeon 2.4 Ghz
- RAM : 8 GB
- HD : 8 x 73 GB
- Archivage : DLT1

D'autre part, il a fallu penser à un système pour rapatrier les fichiers de traces des sondes de mesures vers la plate-forme d'analyse. La solution idéale aurait été de pouvoir utiliser les réseaux académiques, mais face à la charge supplémentaire qu'aurait représenté ces transferts, certains réseaux académiques auraient eu du mal à tenir. Il a donc été décidé d'équiper toutes ces machines avec des lecteurs de bandes au format DLT1, et nous faisons ces transferts en envoyant les bandes par les services postaux. Cette solution a aussi l'avantage de pouvoir utiliser les bandes comme sys-

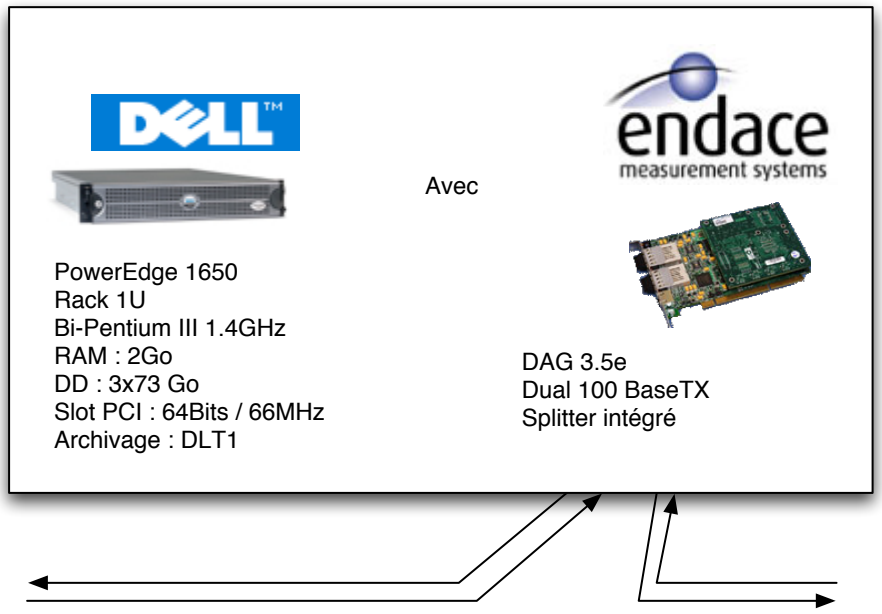


FIG. 33.3 – Conception de la sonde de métrologie passive pour un lien Fast-Ethernet

tème d'archivage des traces que nous n'utilisons plus pendant quelques temps, ce qui a permis de réduire la capacité des disques de la plate-forme d'analyse, et de réduire sensiblement son prix.

33.4 Carte de déploiement des sondes DAG

Nous avons demandé à plusieurs responsables des réseaux académiques français l'autorisation de déployer nos équipements de mesure sur le réseau qu'ils opèrent. Notre objectif était de pouvoir déployer ces sondes sur des réseaux de natures différentes, soit sur le réseau de c'ur, sur les réseaux régionaux et à la sortie des laboratoires. Toutes les autorisations ne nous ont pas été accordées pour des raisons sur lesquelles nous reviendront dans la partie suivante. Toutefois, aujourd'hui, 3 sondes DAG ont été déployées et respectent quasiment le schéma que nous nous étions fixés. Ainsi, comme le montre la figure 33.7, deux sondes en Fast-Ethernet ont été positionnées à la sortie de deux grands laboratoires que sont le LAAS (figure 33.4) et le LIP6 (figure 33.5). La troisième sonde en Giga-Ethernet a, quant à elle, été positionnée à la sortie du réseau de Jussieu sur RAP (figure 33.6), nous permettant ainsi d'avoir accès aux traces du trafic d'un réseau à très haut débit.

Ce choix de positionnement est aussi stratégique par rapport au positionnement des sondes de métrologie passive macroscopique et de sondes de métrologie active dont un exemplaire de chacune d'entre-elles ont été placées également au LAAS et au LIP6 (figure 33.7). Ainsi, il sera possible, pour un même trafic de corréler les analyses micro- et macroscopiques, ainsi que les mesures actives et passives, et ce en plusieurs points du réseau sur leur chemin entre Toulouse et Paris.

Enfin, nous rappelons que la plate-forme d'analyse est hébergée par le LAAS à Toulouse.

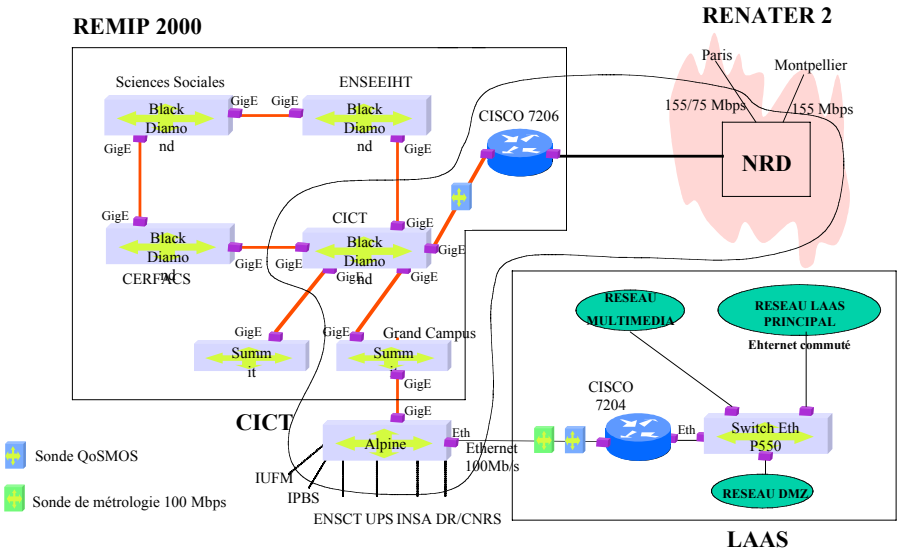


FIG. 33.4 – Schéma de déploiement sur la plate-forme toulousaine

33.5 Problèmes rencontrés / contraintes

Le premier problème rencontré est apparu lors de la définition des types de sondes à mettre en place. Ce choix a été retardé car la plupart des réseaux académiques français étaient en phase de migration ou envisageaient de migrer dans un futur plus ou moins éloigné. Ainsi Renater est passé de sa version 2bis à sa version 3 au printemps 2003, entraînant une modification de technologie support pour le réseau. De la même façon, Remip est passé à une version Giga-Ethernet à la place d'une version ATM à peu près à la même époque, entraînant un certain nombres d'incertitudes quant à la technologie de réseau qui serait utilisée pour raccorder le LAAS à ce nouveau réseau Remip 2000. Enfin, au printemps 2003, le lien d'interconnexion du réseau du campus de Jussieu à RAP a également été modifié pour migrer d'une connexion ATM OC-12 à un lien Giga-Ethernet. Des délais ont donc été induits car nous n'avons commandé les différents équipements que lorsque nous avons obtenu des garanties quant aux technologies réseau qui seraient employées, de sorte à acheter les cartes DAG pour la bonne technologie réseau.

D'un point de vue technique, les problèmes rencontrés avec l'installation et la mise en place des sondes DAG ont dans un premier temps été liés à la nouveauté de ce type d'équipement pour les réseaux Ethernet. En effet, même si la société ENDACE possède une expérience solide avec les cartes DAG sur SONET ou ATM, les cartes pour Ethernet, et notamment Giga-Ethernet, n'ont été disponibles que tardivement, avec des retards de fabrication et de livraison importants. D'autre part, cette « jeunesse » est allée de pair avec quelques difficultés pour la mise en place de ces équipements et leur réglage. Ainsi, le branchement de l'antenne GPS sur la carte DAG a nécessité des travaux de câblage imprévus. De la même façon, les cartes DAG pour réseaux Giga-Ethernet, de part la quantité d'informations qu'elles capturent sur ces liens Ethernet Gigabits, nécessitent des capacités de communications internes aux machines qui les hébergent très importantes qui vont au delà de ce

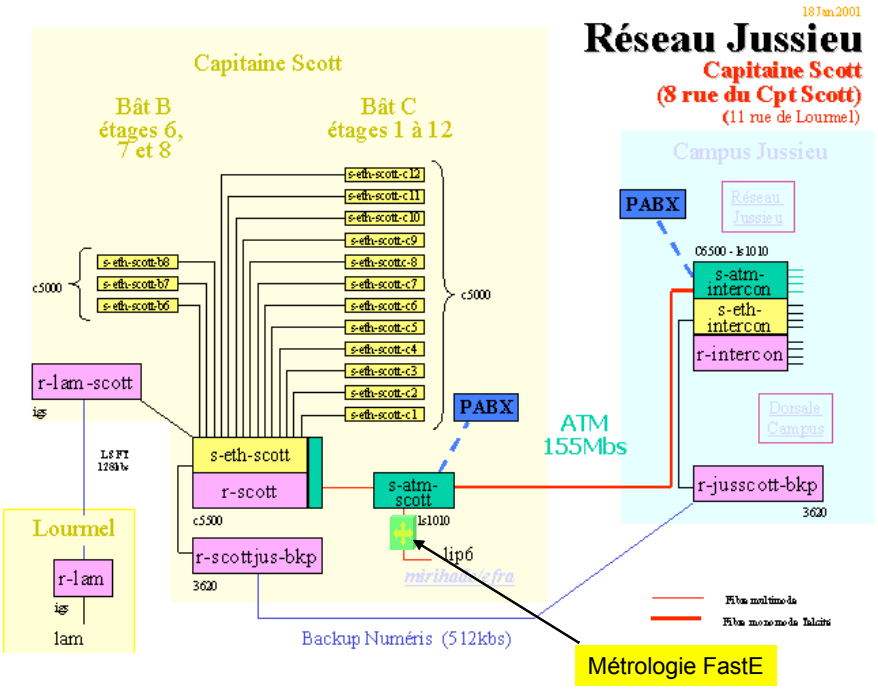


FIG. 33.5 – Schéma de déploiement au LIP6

que l'on pouvait trouver sur les serveurs PC disponibles en 2002. Ces cartes nécessitent donc d'être connectées à des bus PCI 64 bits / 66 MHz qui ne sont apparus sur les stations DELL qu'au début de l'année 2003 (les machines DELL étant imposées par les marchés nationaux du CNRS, car passer de nouveaux marchés pour METROPOLIS aurait pris beaucoup de temps et aurait retardé le projet).

Sur un plan logiciel, ces nouvelles cartes DAG pour Ethernet ont aussi éprouvé la mise en place de nouveaux formats de traces, éloignés des formats des traces capturées par Sprint sur leur réseau, et sur lesquelles certains partenaires de METROPOLIS avaient eu l'occasion de travailler. La configuration de la carte pour qu'elle génère des traces dans un format qui nous convenait a ainsi posé quelques problèmes, en particulier dans la création de passerelles entre le format DAG et le format TCPdump pour lequel il existe déjà une importante bibliothèque de logiciels d'analyse et que nous avons souhaité pouvoir utiliser. La difficulté est venue de la nouvelle nature dynamique du format de stockage des traces DAG. Naturellement, la configuration logicielle de la carte et du système pour pouvoir exploiter les estampilles GPS avec un maximum de précision a aussi été délicate.

Au delà de ces problèmes techniques, une nouvelle contrainte venant des administrateurs des réseaux qui hébergeaient nos sondes est apparue. Au delà de la non intrusivité de nos sondes qui a rapidement été avérée et donc garantissant que les performances des réseaux dont le trafic était analysé ne seraient pas perturbées, les administrateurs ont demandé à ce que les traces soient anonymes.

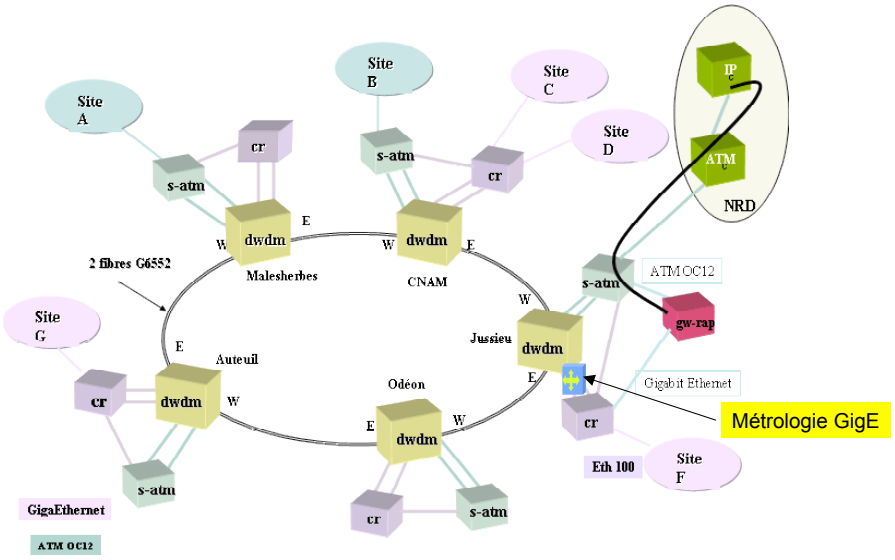


FIG. 33.6 – Schéma de déploiement à Jussieu

D'ailleurs, les refus que nous avons essuyés parfois lors de nos demandes d'installation de sondes sur certains réseaux ont été motivés par l'aspect confidentiel des communications. Rendre les traces anonymes signifie que les adresses des sources et destinations des paquets dont nous capturons les entêtes soient anonymisées, de façon à ne pas pouvoir savoir quels en étaient les auteurs et destinataires. De plus, l'anonymisation nécessite la suppression des données utilisateurs contenues dans la partie applicative de chaque paquet de données prélevés sur le réseau. Naturellement, ces informations portent un certain nombre d'informations qui peuvent être utiles pour découvrir les topologies des réseaux, les usages que font certains du réseau, les protocoles de routage, etc. De tels aspects ne pourront donc pas être traités dans la suite de METROPOLIS. Toutefois, pour pouvoir tout de même ne pas trop limiter les analyses et études que l'on peut faire à partir des traces capturées, il nous a fallu mettre en place un mécanisme d'anonymisation des adresses déterministe (pour pouvoir continuer à reconnaître les flux de l'Internet par exemple) et surtout qui ne casse pas la structure des adresses, dont les différents octets ne sont pas attribués de façon aléatoire. Un algorithme décrit dans [11] [12] a ainsi été mis en place non sans difficulté, en particulier à cause du nouveau format de stockage des traces DAG, qui est dans cette nouvelle version des cartes DAG un format dynamique.

33.6 Conclusion

Au final, la plate-forme de mesure passive existe et a été déployée, même si nous aurions souhaité mettre en place une à deux sondes de plus. D'autre part, tous les problèmes soulevés ont été résolus, qu'il s'agisse des problèmes techniques liés à ce matériel, ou aux contraintes d'anonymisation demandées par nos partenaires administrateurs de réseaux. Sur le plan de nos besoins, ces sondes remplissent parfaitement leur cahier des charges : elles sont totalement transparentes, extrêmement

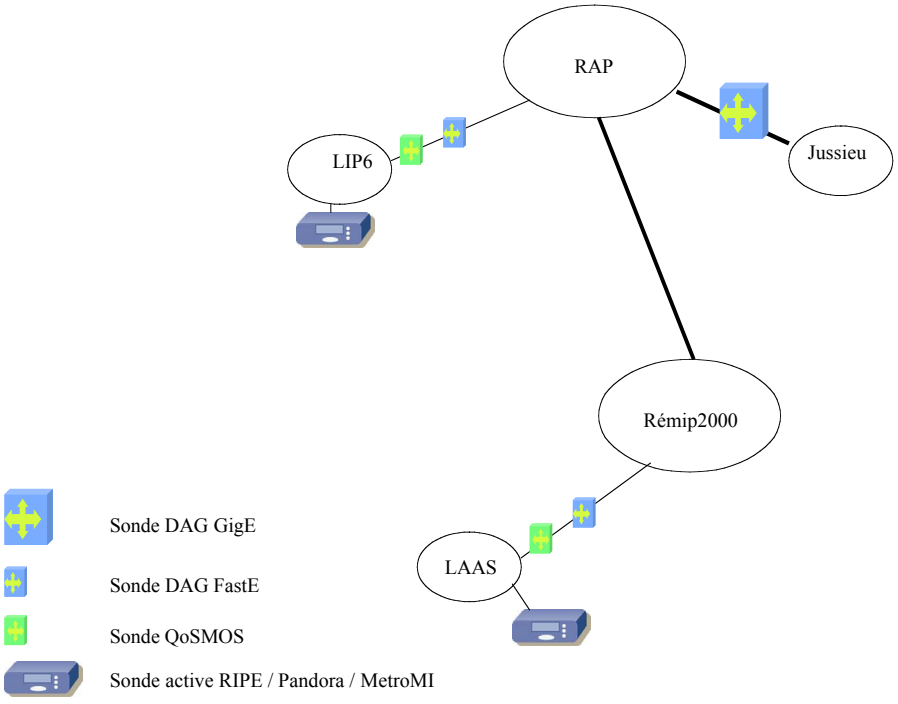


FIG. 33.7 – Carte de déploiement de quelques sondes DAG, QoSMOS et MetroMI

précises et la base de temps GPS (horloges atomiques de référence universelles) est parfaite pour nous permettre de corréler les traces capturées en différents lieux.

Enfin, la base de données de traces est en cours de constitution, et les logiciels d'analyse de ces traces commencent à être bien avancés. Les premiers résultats d'analyse de ces traces sont d'ailleurs très intéressants et vont permettre à METROPOLIS de s'imposer comme une des principales initiatives de métrologie dans le monde. A noter que nous travaillons avec nos partenaires administrateurs de réseaux à la mise en place de serveurs pour la diffusion de ces traces, afin d'augmenter encore la visibilité de METROPOLIS auprès de la communauté « réseaux » internationale.

Bibliographie

[1] *IPPM IP Performance Metrics*.

[2] *Pandora*. http://www-sor.inria.fr/~patarin/pub/patarin_these.pdf.

[3] *Sapphire*. http://www.ripe.net/ttm/worm/.

[4] G. Almes, S. Kalidindi, and M. Zekauskas. A one way delay metric for ippm. *Internet RFC 2679*, September 1999.

[5] G. Almes, S. Kalidindi, and M. Zekauskas. A packet loss metric for ippm. *Internet RFC 2680*, September 1999.

[6] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson. Design principles for accurate passive measurement. *Passive and Active Measurement Workshop*, 2000.

[7] C. Demichelis and P. Chimento. Ip packet delay variation metric for ip performance metrics (ippm). *Internet RFC 3393*, November 2002.

[8] J. Mahdavi and V. Paxson. Ippm metrics for measuring connectivity. *Internet RFC 2678*, September 1999.

[9] P. Owezarski, D. Andreu, C. Fricker, K. Salamatian, C. Chekroun, N. Benameur, P. Olivier, J. Roberts, and F. Guillemin. Projet rnrt metropolis. sous-projet 1 : Rapport d'état de l'art, janvier 2003.

[10] Vern Paxson, Jamshid Mahdavi, Andrew Adams, and Matt Mathis. An architecture for large-scale Internet measurement. *commag*, 36(8) :48–54, August 1998. http://www.psc.edu/networking/papers/nimi.html.

[11] J. Xu, J. Fan, M. Ammar, and S. Moon. On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization. In *ACM SIGCOMM Internet Measurement Workshop (IMW)*, San Francisco, 2001.

[12] J. Xu, J. Fan, M. Ammar, and S. Moon. Prefix-Preserving IP Address Anonymization : Measurement-Based Security Evaluation and a New Cryptography-based Scheme. In *ICNP*, Paris, 2002.