SIARSv2

PF - Packet Filter

Peter N.M. Hansteen & Matthieu Herrb

18 octobre 2018

Agenda

Agenda

Qu'est-ce que PF ?

- Filtre de Paquets réseau niveau 3
- Intervient sur le trafic entrant ou sortant d'une interface
- Permet de :
  - bloquer/autoriser le paquet à continuer son chemin
  - placer un label sur le paquet
  - modifier le paquet (NAT, ...)
  - re-diriger le paquet vers une autre destination

Filtrage à état (stateful)

- Les protocoles réseau sont gérés par des automates (exemple: TCP)
- PF suit l'état de l'automate pour les protocoles connus
- Cela permet de :
  - détecter les paquets qui violent le protocole
  - limiter la complexité des règles de filtrage en laissant passer automatiquement les paquets valides du protocole.
  - contrôler les ressources utilisées par le filtre

Règles de PF

- Stockées dans `/etc/pf.conf`
- S'appliquent au niveau 3 (IP, ICMP, TCP, UDP, ...)
- Lues dans l'ordre où elles apparaissent
- La **dernière** règle qui matche un paquet s'applique.
  (Attention : différent de Cisco / Linux ...)
- Si une règle modifie un paquet, le paquet modifié est utilisé en entrée des règles suivantes.
- Les tables permettent de traiter des listes dynamiques d'adresses IP

PF - traduction d'adresses

- réécriture des adresses source : NAT/PAT vers
  - une adresse unique
  - un pool d'adresses
- réécriture des adresses destination : redirection vers
  - un relais applicatif (hors du noyau)
  - équilibrage de charge
- traductions en cours stockées dans la table d'états des connexions.

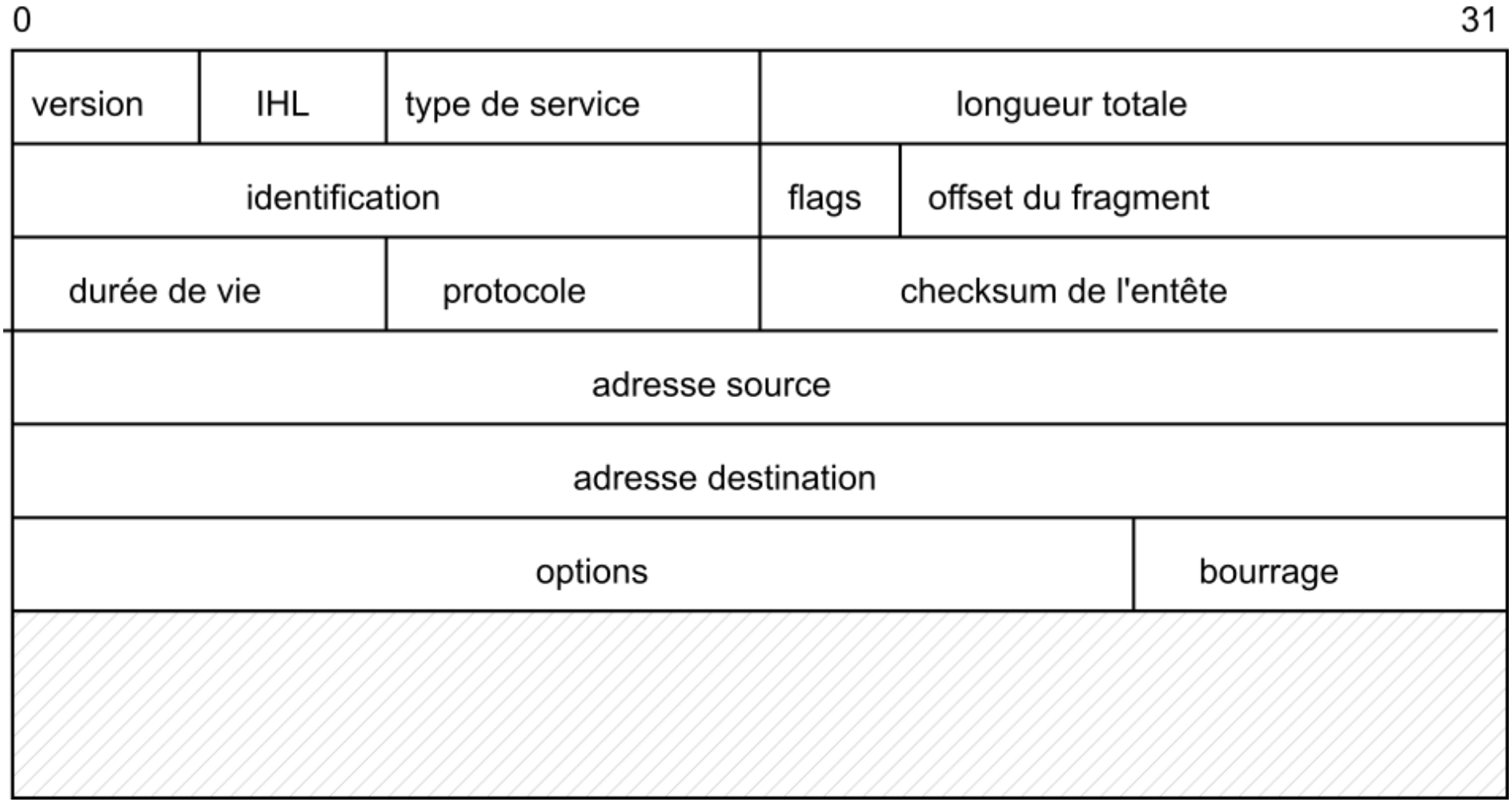
PF - normalisation

- **but** : éliminer les ambiguïtés / erreurs d'interprétation des états et protection des piles IP vulnérables.
- ré-assemble les fragments des paquets IP avant filtrage
- modification (réécriture) des numéros de séquence TCP

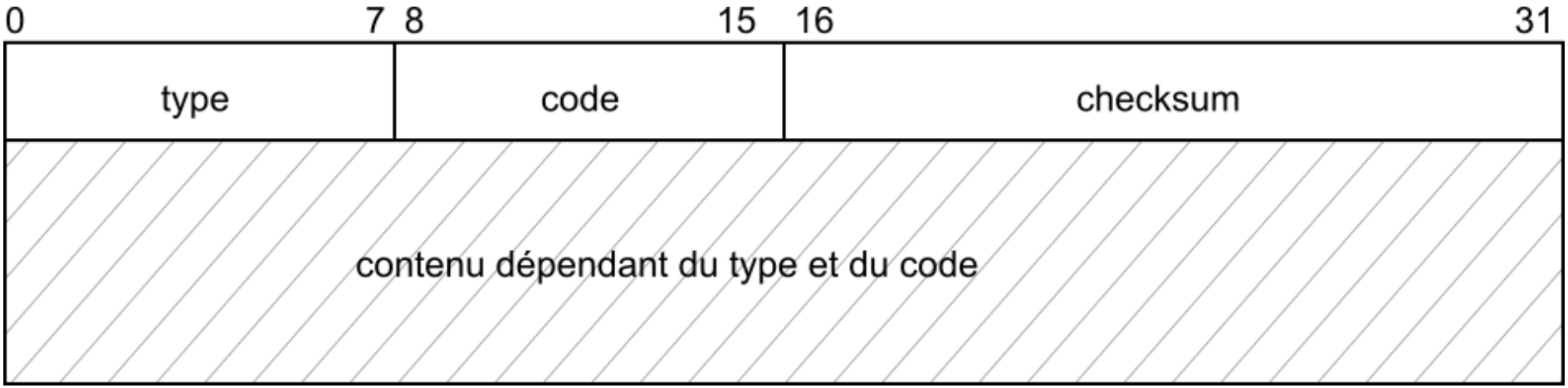
Journalisation

- Utilise le format **pcap** (tcpdump)
- Permet analyse ultérieure avec nombreux outils
- Outils pour redirection vers syslog disponible

Rappel : entêtes IP

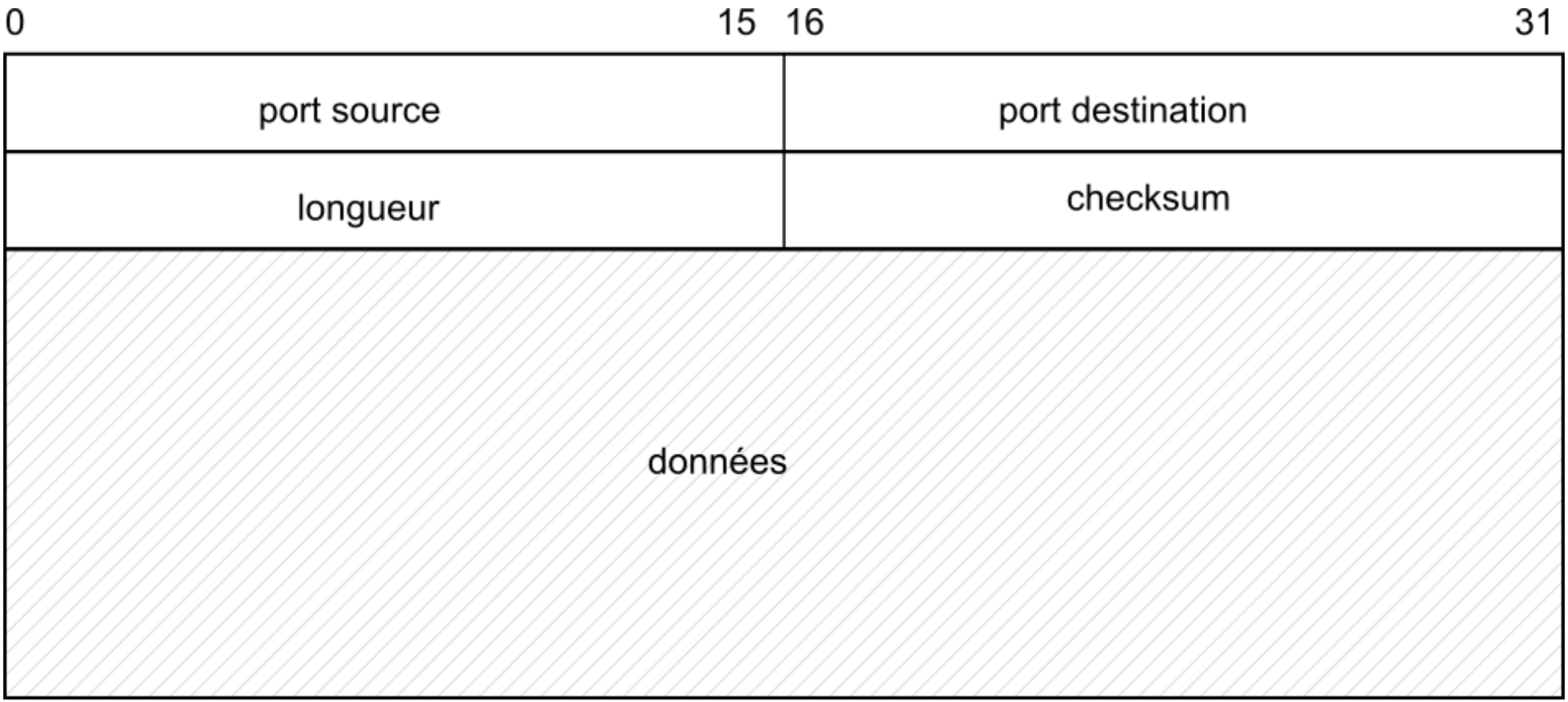


Rappel : paquets ICMP

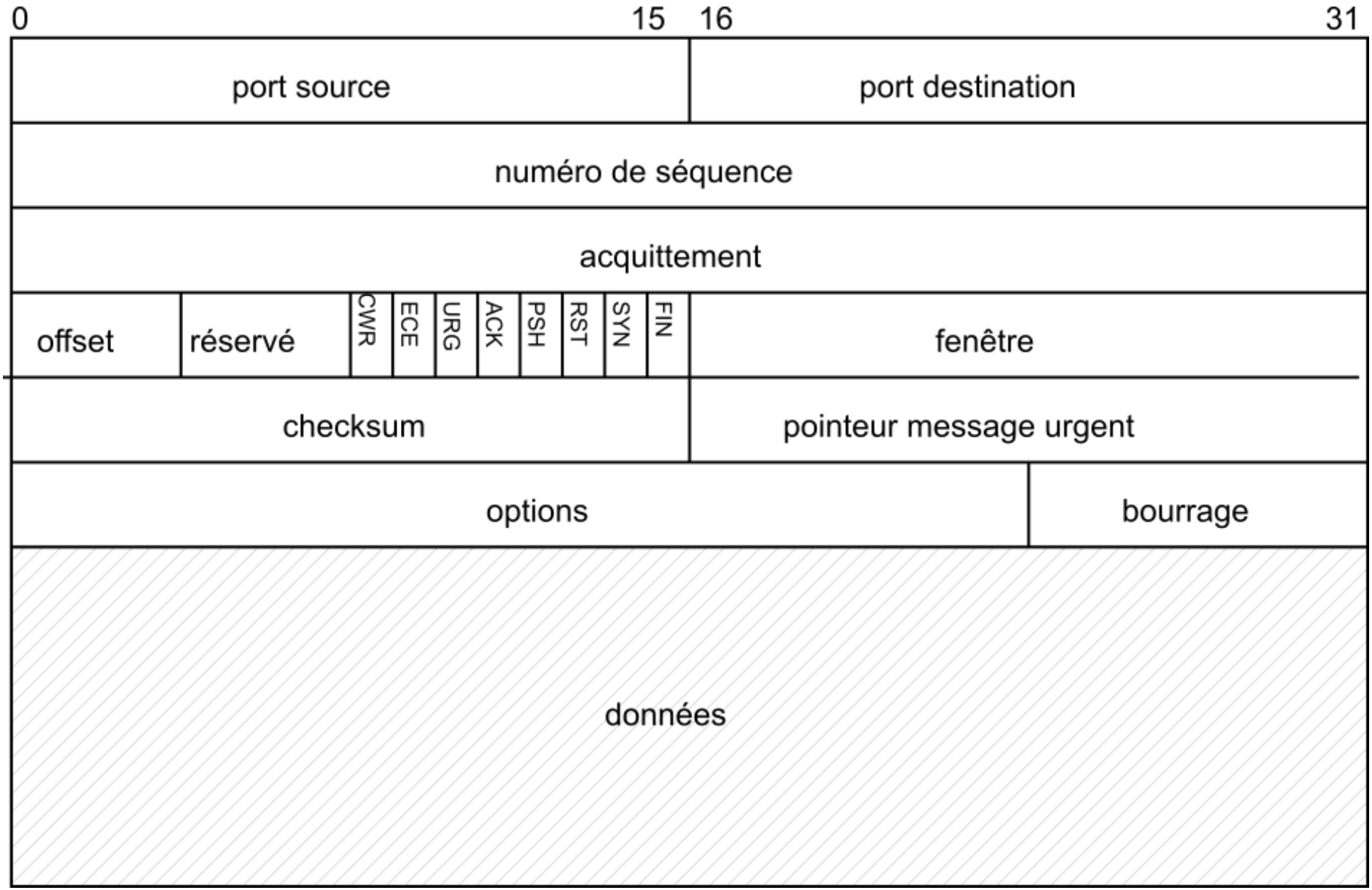


type	code	sémantique
0		**echo reply**
3		**destination unreachable**
	0	Net unreachable
	1	Host unreachable
	4	Fragmentation needed
	13	Communication administratively prohibited
8		**echo request**
11		**time exceeded**
	0	time to live exceeded in transit
	1	fragment reassembly time exceeded

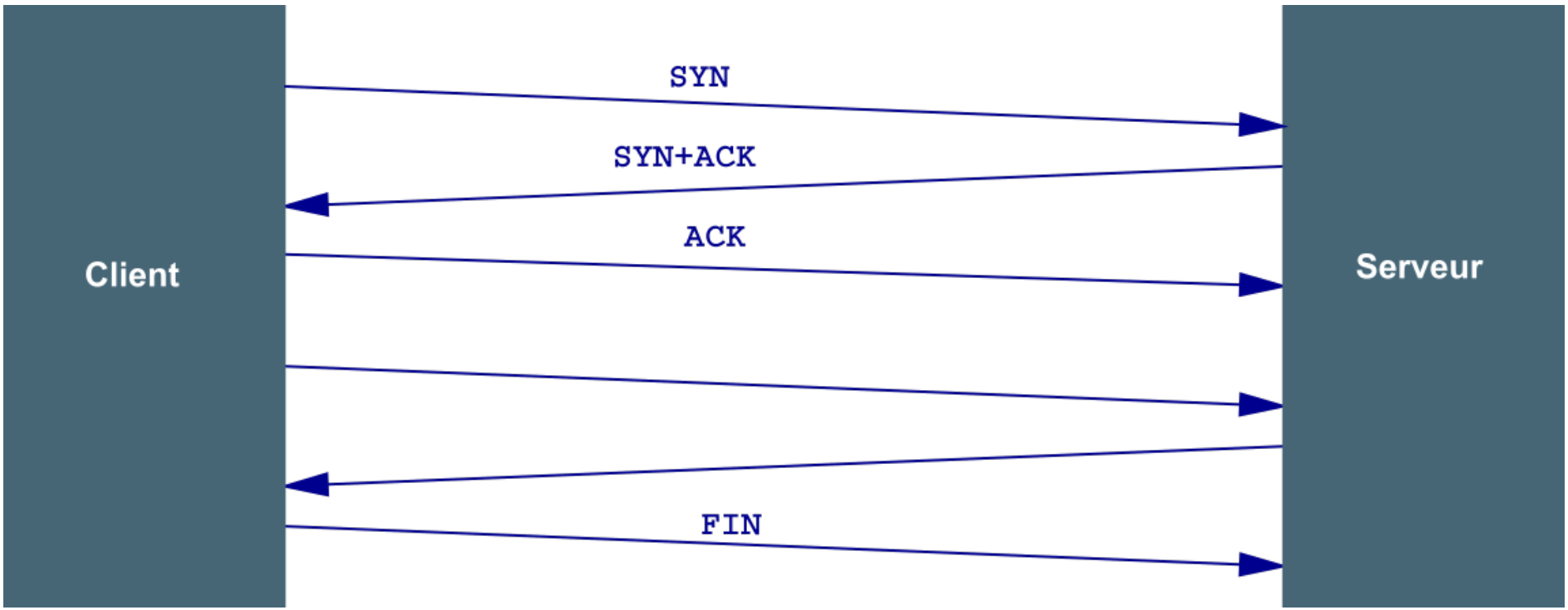
Rappel : paquets UDP



Rappel : paquets TCP



Rappel : 3-Way Handshake TCP



Agenda

Configuration minimale

`/etc/pf.conf` :

```
pass
```

Activer pf : `pfctl -e -f /etc/pf.conf`

Voir les états créés : `pfctl -vs rules`

Protection de la machine locale

```
set skip on lo0
block in
pass out
pass in on egress proto tcp port ssh
```

`egress` : groupe d'interfaces vers l'extérieur (défini automatiquement lors de la configuration réseau par `/etc/netstart`).

Un pare-feu simple

```
intif = ”em0”
dmz = ”em1”
extif = ”em2”

set skip on lo, $intif, $dmz
block in on $extif
pass out on $extif
pass in on $extif proto icmp
pass in on $extif to ($dmz:network) \
       port {ssh, smtp, domain, www, https, smtps, imaps}
```

PF et pont : pare-feu transparent

- Configuration d'un pont entre 2 interfaces
- Si pf est actif traite le trafic sur les 2 interfaces
- Pas de routage : permet de se placer dans une architecture existante
- Pas d'adresse IP sur les interfaces : pas détectable
- Quelques limitations : difficile d'implémenter un proxy sans adresse IP,...

Agenda

Contenu de pf.conf

Plusieurs types de définitions :

macros remplacements textuels

tables structures efficaces pour des listes d'adresses

options configuration du comportement de PF

files d'attente contrôle du débit et des priorités entre flux

règles de filtrage bloque ou laisse passer les paquets, avec réécriture si nécessaire.

points d'ancrage pour l'ajout dynamique de règles

Règles syntaxiques :

- Les lignes blanches sont ignorées
- Les lignes commençant par # sont des commentaires
- Utiliser \ pour continuer sur plusieurs lignes

Listes

Une liste permet de spécifier plusieurs règles similaires.
Définie par des éléments entre `{ }`

```
block in on vr0 from { 192.168.0.1, 10.5.32.6 } to any
```

est expansée en :

```
block in on vr0 from 192.168.0.1 to any
block in on vr0 from 10.5.32.6 to any
```

Macros

Variables définies par l'utilisateur.
Permettent de réduire la complexité et d'augmenter la lisibilité.
Pas expansées entre "..."

```
extif = "fxp0"

block in on $extif from any to any
```

Peuvent contenir des listes :

```
amis = "{ 192.168.1.1, 10.0.2.5, 192.168.43.53 }"

host1 = "192.168.1.1"
host2 = "192.168.1.2"
my_hosts = "{" $host1 $host2 "}"
```

Tables

- stocker de façon efficace et compacte des listes d'adresses IP
- initialisées dans `pf.conf` ou un fichier dédié
- modifiables dynamiquement avec `pfctl`
- utilisable presque partout où pf attend une adresse.
- mot clé `table`

Exemple :

```
table <rfc1918> const { 192.168.0.0/16, \
                        172.16.0.0/12, 10.0.0/8}
table <spammers> persist file ”/etc/pf/spammers”

block from <spammers> to any
block from <any> to <rfc1928>
```

Tables - manipulations

`pfctl -t table -T operation`

table nom de la table à manipuler

opération commande à appliquer

`add adresse`	ajoute une adresse
`delete adresse`	supprime une adresse
`load`	recharge la définition de la table
`zero`	vide la table
`expire secondes`	supprime les entrées sans trafic depuis *secondes*
`show`	liste le contenu de la table

Négations - attention danger !

`!adresse` matche toutes les adresses sauf celle-ci.

Comportement non intuitif dans une liste !

```
block from { 192.168.2.0/24, !192.168.2.1 }
```

devient :

```
block from 192.168.2.0/24
block from !192.168.2.1
```

→ bloque tout !

Mais avec une table :

```
table <badguys> { 192.168.2.0/24, !192.168.2.1 }
block from <badguys>
```

→ bloque 192.168.2.0/24 sauf 192.168.2.1

Règles de filtrage

[Attention nouvelle syntaxe pour OpenBSD!]

Syntaxe **simplifiée** :

```
block|match|pass [in|out] [log] [quick] \
  [on interface ] \
  [inet|inet6] [protocol proto ] \
  [from adr_src ] [port port_src ] \
  [to adr_dest ] [port port_dst ] \
  [flags tcp_flags ] [paramètres ]
```

Action

block bloque le paquet.
Paramètre global `block-policy` :

drop	paquet ignoré en silence
return	répond RST pour TCP et ICMP Unreachable pour les autres
return-icmp	répond tjs ICMP Unreachable

match la règle est évaluée sans changer la décision, par ex. pour `scrub` ou NAT.

pass le paquet passe avec création d'état sauf si `no state` est spécifié

Options

In/Out permet de spécifier le sens (entrant ou sortant). Par défaut la règle s'applique dans les 2 sens.

log permet de tracer le paquet via l'interface `pflog0`.

quick permet d'arrêter l'exploration des règles pour ce paquet si il matche la règle.
(Rappel : c'est la *dernière* règle matchant un paquet qui s'applique).

on interface spécifie l'interface concernée.
Par défaut la règle s'applique à toutes les interfaces actives

Famille d'adresse et protocole

- La famille d'adresse peut limiter aux paquets IPv4 (**inet**) ou IPv6 (**inet6**) seuls. Par défaut les 2 versions du protocole sont considérées.
- Le protocole limite la règle au protocole (défini dans le RFC 1340 et dans `/etc/protocols`)

La suite de la règle dépend du protocole.
`pfctl` génère une erreur si incohérence entre protocole et le reste.

Adresses source et destination

- une adresse IPv4 ou IPv6 seule
- un bloc d'adresses en notation CIDR
- un nom résolu dans DNS, remplacé par toutes les adresses contenues dans la réponse
- le nom d'une interface remplacé par ses adresses.
  Modificateurs :

`:network`	adresse du réseau
`:peer`	le correspondant d'une liaison point-à-point
`:0`	l'adresse principale seulement (sans aliases)

- une table
- une liste d'adresses ci-dessus
- une adresse précédée de **!** (négation)
- le mot clé **any**

Numéros de ports (TCP & UDP)

- un nombre entre 1 et 65535
- un nom de `/etc/services`
- un ensemble de ports défini par une liste
- une expression définissant un ensemble:

`!=`	différent
`<, >, <=, >=`	inférieur, supérieur (resp ou égal)
`><`	entre les deux (exclusif)
`:`	entre les deux (inclusif)
`<>`	inférieur au premier ou supérieur au second

Exemples :

```
port {80, 443}
port >= 1024
port 6881:6889
```

Drapeaux TCP

Sous la forme *test*/*masque* ou bien `any`
Matche si les drapeaux de *test* parmi *masque* sont positionnés

Abréviations : `(F)IN, (S)YN, (R)ST, (P)USH, (A)CK, (U)RG, (E)CE, C(W)R`

Exemples :

`flags S/S`	le flag SYN est actif. Les autres sont ignorés
`flags S/SA`	le flag SYN est actif, mais pas ACK. Les autres sont ignorés
`flags /SFRA`	aucun des flags SYN, FIN, RST ou ACK n'est positionné.

Création d'état

Par défaut un état est créé pour tous les paquets TCP qui passent avec les flags S/SA.
`no state` permet de ne pas créer d'état.
`flags any` permet de créer un état avec n'importe quel paquet, pour « attraper » une connexion existante.

(Rappel : la table des états est consultée avant les règles et si un paquet match un état existant il passe directement)

Translation d'adresses et redirection

nat-to adresse

réécrit l'adresse source du paquet,
et éventuellement le numéro de port source.

Exemple :

```
match out on $ext_if from 192.168.1.0/24 nat-to ($ext_if)
```

rdr-to adresse

réécrit l'adresse destination et le port destination

Exemple :

```
pass in quick on $ext_if proto tcp to port smtp \
                        rdr-to 127.0.0.1 port 8025
```

Routeur NAT

```
extif = ”em0”
intif = ”em1”
bastion = ”192.168.1.1”
set skip on {lo0, $intif}

# NAT en sortie
match on $extif inet from $intif:network nat-to ($extif)

block in log all
pass out on $extif
# ping
pass in on $extif proto icmp all icmp-type echoreq
# Traceroute
pass in on $extif proto udp all port 33433 >< 33626

# SSH sur port 2222 vers bastion
pass in log on $extif proto tcp to ($extif) port 2222 \
    rdr-to $bastion port 22
```

Agenda

Bloquer les brutes

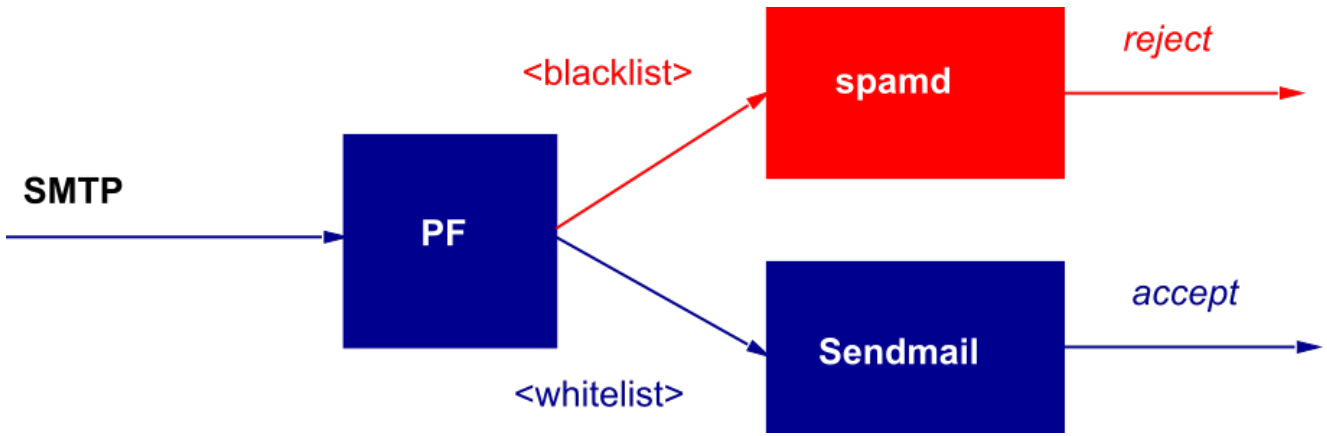
- force brute SSH : nombreuses connexions par seconde
- solution : limiter le taux de connexions

```
table <bruteforce> persist

# Bloque attaques brute force et autres emm...
block in quick log on em2 from <bruteforce>

pass in proto tcp from any to <administrees> port ssh \
       flags S/SA keep state \
       (max-src-conn 20, max-src-conn-rate 15/30, \
       overload <bruteforce> flush global)
```

Anti-spam : spamd

protection anti-spam au niveau du pare-feu

- black list
- white list
- greylisting
- greytrapping



Voir tous les cas d'utilisation dans la page `spamd(8)`

Agenda

Logs de pf

- pf logue des paquets bruts marqués `log` dans les règles
- pf crée une interface dédiée : `pflog0`
- pf utilise le format **pcap** (tcpdump)
- le démon `pflogd(8)` permet de stocker ces paquets dans `/var/log/pflog`
- `tcpdump(8)` peut être utilisé pour visualiser les paquets. Options supplémentaires pour examiner l'état de pf associé.

Logs : exemple

tcpdump -e -n -i pflog0

```
19:09:09.719965 rule 17/(match) block out on em2: 140.93.1.176 > 192.168.3.1: icmp: echo request
19:09:09.963940 rule 43/(match) block out on em2: 140.93.65.166.1343 > 218.61.22.27.25: \
        S 1926308279:1926308279(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
19:09:10.459138 rule 43/(match) block out on em2: 140.93.65.166.1343 > 218.61.22.27.25: \
        S 1926308279:1926308279(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
19:09:10.895738 rule 17/(match) block out on em2: 140.93.0.15.60086 > 172.31.32.250.25: \
        S 192965518:192965518(0) win 49640 <mss 1460,nop,wscale 0,nop,nop,sackOK> (DF)
19:09:10.958700 rule 0/(match) block out on em2: 140.93.254.76.1230 > 169.254.56.138.123: \
        v1 client strat 0 poll 0 prec 0
19:09:10.960826 rule 43/(match) block out on em2: 140.93.65.166.1343 > 218.61.22.27.25: \
        S 1926308279:1926308279(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
19:09:13.118242 rule 0/(match) block out on em2: 140.93.254.232.1230 > 169.254.56.138.123:\
         v1 client strat 0 poll 0 prec 0
19:09:15.297774 rule 17/(match) block out on em2: 195.83.132.135.123 > 10.1.1.199.123: v3 \
        sym_pas strat 3 poll 4 prec -6 [tos 0x10]
19:09:17.565871 rule 17/(match) block out on em2: 140.93.2.193.58642 > 172.31.32.250.25: \
        S 2821787999:2821787999(0) win 49640 <mss 1460,nop,wscale 0,nop,nop,sackOK> (DF)
19:09:19.884189 rule 0/(match) block out on em2: 140.93.254.107.1230 > 169.254.56.138.123: \
        v1 client strat 0 poll 0 prec 0
```

Informations sur l'état du système

`pfctl(8)` permet de voir :

- l'état général de pf : `pfctl -si`
- les règles actives : `pfctl -vvsr`
- la table des états : `pfctl -vvss`
- les tables : `pfctl -vvsT`
- ...

`systat(8)` permet de voir en temps réel :

- l'état général de pf : `systat pf`
- les règles actives : `systat rules`
- le début de la table des états : `systat states`

Statistiques

Nombreuses briques de base :

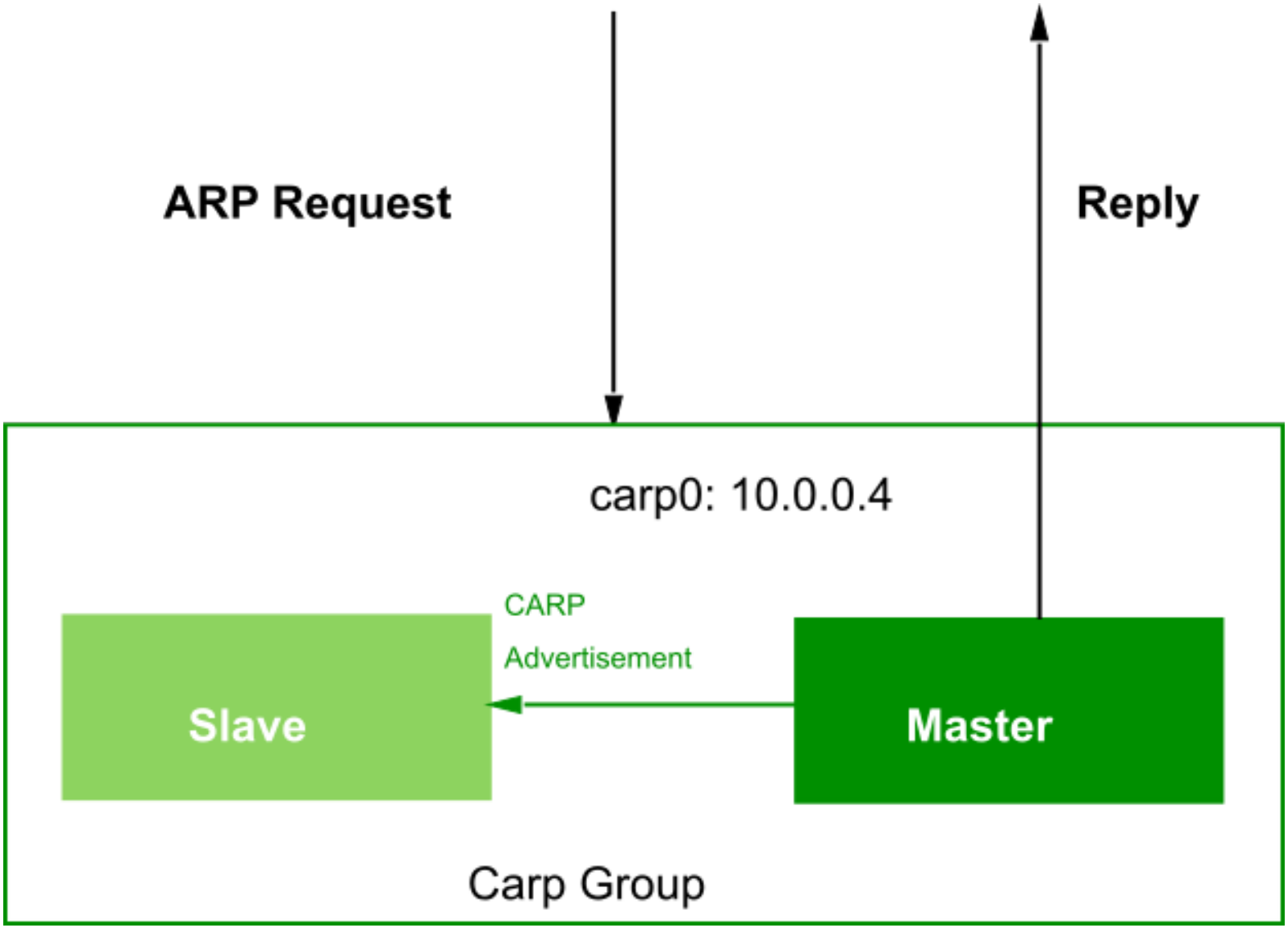
- `snmpd(8)` permet d'exporter les compteurs par interface.
  (Y compris `pflog0`).
- `pfstat(1)` dans les ports permet de faire des graphiques avec `rrdtool`
- `pflow(4)` permet d'exporter des données *netflow* v5.

Agenda

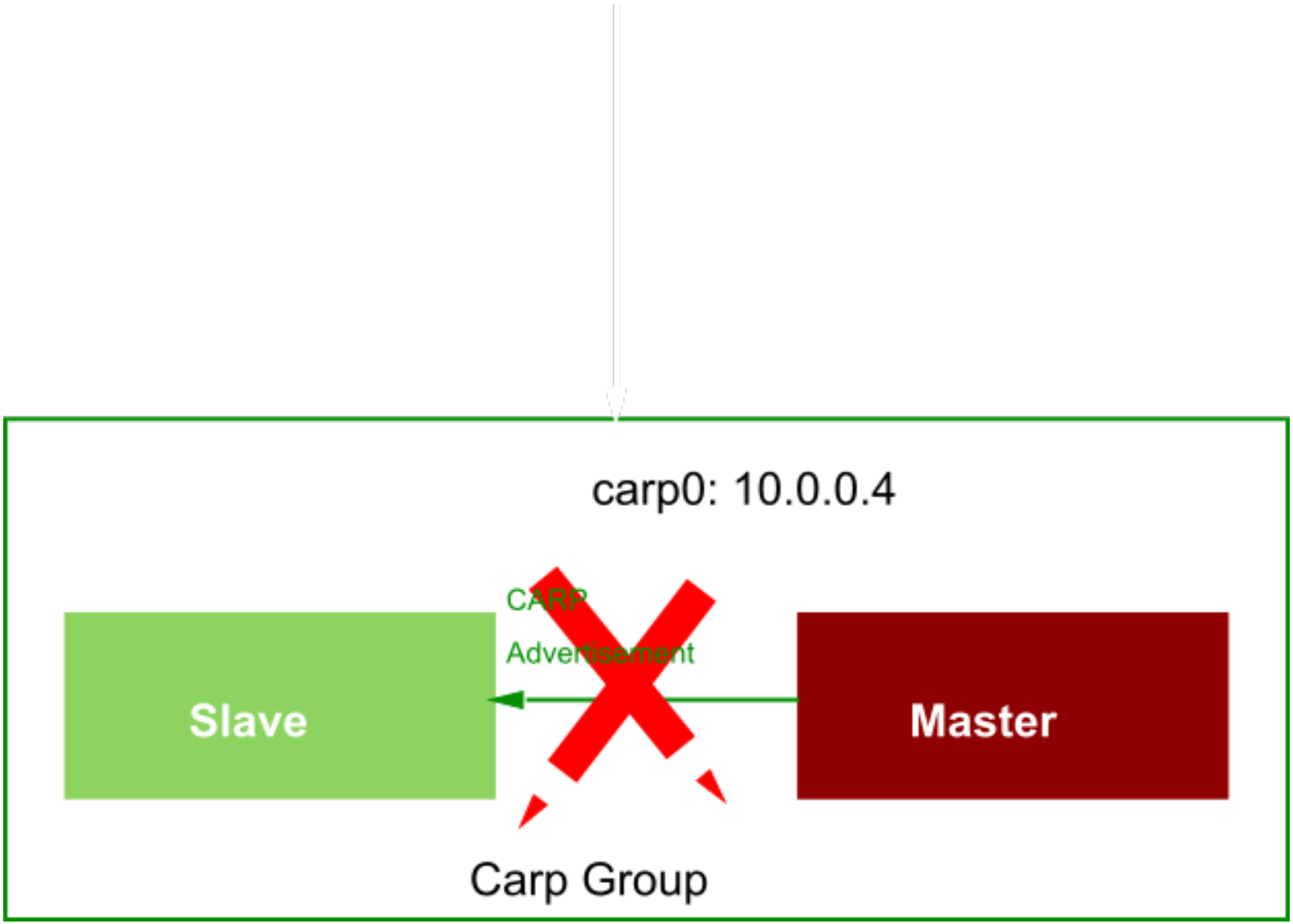
Principe

- CARP gère la redondance à la frontière des niveaux 2 et 3.
- **groupe CARP :** une adresse MAC et une adresse IP
- répond aux requêtes ARP pour l'adresse IP
- utilise l'adresse MAC du groupe
  $\rightarrow$ mise à jour des tables MAC des switches.
- Le maître envoie périodiquement des CARP advertisements (multicast)
  $\rightarrow$ indique qu'il est vivant
- Si plus d'advertisments, autres membres envoient des advertisements, puis élection d'un nouveau maître.

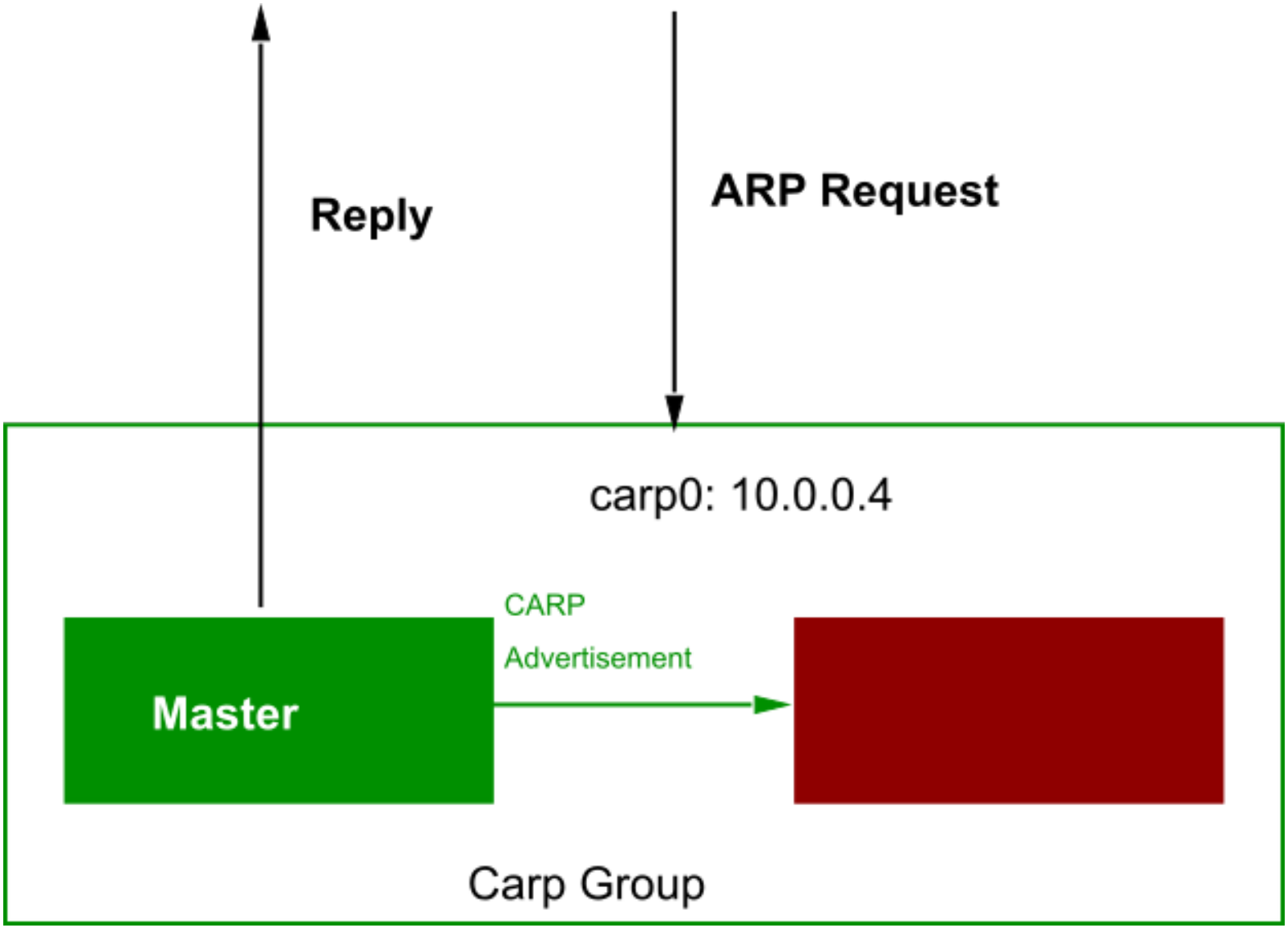
CARP: illustration



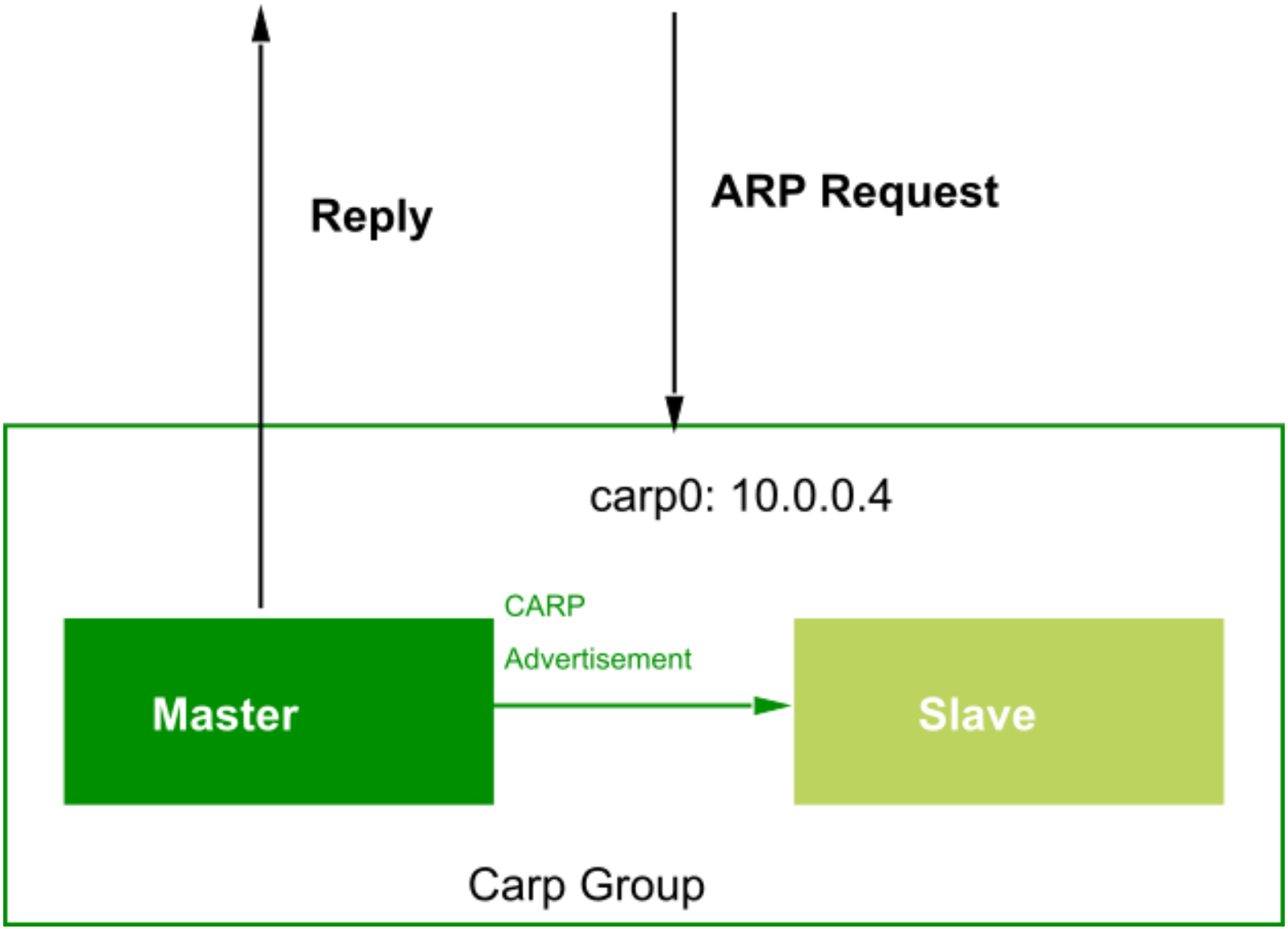
CARP: illustration



CARP: illustration



CARP: illustration



Préemption

Permet de définir des priorités entre les membres d'un groupe

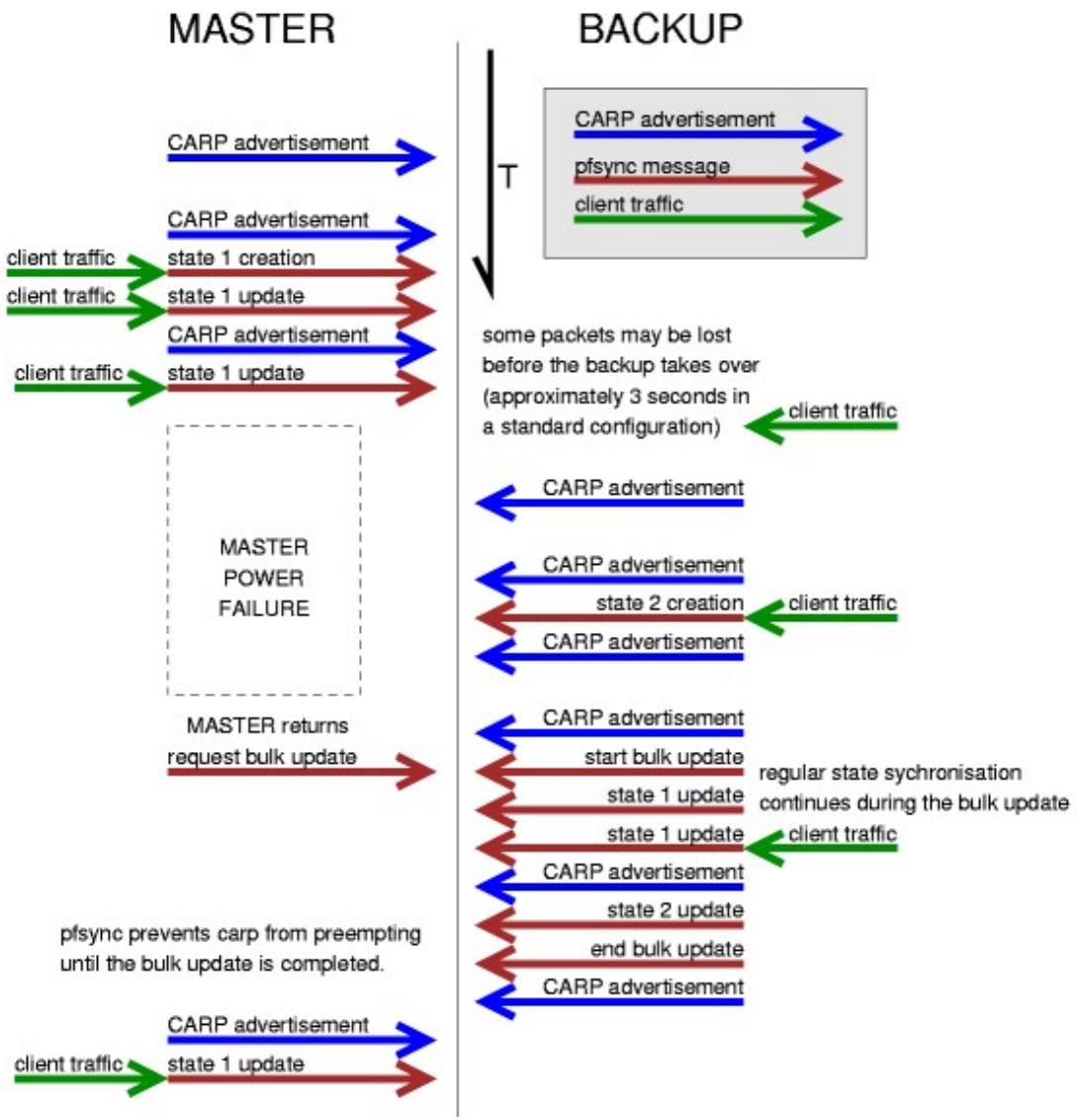
- celui qui a la plus haute priorité (re)devient le maître
- répartition de charge à partir des priorités
  - au niveau ARP
  - au niveau IP

Synchronisation de l'état entre les pare-feu.

pfsync

- protocole multicast 240
- diffuse les créations, mises à jour et destructions d'états
- écoute les mises à jour des voisins
- met à jour la table d'états locale

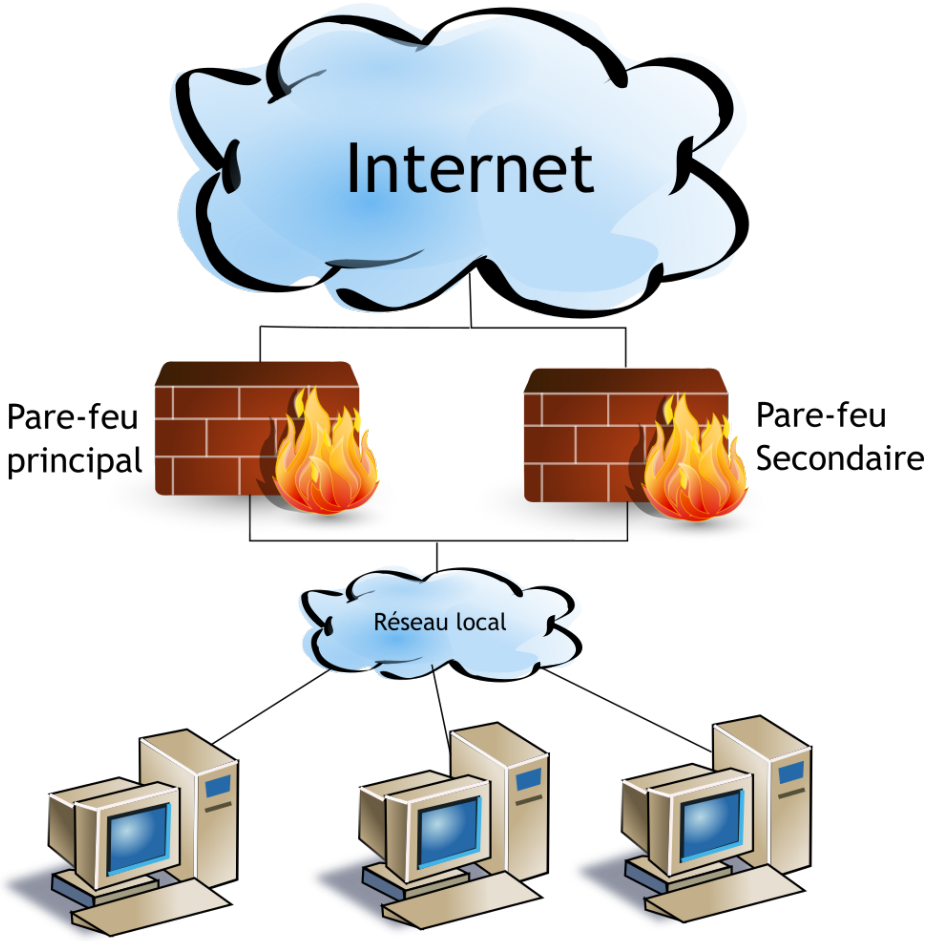
Chronogramme CARP + pfsync

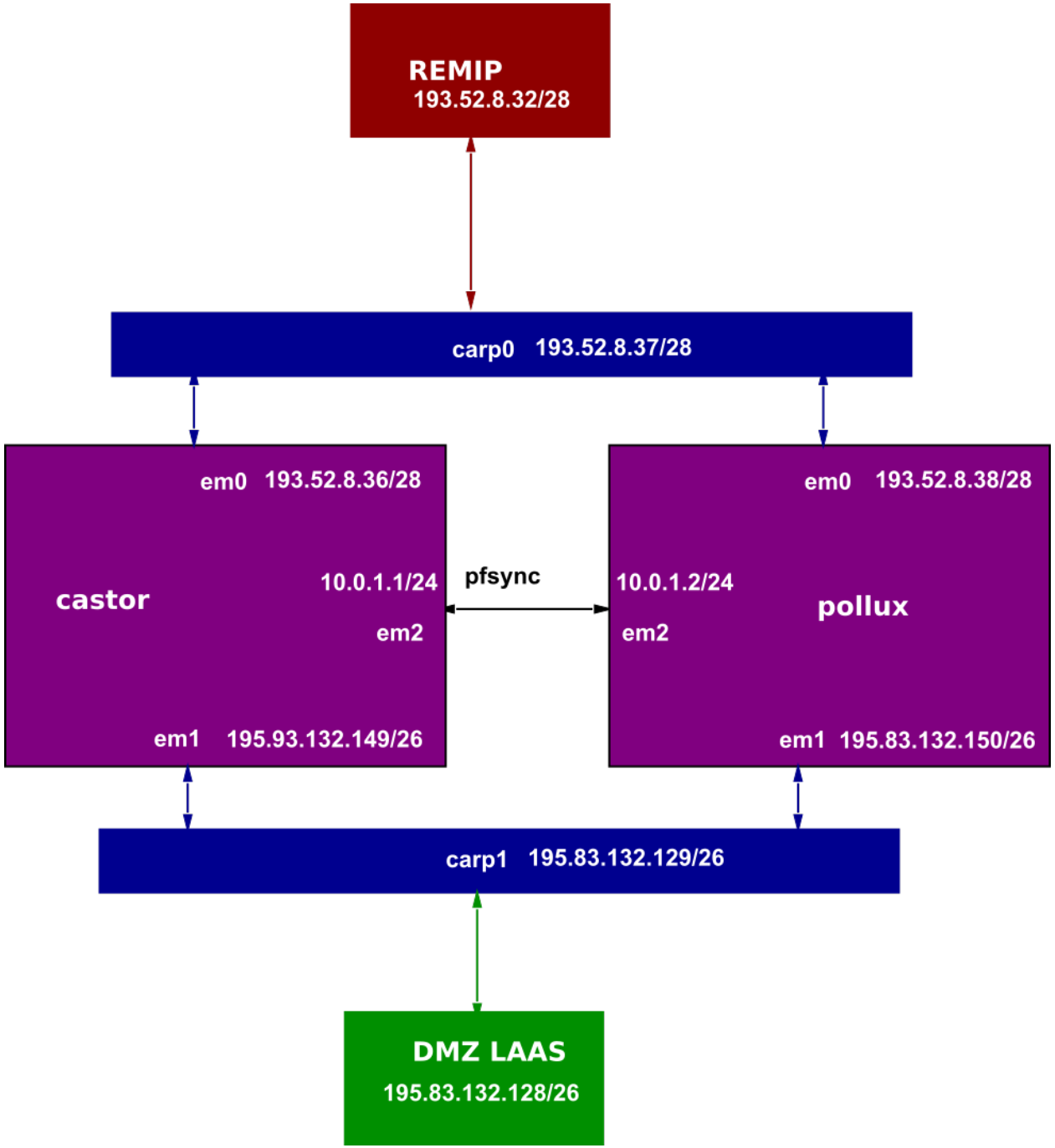


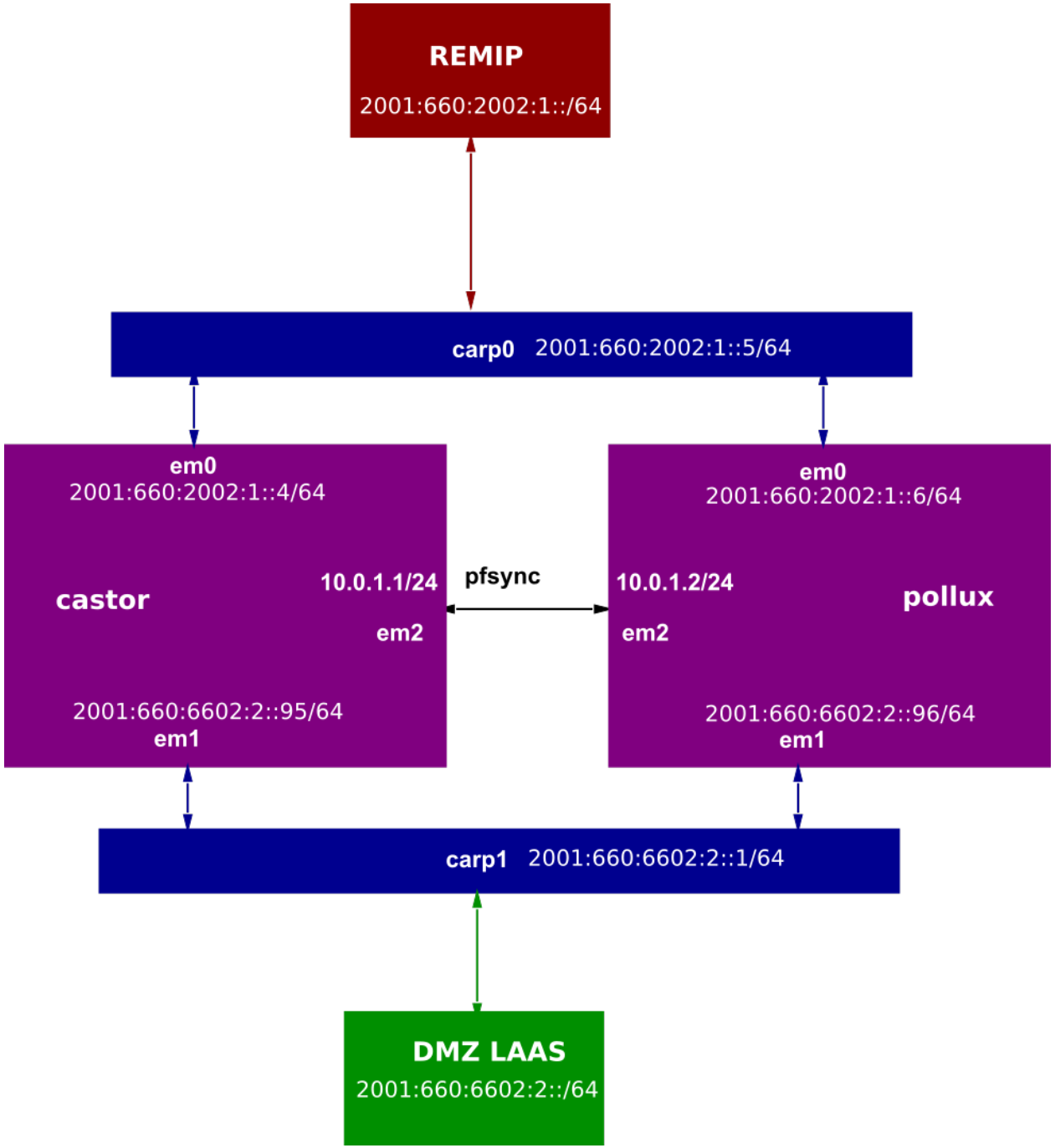
CARP : Spécificités par rapport à VRRP

- indépendant de la version du protocole.
  Supporte IPv4 et IPv6
- fonction arp balance pour faire du partage de charge.
- protection des advertisements par HMAC SHA-1

Configuration




REMIP
193.52.8.32/28
carp0 193.52.8.37/28
em0 193.52.8.36/28
castor
10.0.1.1/24
em2
em1 195.93.132.149/26
pfsync
em0 193.52.8.38/28
10.0.1.2/24
em2
pollux
em1 195.83.132.150/26
carp1 195.83.132.129/26
DMZ LAAS
195.83.132.128/26


REMIP
2001:660:2002:1::/64
carp0 2001:660:2002:1::5/64
em0
2001:660:2002:1::4/64
castor
10.0.1.1/24
em2
2001:660:6602:2::95/64
em1
pfsync
em0
2001:660:2002:1::6/64
10.0.1.2/24
em2
pollux
2001:660:6602:2::96/64
em1
carp1 2001:660:6602:2::1/64
DMZ LAAS
2001:660:6602:2::/64

configuration maître

/etc/hostname.carp0

```
vhid 100 carpdev em0 pass pass1
inet 193.52.8.37 255.255.255.240
inet6 2001:660:2002:1::5 64
```

/etc/hostname.carp1

```
vhid 101 carpdev em1 pass pass2
inet 195.83.132.129 255.255.255.192
inet6 2001:660:6602:2::1 64
```

/etc/hostname.pfsync0

```
up syncif em2 syncpeer 10.0.1.2
```

configuration esclave

/etc/hostname.carp0

```
vhid 100 advskew 200 carpdev em0 pass pass1
inet 193.52.8.37 255.255.255.240
inet6 2001:660:2002:1::5 64
```

/etc/hostname.carp1

```
vhid 101 advskew 100 carpdev em1 pass pass2
inet 195.83.132.129 255.255.255.192
inet6 2001:660:6602:2::1 64
```

/etc/hostname.pfsync0

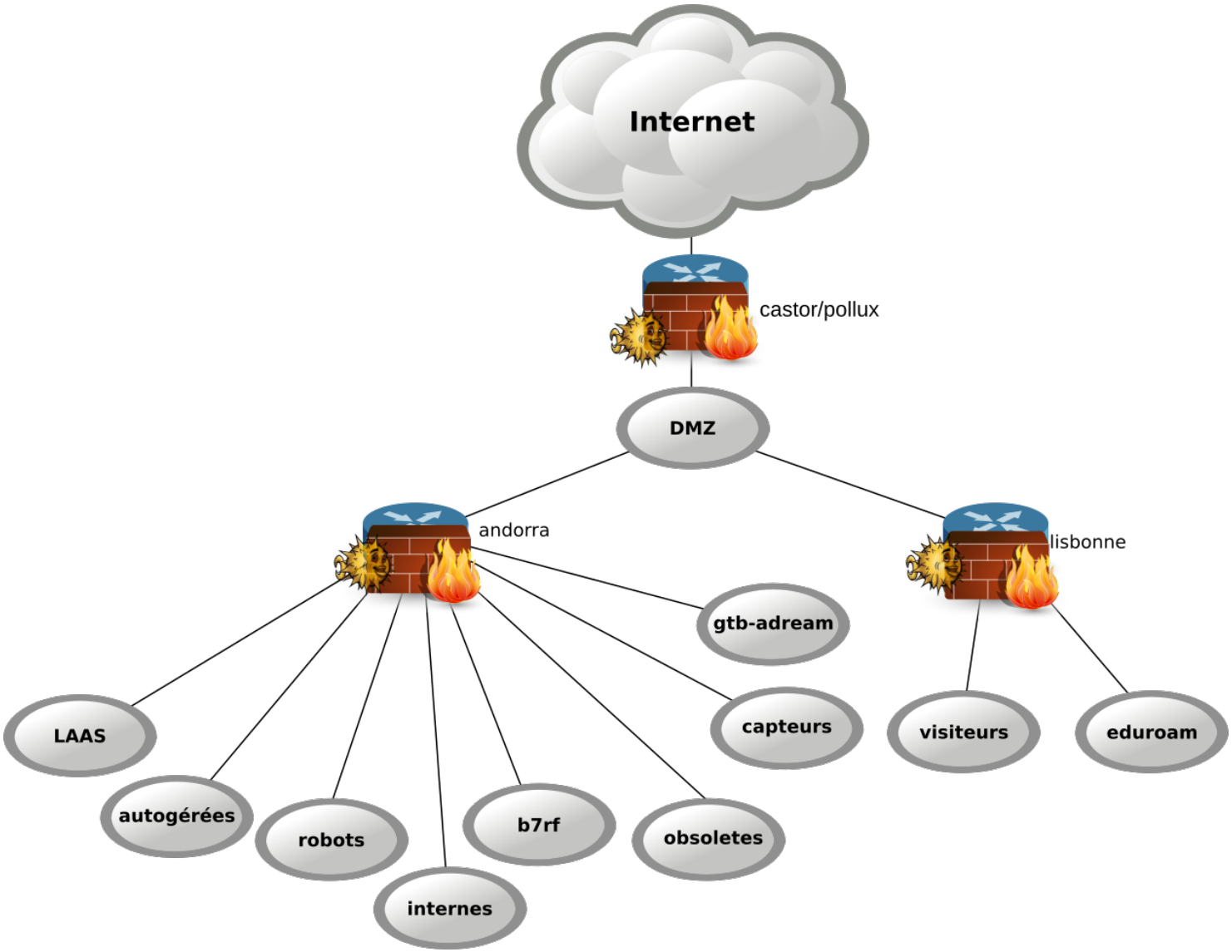
```
up syncif em2 syncpeer 10.0.1.1
```

Configuration pf

/etc/pf.conf

```
block in log
block out log

pass on $carpdevs proto carp
pass on $syncdev proto pfsync keep-state (no-sync)
```

Exemple: architecture LAAS



Questions ?