QoS Prototype

Riadh BEN HALIMA & Khalil DRIRA

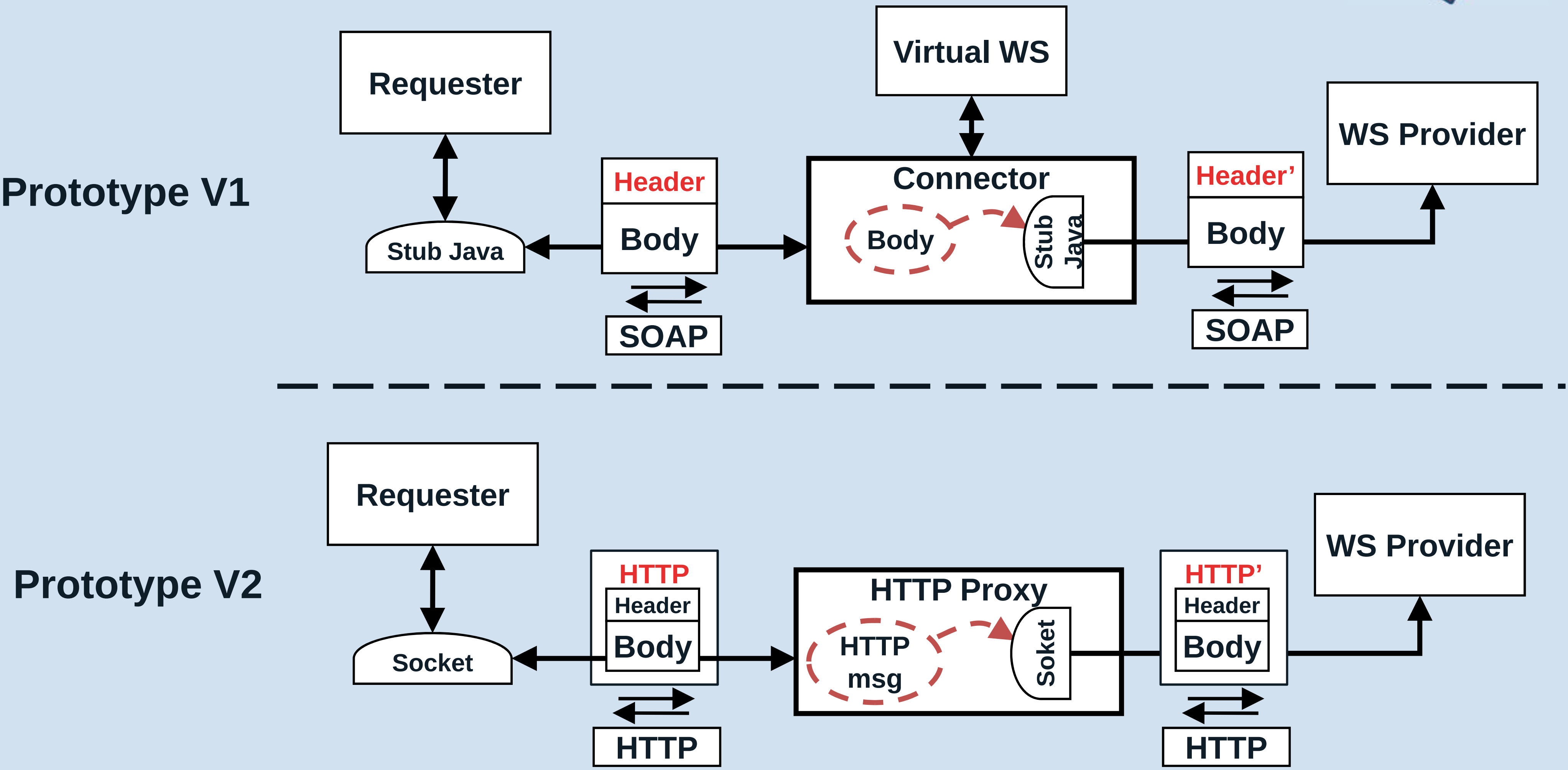
LAAS-CNRS

Toulouse meeting: 4-6 June 2008

Introduction

- QoS management in WS-DIAMOND
  - Objective: QoS-oriented self-healing for WS
  - Approach: Class-level Monitoring and repair based on statistical analysis of QoS values (response time mainly)
  - Implementations:
    - Prototype V1 (demo review1)
    - Prototype V2 (demo review2)
  - Experiments:
    - Integration with Polimi Foodshop implementation (V1,V2)
    - Integrated With UNITO Logger (V2)

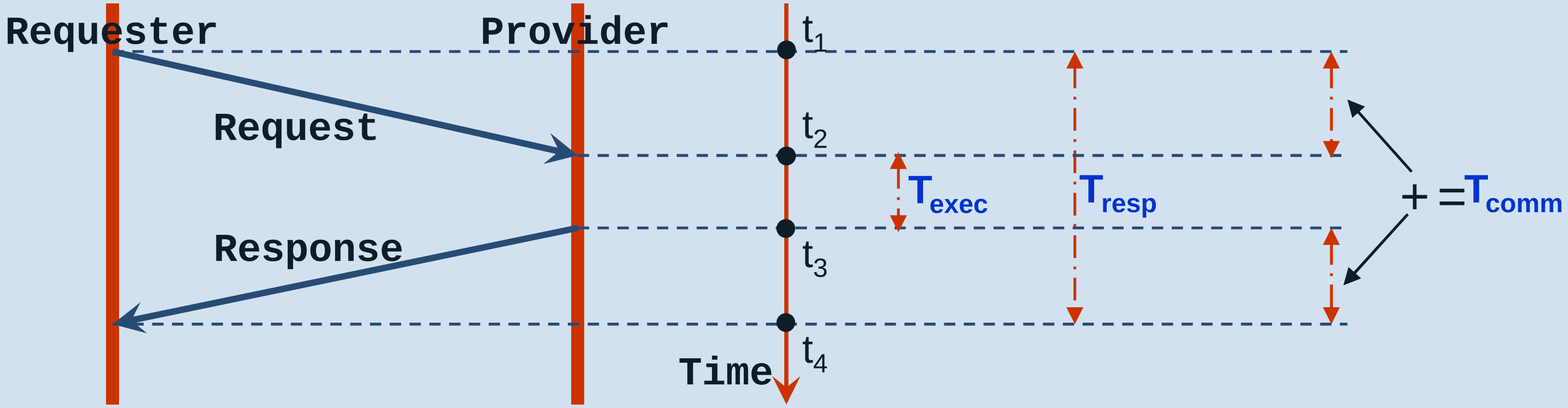
Prototype V1 vs Prototype V2



QoS Manager: Evolution in Prototype V2

- Main characteristics: Management at the HTTP level
  - Low level programming, Socket-based
    - HTTP proxies, handling of HTTP messages(including SOAP part)
  - HMM based degradation detection
    - Provides events for chronicles
  - Act at communication-level
  - Handle WS as a black box
  - Appropriate for asynchronous WS because SOAP Header information (MessageId, RelatesTo, Source) is not affected by using intermediates (HTTP Proxies) between requesters and providers
  - Appropriate for stateful WS because the intermediate reroutes HTTP by modifying IP address of the destination without affecting the SOAP Envelop (Header and Body)

Considered QoS parameters



- **Response Time** : The time between sending a request and receiving the response:
  - ***Tresponse = t4 – t1*** *(**RTT: Round Trip Time**)*

- **Execution Time**: The time that the provider needs to achieve the processing of the request:
  - ***Texecution = t3 – t2*** *(Has been considered for the prototype V1)*
- **Communication Time**: The time that the SOAP message needs to reach its destination:
  - ***Tcommunication = Tresponse – Texecution*** *(Has been considered for other scenarios)*

Implemented functions (1/2)

- Automatic and dynamic discovering of all involved parties for any applications (application profile):
  - IP address of the deployment computers
  - Names of the communicating WSs
  - Names of the operations, their kinds (synchronous/asynchronous) and their execution durations
- Automatic and dynamic building and graphical visualization of:
  - Which deployment computer hosts which WSs
  - Which operation being executed by which WSs
  - Sequences of invocation between operations

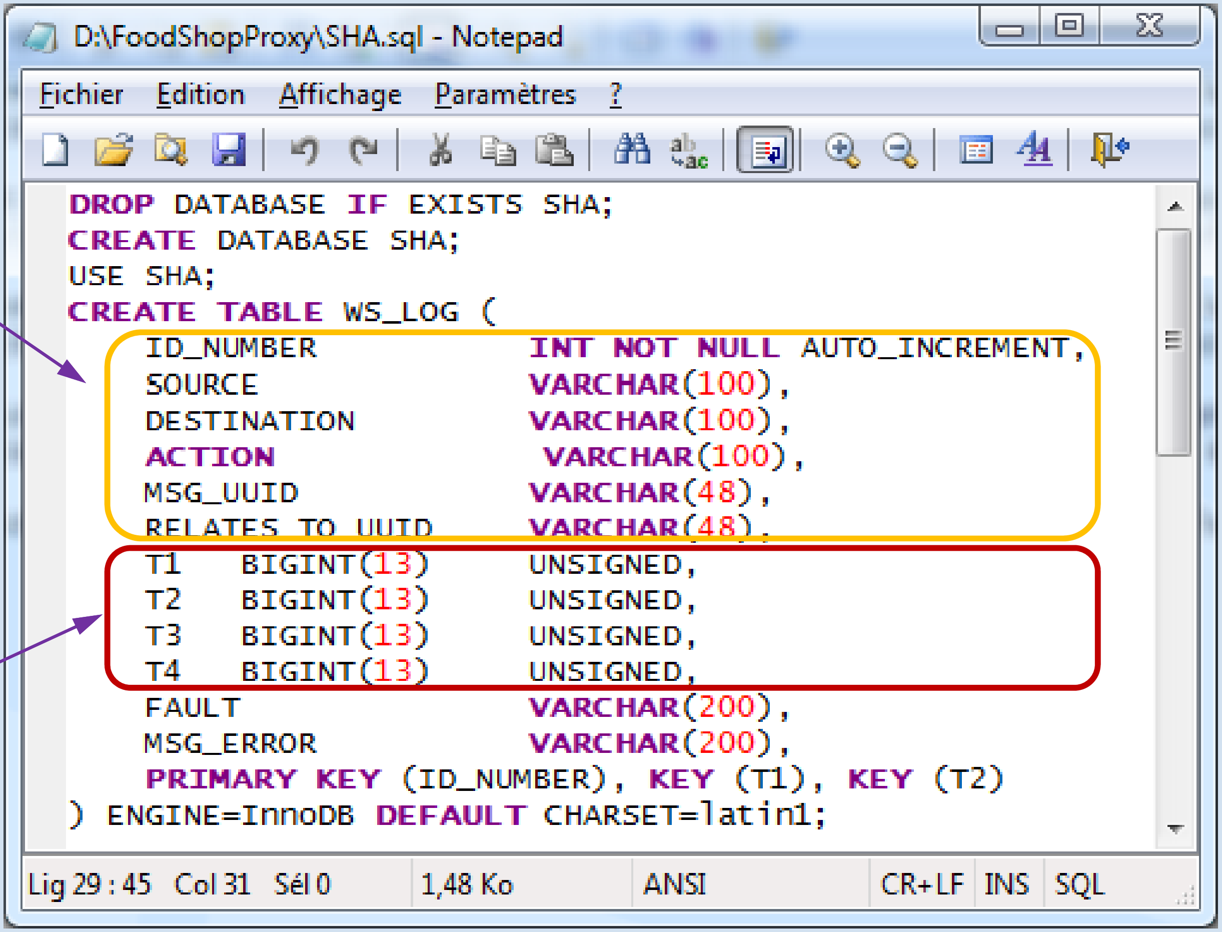
Implemented functions (2/2)

- Two application-independent parts:
  - HTTP Proxy (1144 Java code lines)
    - Monitoring, logging, and rerouting requests
    - 2 DB tables are maintained and used: WS_LOG, ROUTING
  - QoS Analysis & Graphical monitoring window (1326 Java code lines)
    - Extract logs, build and show application profile
    - Analyze and show WSs status (using QoS values)
    - 1 DB table is maintained and used: STATUS
- Two application-specific parts:
  - The WSs implementing the FoodShop (Polimi implementation)
  - A request generators (randomly and permanently generation of requests, instead of SoapUI)

Logs: logged information extracted from traffic monitoring

Application level information useful for building application profile

QoS related added information useful for the analysis

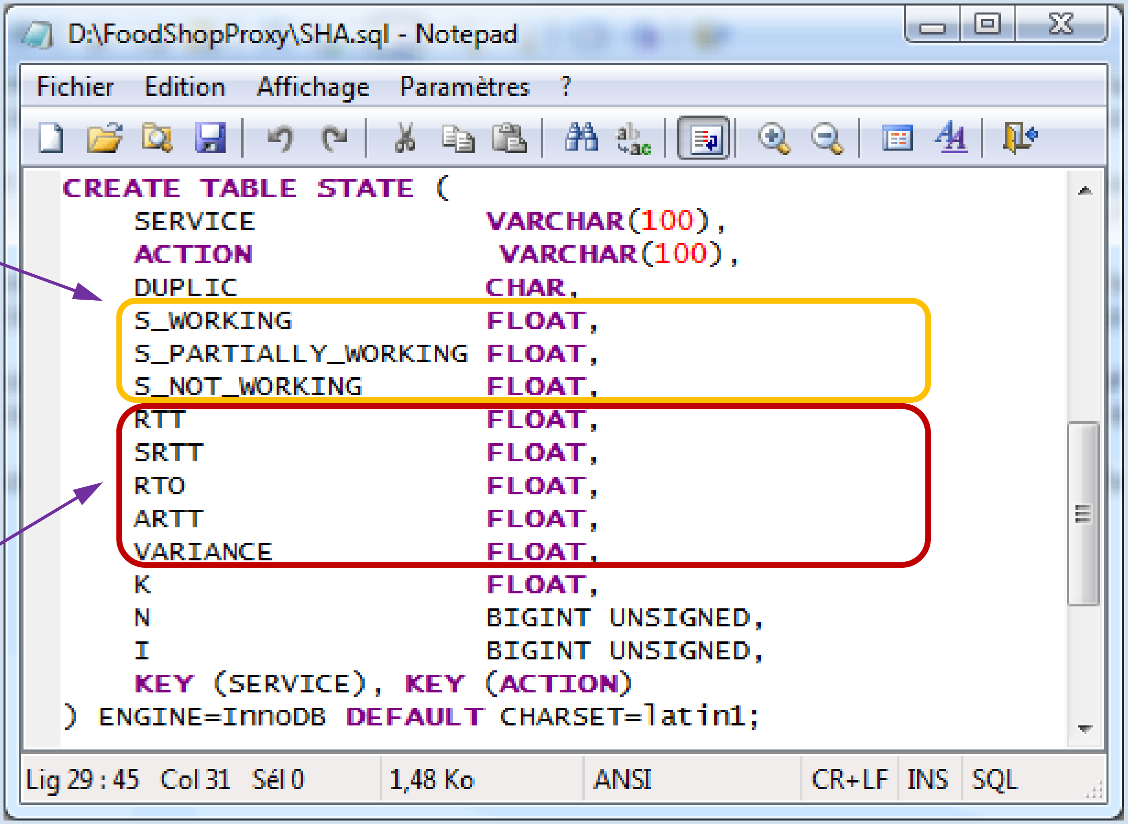


```
DROP DATABASE IF EXISTS SHA;
CREATE DATABASE SHA;
USE SHA;
CREATE TABLE WS_LOG (
   ID_NUMBER                INT NOT NULL AUTO_INCREMENT,
   SOURCE                   VARCHAR(100),
   DESTINATION              VARCHAR(100),
   ACTION                    VARCHAR(100),
   MSG_UUID                 VARCHAR(48),
   RELATES_TO_UUID          VARCHAR(48),
   T1    BIGINT(13)         UNSIGNED,
   T2    BIGINT(13)         UNSIGNED,
   T3    BIGINT(13)         UNSIGNED,
   T4    BIGINT(13)         UNSIGNED,
   FAULT                    VARCHAR(200),
   MSG_ERROR                VARCHAR(200),
   PRIMARY KEY (ID_NUMBER), KEY (T1), KEY (T2)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Analysis: compute WSs status by operation

WSs operation Status: probabilities indicating the current estimation of the WS status following a HMM

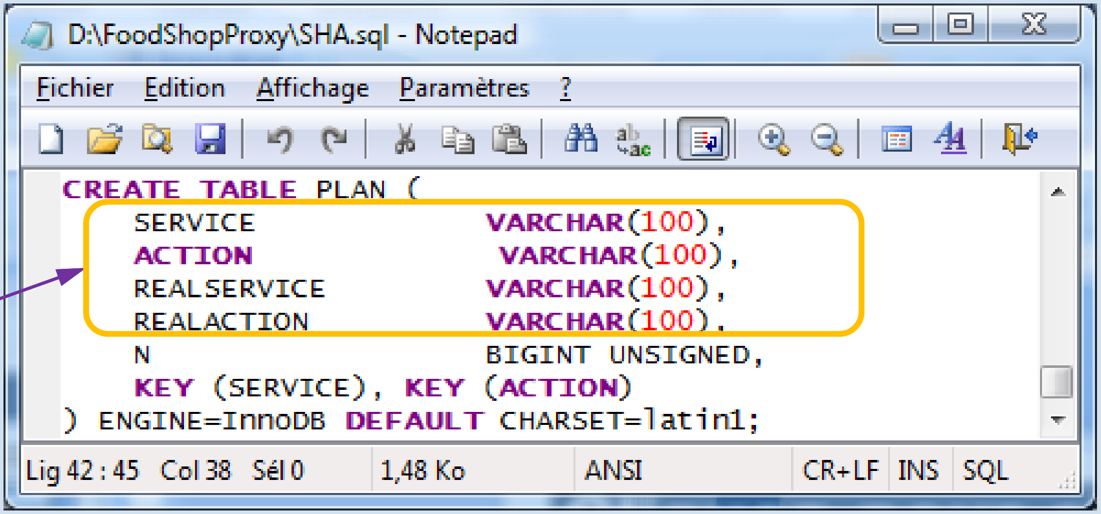
Computed QoS values used as inputs for the estimation process



```
CREATE TABLE STATE (
    SERVICE             VARCHAR(100),
    ACTION               VARCHAR(100),
    DUPLIC              CHAR,
    S_WORKING           FLOAT,
    S_PARTIALLY_WORKING FLOAT,
    S_NOT_WORKING       FLOAT,
    RTT                 FLOAT,
    SRTT                FLOAT,
    RTO                 FLOAT,
    ARTT                FLOAT,
    VARIANCE            FLOAT,
    K                   FLOAT,
    N                   BIGINT UNSIGNED,
    I                   BIGINT UNSIGNED,
    KEY (SERVICE), KEY (ACTION)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

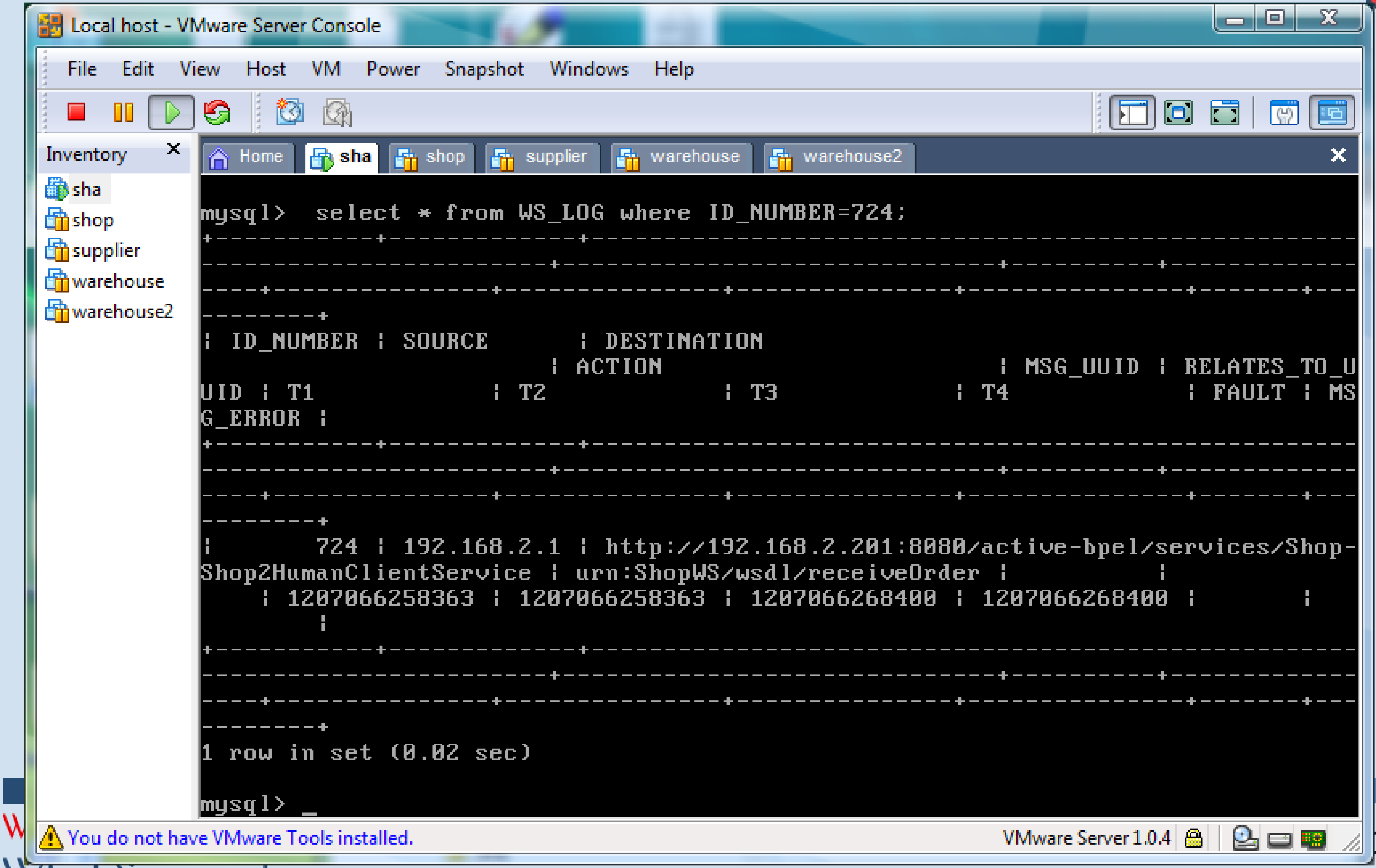
Repair: rerouting requests by operation

Services and operations names and their substitutions according to the reconfiguration decision



```
CREATE TABLE PLAN (
    SERVICE                 VARCHAR(100),
    ACTION                   VARCHAR(100),
    REALSERVICE             VARCHAR(100),
    REALACTION              VARCHAR(100),
    N                       BIGINT UNSIGNED,
    KEY (SERVICE), KEY (ACTION)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Example of the logged traffic monitoring values



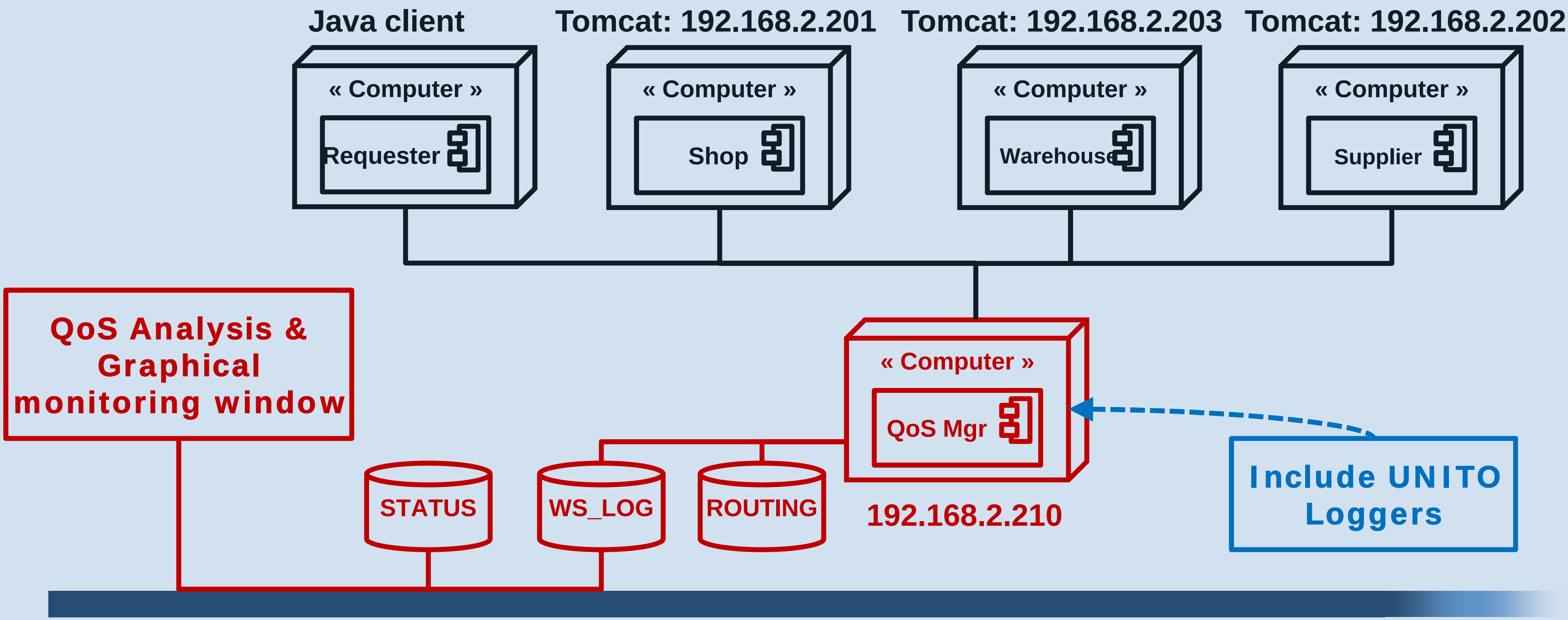
```
mysql>  select * from WS_LOG where ID_NUMBER=724;
+------------+--------------+----------------------------------------------------
-------------------------+-------------------------------+-----------+-----------
----+---------------+---------------+---------------+---------------+-------+---
--------+
| ID_NUMBER | SOURCE      | DESTINATION
                        | ACTION                         | MSG_UUID | RELATES_TO_U
UID | T1            | T2            | T3            | T4            | FAULT | MS
G_ERROR |
+------------+--------------+----------------------------------------------------
-------------------------+-------------------------------+-----------+-----------
----+---------------+---------------+---------------+---------------+-------+---
--------+
|        724 | 192.168.2.1 | http://192.168.2.201:8080/active-bpel/services/Shop-
Shop2HumanClientService | urn:ShopWS/wsdl/receiveOrder |           |
    | 1207066258363 | 1207066258363 | 1207066268400 | 1207066268400 |       |
        |
+------------+--------------+----------------------------------------------------
-------------------------+-------------------------------+-----------+-----------
----+---------------+---------------+---------------+---------------+-------+---
--------+
1 row in set (0.02 sec)

mysql> _
```

Foodshop with centralized QoS Manager

- Current configuration, used for the demonstration
- Configure Tomcat (add the following line in the catalina.sh file):

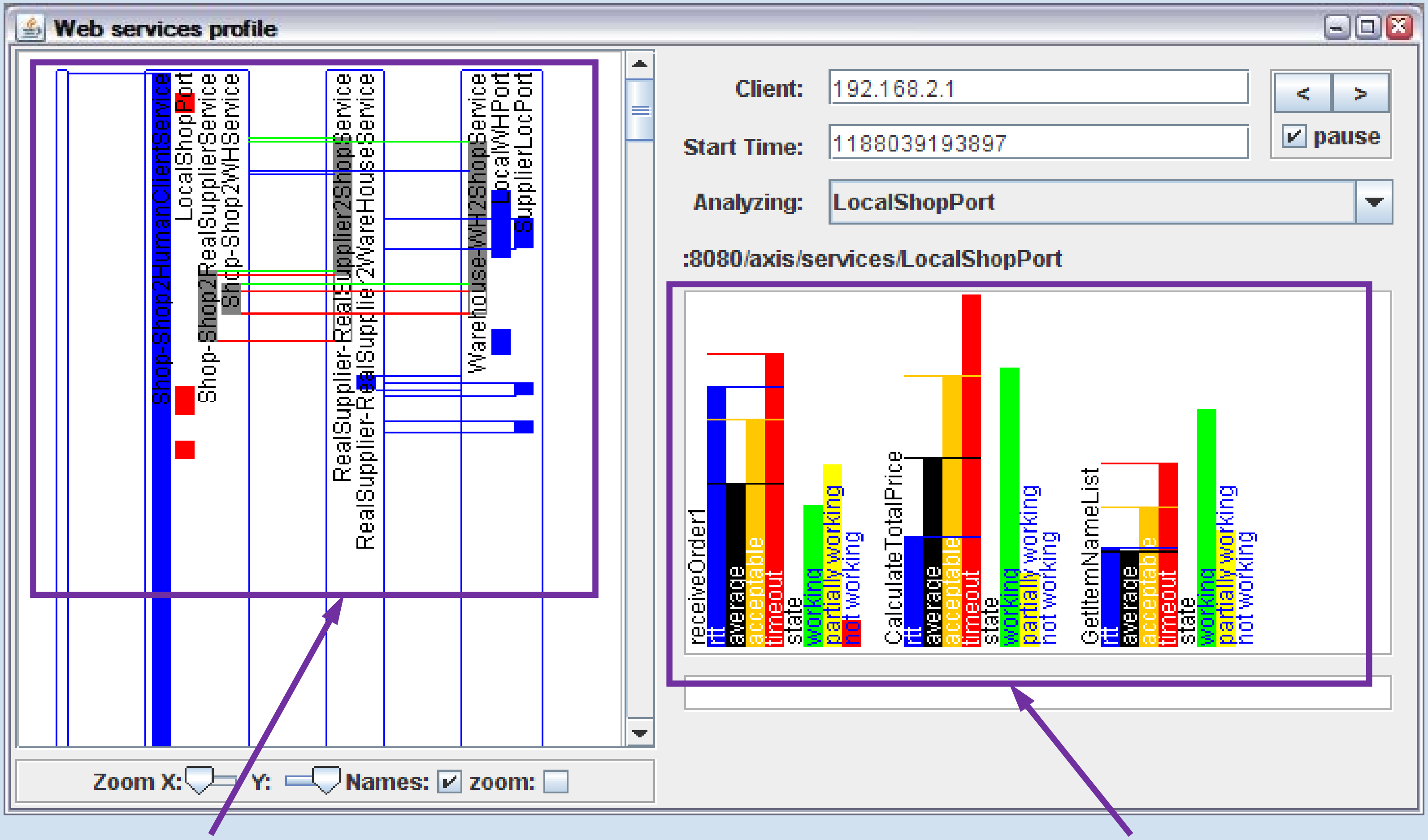
```
JAVA_OPTS= "-Dhttp.proxyHost=192.168.2.210 -Dhttp.proxyPort=8080 -DproxySet=true"
```



Distribution of prototype and application WSs

- Five Deployment computers (associated to five independent virtual machines: VMware)
  - M1: Shop, M2: Warehouse, M3: Supplier
  - M4: Additional Warehouse (for substitution)
  - M5: QoS Manager
- Additional execution computer(real machine)
  - A request generators (Periodic invocation)
  - Graphical monitoring window (status of the WSs)

Graphical monitoring window



Web services profile

Client: 192.168.2.1

Start Time: 1188039193897

☑ pause

Analyzing: LocalShopPort

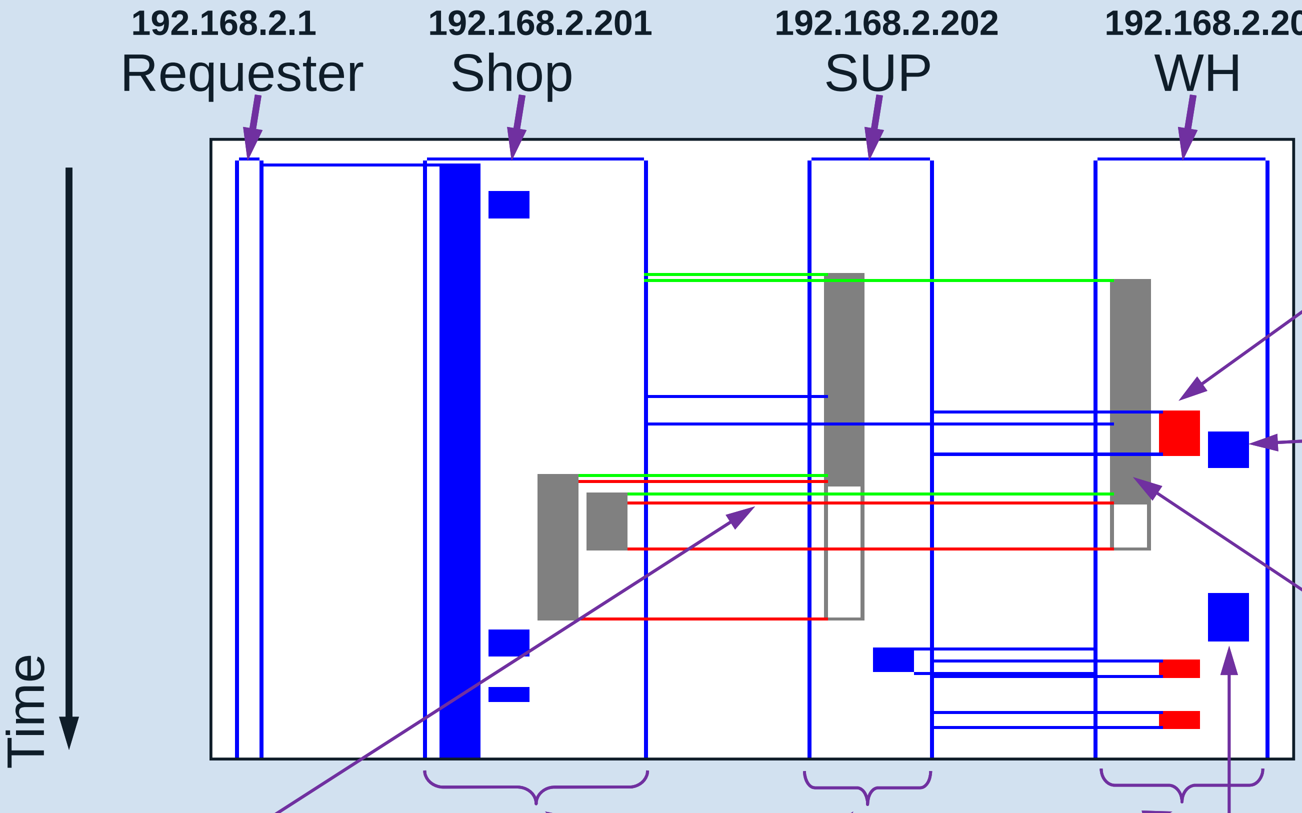
:8080/axis/services/LocalShopPort

Zoom X: Y: Names: ☑ zoom: □

Application profile (computed dynamically from the log)

Statistics of operations inside a service

The application profile = Conversation sequences



Rectangle in red: the selected WS operation for analysis

Rectangle in blue: synchronous operations

Rectangle in gray: asynchronous operations

1Line= 1 message between operations

1 Swimlane= 1 deployment computer

1 Column= 1 operation's executions

Analysis of QoS values for state estimation

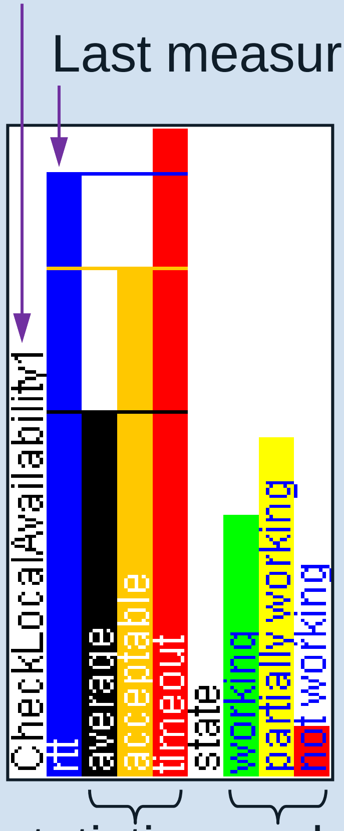
- Statistics
  - 🟦 Round-Trip Time (RTT) = response time
  - ⬛ Average RTT:

    $$SRTT_i = (1-\alpha).SRTT_{i-1} + \alpha.RTT_i$$

  - 🟧 Acceptable Round -Trip Time (ARTT): $ARTT_i = SRTT_i + \frac{K}{2}.\sqrt{\sigma_i^2}$
  - 🟥 Retransmission Timeout (RTO): $RTO_i = SRTT_i + K.\sqrt{\sigma_i^2}$
- Model States
  - Working, PartiallyWorking, NOTWorking
  - Hidden Markov Model

Operation name

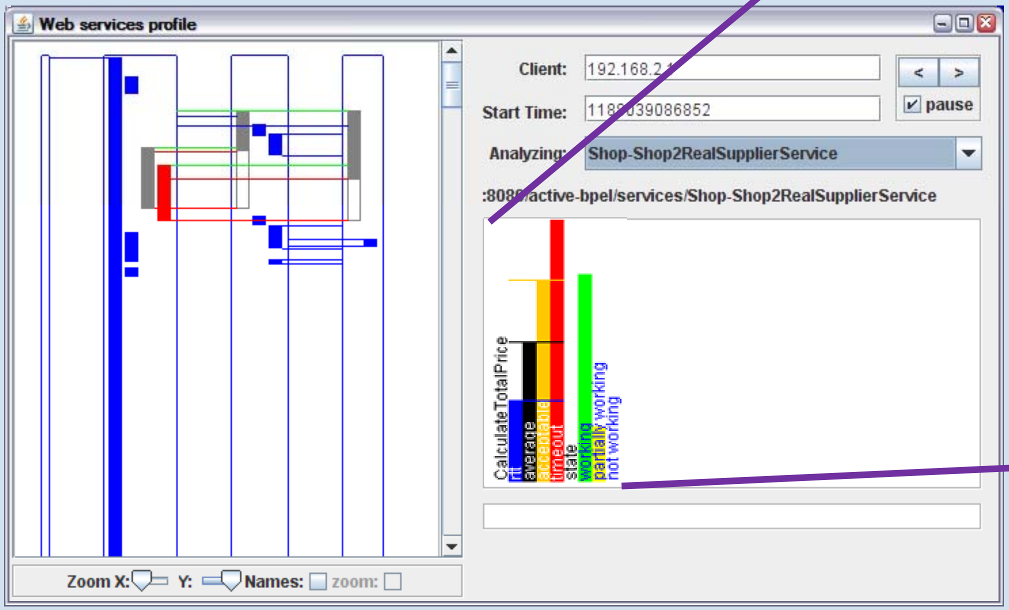
Last measured RTT

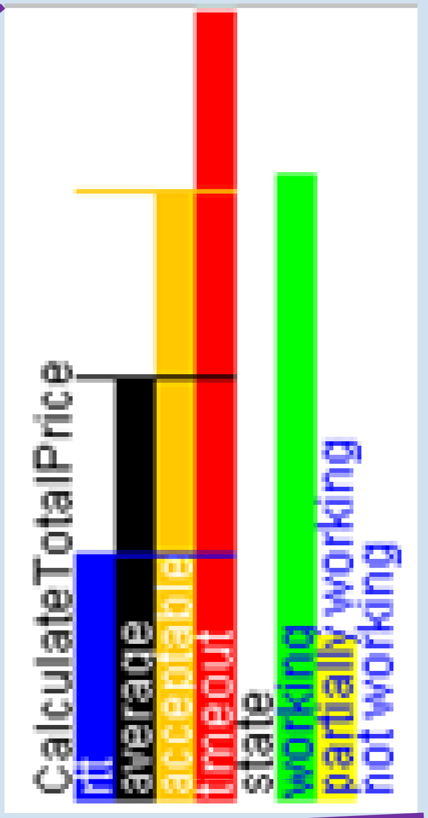


statistics model

State = Working

- A Web service operation in Working state:
  - is working normally (green highest)
  - RTT < ARTT

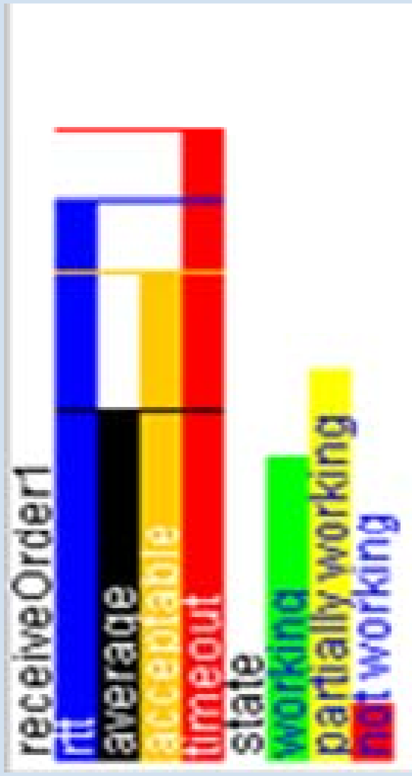




Shop: CalculateTotalPrice()

State = PartiallyWorking

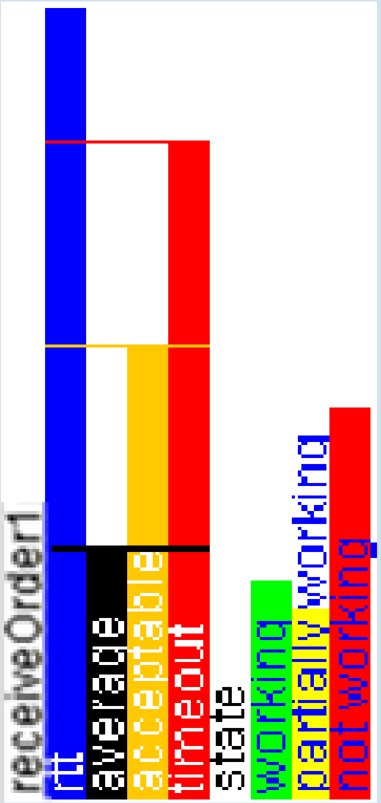
- A Web service operation in PartiallyWorking state: (yellow highest)
  - After some times with ARTT ≤ RTT < RTO
  - Web service is working, but shows some disagreements with the expected QoS



Shop: receiveOrder()

State = NOTWorking

- A Web service operation in NOTWorking state: (red highest)
  - RTT > RTO
  - Web service does not work or frequently disagrees with expected QoS



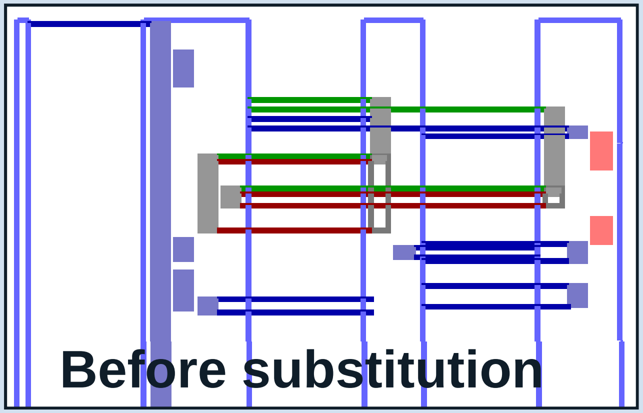
Shop: receiveOrder()

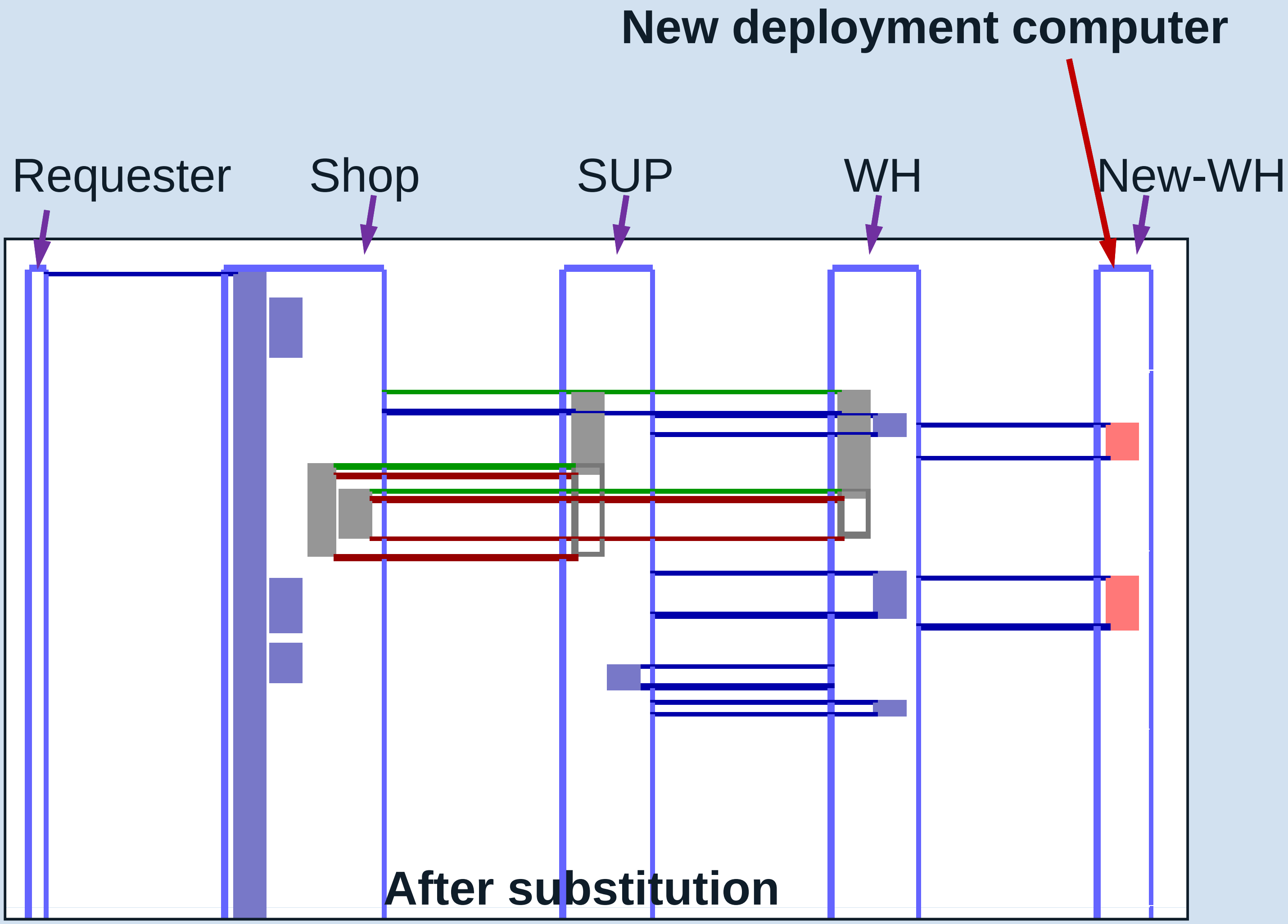
Reconfiguration (1/2)

- New plan for reconfiguration
  - Substitution of a service
  - Substitution of an operation

- 1 plan= 1 sql-request
  - INSERT INTO PLAN SET SERVICE=“old_wsdl_address”, ACTION=“old_operation”,REALSERVICE=“new_wsdl_address”, REALACTION=“new_Operation”;

Reconfiguration (2/2)





Invocation of the operation (Red rectangle) is rerouted towards a New-WH on the new deployment computer

Summary (1/2): QoS prototype implementation

- Prototype V1: [IEEE ISWS/WETICE’07]
  - SOAP-level management:
    - Dynamic connector-based architecture
- Prototype V2: [ICEIS’08]
  - HTTP Proxy : Monitoring and Reconfiguration
    - Integrated and experimented with the FoodShop WS-based application
    - May be adapted for other WS-based applications
  - QoS analysis & Graphical monitoring window
    - Draw application WSs interaction and show status
    - Applied to the FoodShop application log
    - May be integrated with other loggers (as UNITO log)

Summary (2/2): QoS-related studies and models

- Algorithms and frameworks:
  - Local/Global detection algorithms of QoS degradation [IEEE ICADIWT’08]
  - Reconfiguration algorithm and framework: [IEEE ICWS’08]
- Models
  - Degradation detection and source identification chronicles [D3.2]
  - Hidden Markovian Model for QoS-based estimation of WS status [ICEIS’08]
  - Self-healing ontology [DMVE/DEXA’08]

Publications

- **[IEEE ISWS/WETICE'07]**
  - Riadh Ben Halima, Mohamed Jmaiel, and Khalil Drira. *A QoS-driven reconfiguration management system extending Web services with self-healing properties*.
- **[D3.2]**
  - *Specification of execution mechanisms and composition strategies for self-healing Web services. Phase 2*
- **[IEEE ICADIWT'08]**
  - Riadh Ben Halima, Karim Guennoun , Mohamed Jmaiel, and Khalil Drira. *Non-intrusive QoS Monitoring and Analysis for Self-Healing Web Services.*
- **[IEEE ICWS'08]**
  - Riadh Ben Halima, Mohamed Jmaiel, and Khalil Drira. *A QoS-Oriented Reconfigurable Middleware For Self-HealingWeb Services*
- **[ICEIS'08]**
  - René Pegoraro, Riadh Ben Halima, Khalil Drira, Karim Guennoun, and Joao Mauricio Rosrio. A framework for monitoring and runtime recovery of web service-based applications.
- **[DMVE/DEXA'08]**
  - O. Nabuco, R. Ben Halima, K. Drira, M.G. Fugini, S. Modafferi, and E. Mussi. *Model-based QoS-enabled self-healing Web Services.*

Thank you