Project Number:	**IST-1999-10077**
Project Title:	 **Adaptive Resource Control for QoS Using an IP-based Layered Architecture**
Deliverable Type:	**PB – public**

Deliverable Number:	**IST-1999-10077-WP3.1-NTU-3102-PU-R/b0**
Contractual Date of Delivery to the CEC:	**October 31, 2002**
Actual Date of Delivery to the CEC:	**October 30, 2002**
Title of Deliverable:	**Second trial integration report**
Workpackage contributing to the Deliverable:	**WP 3.1**
Nature of the Deliverable:	**R – Report**
Editor:	**Petros Sampatakos (NTU)**
Author(s):	**R. Frank, M. Winter (SAG), F. Kemmel, A. Thomas (TUD), W. Burakowski, M. Dabrowski, A. Bak, A. Beben (WUT), Z. Kopertowski (TPS), D. Katzengruber, G. Pieler, M. Titze (TAA), J. Karadimas (QSY), N. Miettinen, T. Kilkanen (ELI), F. Strohmeier, T. Pfeiffenberger, M. Herfurt (SPU), R. Widera, H. Doerken, J. Mende, C. Heidemann (DTA), P. Sampatakos, L. Dimopoulou, E. Nikolouzou, C. Tsetsekas (NTU)**

Abstract:	This document describes the results of the second trial integration of the AQUILA network. The tests needed for the verification of its functionality and the procedure for the set up of new trial sites are presented.
Keyword List:	AQUILA, integration, inter-domain, testbed, Internet, test

Executive Summary

The present document is the second major deliverable provided by WP 3.1 of the AQUILA project. It describes in detail the procedures that took place for the verification of the correct operation of the AQUILA Resource Control Layer, Inter-domain Layer, Management Layer and Measurement Tools. It also contains instructions for the installation of the working prototypes in other trial sites as well as instructions for their use during the trials.

The second trial required the setup of three trial sites, Vienna, Warsaw and Helsinki. The network topology as well as the hardware/software components installed can be found in chapter 3.

One of the main challenges of the second trial, from the integration point of view, was the interconnection of the trial sites (namely Vienna and Warsaw). Towards this end, the assistance of the operators of the European research network (GEANT) has been requested. The response was imminent and thanks to a fruitful cooperation the interconnecting link was up on time. Test cases regarding the interconnectivity have been successfully performed.

In addition, new features and enhancements of the AQUILA software components have been introduced since the first trial that should be tested to ensure their proper functionality. Therefore, corresponding tests have been carried out in order to validate the implementation. The test cases included the validation of the:

- MBAC implementation for traffic classes (TCL) 1 and 3
- new admission control algorithms for TCL3
- Joint admission control implementation on high speed links
- Inter-domain admission control implementation
- secondary access link admission control implementation
- resource pool mechanisms

Along with the aforementioned tests, the main functionality of the AQUILA components has been also examined. Several experiments have been made to assure the correct interoperability between all the packages. It should be noted here that all the tests were carried out successfully.

The document is structured as follows: in section 2 the integration procedure and the time plan are provided. In section 3, the topology and the equipment of the three trial sites (Vienna, Warsaw and Helsinki) are described in detail. Section 4 includes the test cases used for the verification of the produced software and the validation of the implementation of the newly introduced enhancements. In section 5, detailed instructions for the establishment of new trial sites can be found, including the requirements for each component along with the user guide. The last section summarizes the results and experiences of the integration phase.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The present deliverable describes the methodology and the steps followed in order to carry out the integration deployment of the AQUILA experimental network for the second trial. It also provides detailed instructions for the establishment of new trial sites.

The second trial required the setup of three trial sites, Vienna, Warsaw and Helsinki. The network topology as well as the hardware/software components installed can be found in chapter 3.

One of the main challenges of the second trial, from the integration point of view, was the interconnection of the trial sites (namely Vienna and Warsaw). Towards this end, the assistance of the operators of the European research network (GEANT) has been requested. The response was imminent and thanks to a fruitful cooperation the interconnecting link was up on time. Test cases regarding the interconnectivity have been successfully performed as described in chapter 4.1.

In addition, new features and enhancements of the AQUILA software components have been introduced since the first trial that should be tested to ensure their proper functionality. Therefore, corresponding tests have been carried out in order to validate the implementation. The test cases included the validation of the:

- MBAC implementation for traffic classes (TCL) 1 and 3
- new admission control algorithms for TCL3
- Joint admission control implementation on high speed links
- Inter-domain admission control implementation
- secondary access link admission control implementation
- resource pool mechanisms

The details for these test cases are provided in chapters 4.2 and 4.3.

Along with the aforementioned tests, the main functionality of the AQUILA components has been also examined. Several experiments have been made as described in chapter 4.4. All the tests were carried out successfully.

To facilitate the easier setup and operation of the trial sites, installation guides and user manuals have been included, as can be seen in chapter 5.

Finally, chapter 6 summarizes the results and experiences gained from the integration procedure.

2 Integration methodology and time plan

In order to fulfil the objective of integrating the AQUILA systems and components, a phased integration methodology was followed. The integration process is composed of different steps, each of which requires the verification of specific characteristics of the equipment. The output of each phase should be provided as an input to the next one, so that an add-on approach is followed.

Three distinct phases have been identified:

- Pre-integration
- Overall integration (Starting at Vienna integration meeting)
- Validation of the Implementation

The time plan followed by the different integration steps can be seen in the following figure:

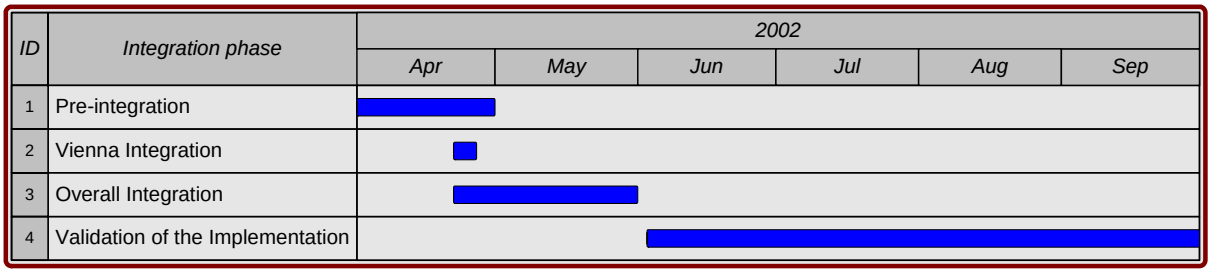


Figure 2-1: Integration time plan

During the pre-integration phase all the necessary procedures and preparations took place in order to assure that the integration sites (namely Warsaw, Vienna and Helsinki) will be ready for the overall integration phase. The procedures included the setup of the interconnection between the Vienna and Warsaw trial site via GEANT (inter-domain link) and the link between Vienna and the Salzburg (intra-domain link). An IP tunnel has been used in order to preserve the marking of the packets and provide a transparent connection between the testbeds.

The first phase took place 1-19 April while the overall integration was started at the integration meeting in Vienna 22-26 April. During the meeting, all the components were successfully installed and tested in the local trial site while the other two remote sites have been basically configured.

In the next weeks the configuration of Warsaw and Helsinki testbeds has been completed (including the integration of MEDIAZINE in Warsaw testbed) facilitating the beginning of the "Validation of the Implementation" phase. During that phase the newly introduced enhance-

ments of the software components have been tested. A detailed description of the test cases and the results can be found in chapter 4.

3 Integration sites topology

This section contains the descriptions of the partners involved in the integration phase. As already mentioned for the second trial three sites are forming the testbed, Vienna, Warsaw and Helsinki.

3.1 Integration at Vienna

3.1.1 Network topology

3.1.2 TAA testbed for the 2nd trial

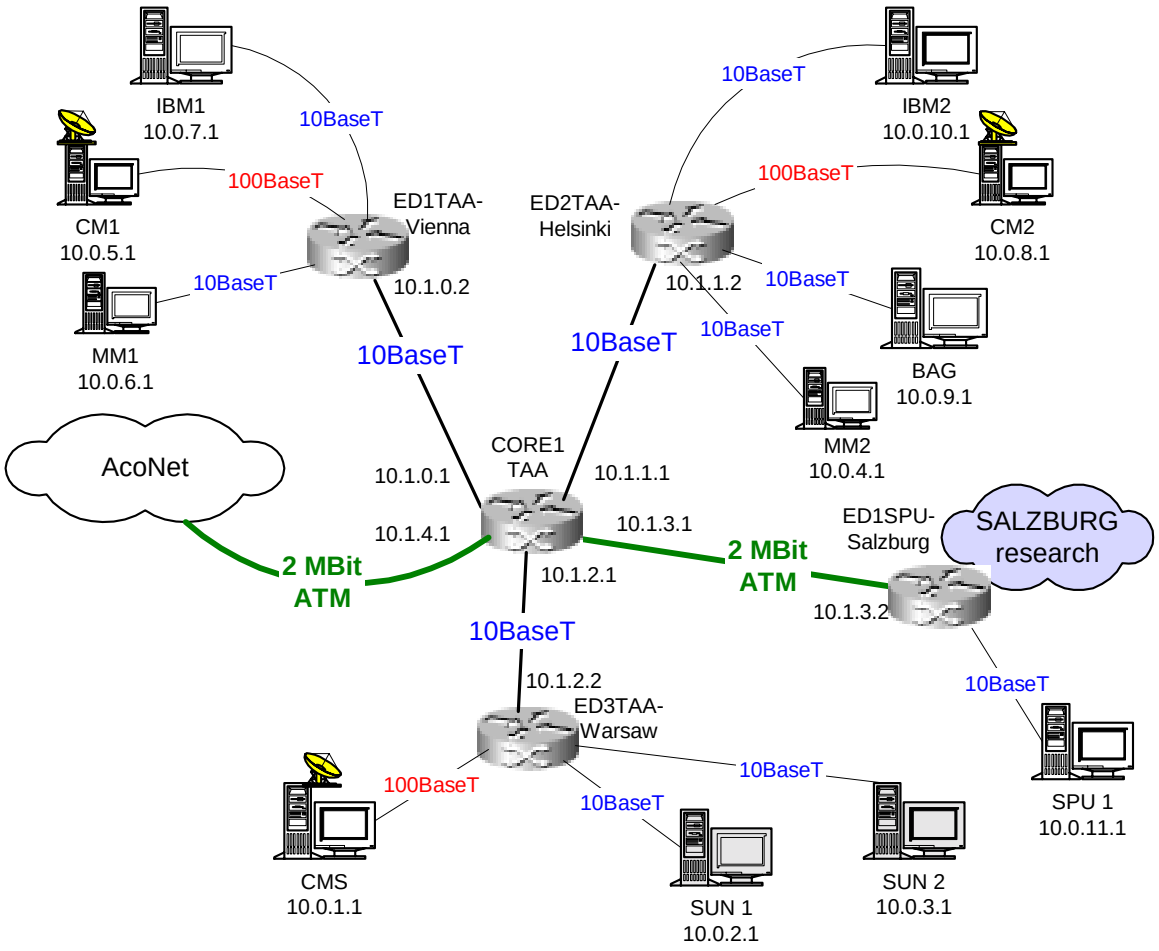


Figure 3-1: TAA testbed for the 2nd trial

3.1.2.1 Network addressing scheme

In order to guarantee reachability in networks for each router a loopback interface (address) is configured.

	CORE1TAA	ED1TAA	ED2TAA	ED3TAA	ED1SPU
Loopback IP	10.2.0.1 /30	10.2.1.1 /30	10.2.2.1 /30	10.2.3.1 /30	10.2.4.1 /30

Table 3-1: Loopback addresses

Please Note that the router interface address of the connected hosts is 10.0.x.254, which represents the default gateway for the hosts.

The addressing scheme is summarised in the following table:

	IP-address	Subnet mask
LAN Segments	10.0.x.1	255.255.255.0
Router addresses	10.1.x.1 – 10.1.x.2	255.255.255.252
Default Gateway	10.x.x.254	255.255.255.0
Loopback address	10.2.x.1	255.255.255.252

Table 3-2: TAA addressing scheme

The following table indicates the hosts and their designated usage.

Host	Usage	GPS	OS	Software
CMS	Measurement Server	✓	Linux7.3	
CM1	Measurement Client	✓	Linux7.3	
CM2	Measurement Client	✓	Linux7.3	
BAG	Server	✓	Linux7.3	DNS
MM1	Client		Linux7.3 / W98	NetMeeting
MM2	Client		Linux7.3 / W98	NetMeeting
IBM1	Client		Linux7.3 / W2k	NetMeeting, SIGMA

IBM2	Client		Linux7.3 / W2k	NetMeeting, SIGMA
SUN1	RCA, ACA, EAT		Solaris 5.6	
SUN2	RCA, ACA, EAT		Solaris 5.6	

Table 3-3: Designated usage of the hosts

3.1.2.2 OSPF configuration

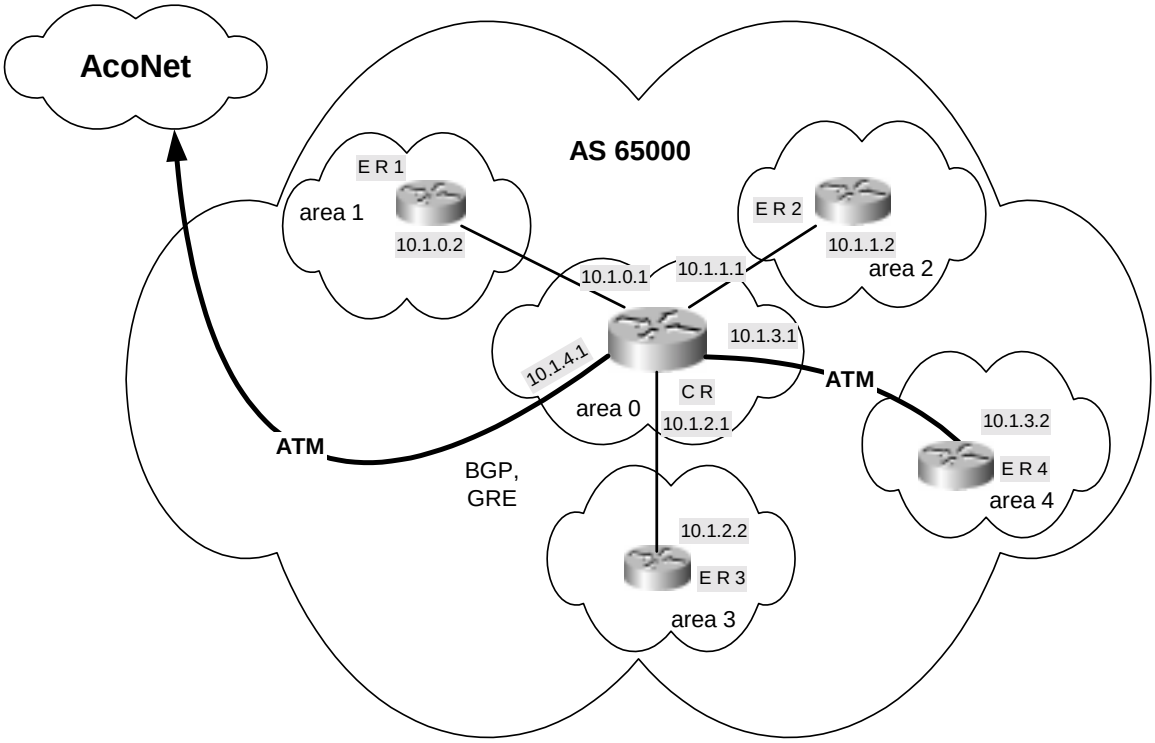


Figure 3-2: OSPF configuration

Router ED1 – ED4 act as “Area Border Router” whereas the core router (CR) represents an “Autonomous System Boundary Router”.

Configuration example for ED1:

```
interface ethernet 0/0
ip address 10.1.0.2 255.255.255.252

interface ethernet 0/1
ip address 10.0.6.254 255.255.255.0

interface ethernet 0/2
ip address 10.0.7.254 255.255.255.0
```

```
interface ethernet 3/0
ip address 10.0.5.254 255.255.255.0

router ospf 65000
network 10.0.0.0 0.0.255.255 area 1
network 10.1.0.0 0.0.255.255 area 0
```

Configuration example for CR:

```
interface ethernet 3/0
ip address 10.1.0.1 255.255.255.252

interface ethernet 3/1
ip address 10.1.1.1 255.255.255.252

interface ethernet 3/2
ip address 10.1.2.1 255.255.255.252

interface ATM1/0.100 point-to-point
 description aquila salzburg
 ip address 10.1.3.1 255.255.255.252
 ip nat inside
 pvc 100/100
  vbr-nrt 4000 2000 192
  encapsulation aal5snap

interface ATM1/0.101 point-to-point
 description aquila warsaw
 ip address 10.1.4.1 255.255.255.252
 ip nat inside
 pvc 101/101
  vbr-nrt 4000 2000 192
  encapsulation aal5snap

router ospf 65000
network 10.1.0.0 0.0.255.255 area 0
redistribute bgp 65000 metric 1 metric-type 1

router bgp 65000
network 10.0.0.0 0.0.255.255
network 10.1.0.0 0.0.255.255
network 10.2.0.0 0.0.255.255
neighbor 10.10.x.x 0.0.255.255 remote-as 65010
neighbor 10.11.x.x 0.0.255.255 remote-as 65010
neighbor 10.12.x.x 0.0.255.255 remote-as 65010
```

3.1.3 Proposal for interconnection addressing scheme

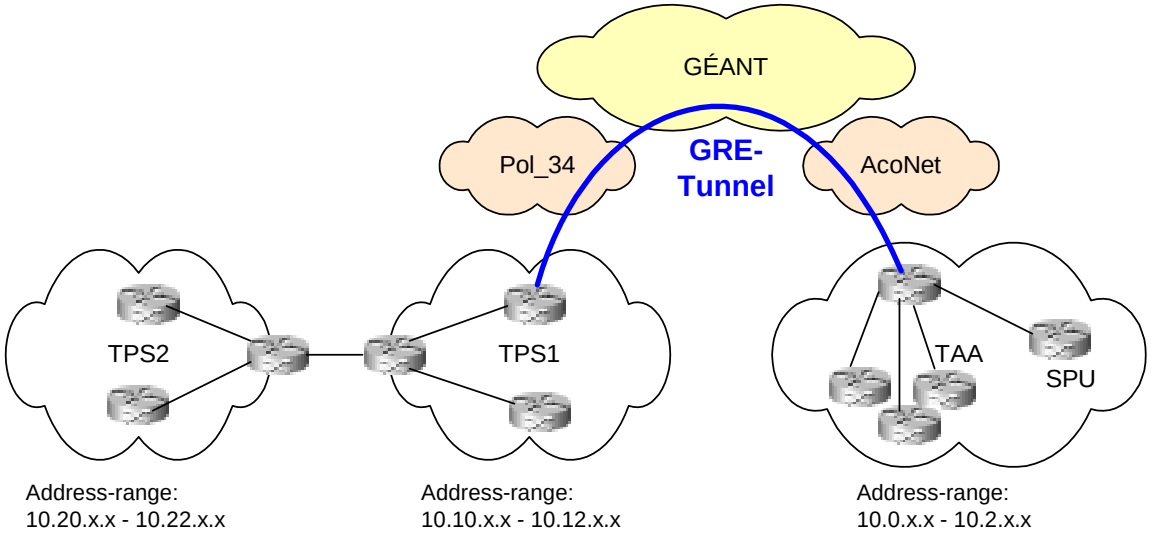


Figure 3-3: Interconnection addressing scheme

The addressing scheme is summarised in the following table:

	TPS 2	TPS 1	TAA
	IP / Subnet	IP / Subnet	IP / Subnet
Autonomous System (AS)	65020	65010	65000
LAN Segments	10.20.x.1 /24	10.10.x.1 /24	10.00.x.1 /24
Router addresses	10.21.x.x /30	10.11.x.x /30	10.01.x.x /30
Default Gateway	10.2x.x.254 /24	10.1x.x.254	10.0x.x.254 /24
Loopback address	10.22.x.1 /30	10.12.x.1 /30	10.02.x.1 /30

Table 3-4: Proposal for an inter-domain addressing scheme

The various x represents the different network addresses, which have to configured. Please have a look to the detailed configuration example in Figure 3-1.

The addressing scheme could also be extended by the Helsinki testbed (10.30.x.x – 10.32.x.x).

3.1.4 Hardware / Software equipment

The equipment described below is dedicated to the AQUILA project for the whole period of its duration. We will use CISCO routers exclusively.

3.1.4.1 List of routers

4 CISCO routers will be available in the TAA network laboratory. 1 additional router situated in Salzburg and connected via a 2Mbit/s ATM link is also available in the TAA testbed.

Router Cisco 7500 - (1 router)	
Description	**Number of modules**
CISCO 7500	1
IOS: 12.1(4)E, Feature Set: IP	
3 Ethernet 10BaseT Ports	1
1 ATM module	1

Router Cisco 3640 (3 routers)	
Description	**Number of modules**
CISCO 3600 4-slot Modular Router	1
IOS: 12.1(2)T, Feature Set: IP PLUS	
4-Port Ethernet Network Module	1
1-Port Fast Ethernet Network Module	1

Router Cisco 7204 - (1 router)	
Description	**Number of modules**
CISCO 7204	1
IOS: 12.2(7), Feature Set: IP-PLUS	

Ethernet 10BaseT Ports	1
1 ATM module	1

3.1.4.2 Terminals – hardware

- 4 (relatively) Hi-Performance Client-PCs

CASE	NN Case MIDI-Tower ATX
MAINBOARD	MSI K7T-PRO DURON Socket A Audio
CPU	AMD T800 Socket A (800 MHz)
RAM	128M-168P SDRAM 100MHz
HD	Seagate ST320423 20,4GB; U10, U/66, 8,9ms, 5400RPMs
CD	Creative 52x DIE
NIC	3Com PCI10/100 TP/BNC/AUI
AUDIO	Soundblaster Live 1024 (for 2 PCs)
VIDEO	3D Prophet II Gforce2 MX

Table 3-5: Client PC hardware description

- 2 (relatively) Hi-Performance Server-PCs

CASE	NN Case MIDI-Tower ATX
MAINBOARD	MSI K7T-PRO DURON Socket A Audio
CPU	AMD T800 Socket A
RAM	256M-168P SDRAM 100MHz
HD	Seagate ST320423 20,4GB; U10, U/66, 8,9ms, 5400RPMs
CD	Creative 52x DIE
NIC	3Com PCI10/100 TP/BNC/AUI

VIDEO	ATI XPERT2000 AGP 16MB

Table 3-6: Server PC Hardware description

- 2 Hi-Performance Client-PCs

CASE	IBM Desktop (NetVista)
MAINBOARD	?
CPU	Intel® Pentium III, 1GHz
RAM	256MB
HD	20,4GB
CD	Samsung CD-ROM SC-148C
NIC	Intel® PRO/100 VE Desktop Connection
AUDIO	Creative Soundblaster AudioPCI
VIDEO	Intel® 82815 Graphics Controller

Table 3-7: IBM-Client PC description

- 2 Sun Solaris

Type	SUN Ultra 30
MAINBOARD	PCI Bus mit 66 MHz
CPU	300 MHz Ultra Sparc II Prozessor
RAM	256 MB RAM
HD	5 GB HDD
NIC	10 Mbit/s

Table 3-8: Server PC Hardware description

3.1.4.3 Additional equipment

For the user impressions, we propose to use standard PC "WebCams" and headsets as used most commonly.

WebCam	Creative WebCam3 USB, (Colour video writes at 30 fps at 320x240 in 16 million colours, at 640X480 v up to 15 fps)
Headset	Plantronics headset

Table 3-9: Additional equipment

3.1.4.4 Operating Systems

The following table summarises the available operating systems and their planed usage.

OS	Usage
Windows 98 SE	NetMeeting
Windows 2000	NetMeeting, SIGMA, Mediazine
Linux 7.3	Measurement (Server and Client)
Sun Solaris 5.6	RCL (RCA, ACA, EAT)

Table 3-10: Operating system and usage

3.2 Integration at Warsaw

3.2.1 Network topology

3.2.1.1 Single domain network topology in the Warsaw testbed

This topology will be used for Network Services and Real Users trials

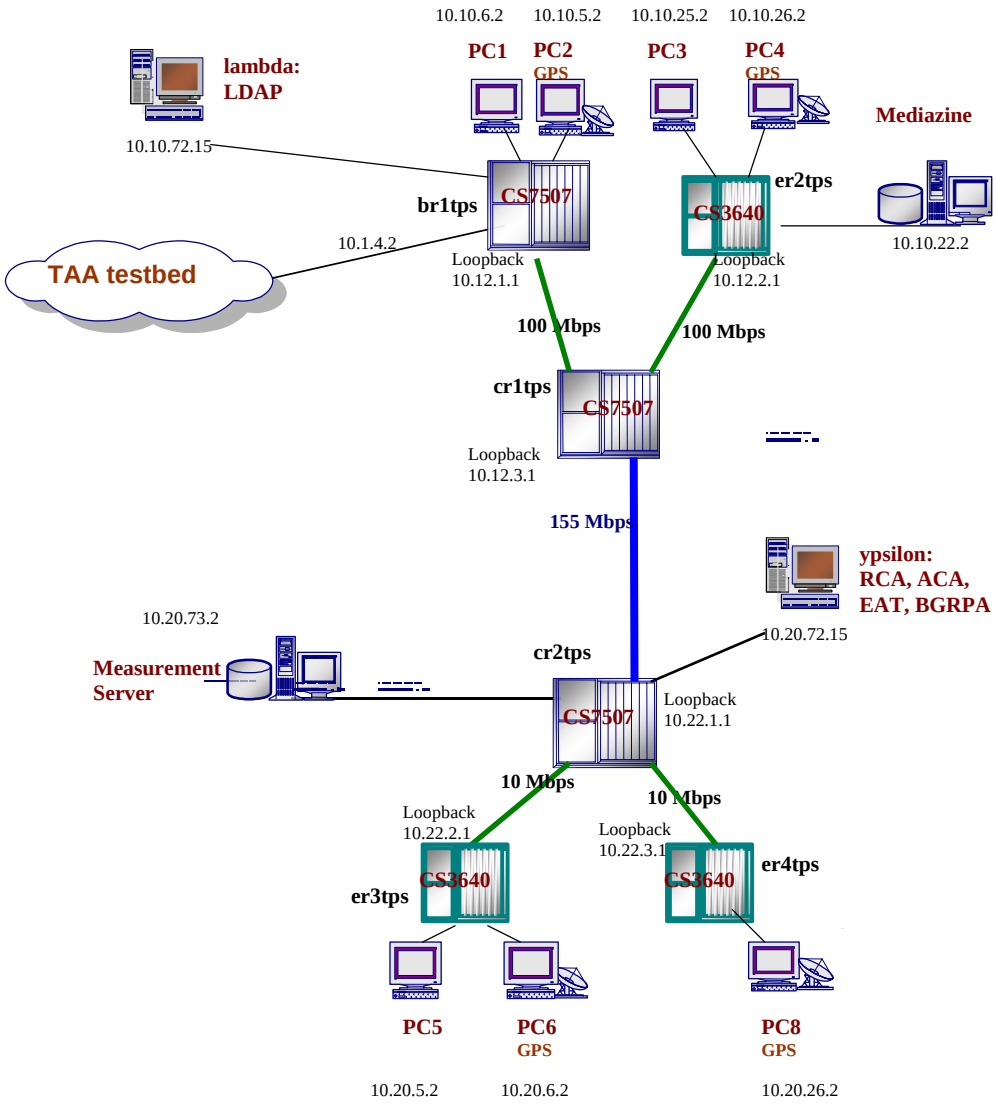


Figure 3-4: Single domain network topology

The addressing scheme is the following:

	Domain TPS	
	IP / Subnet	IP / Subnet
Autonomous System (AS)	65010	
LAN Segments	10.10.x.1 /24	10.20.x.1 /24
Loopback address	10.12.x.1 /30	10.22.x.1 /30

Table 3-11: Single domain addressing scheme

3.2.1.2 Topology with 2 domains in the Warsaw Testbed

This topology will be used for Inter-Domain Network Services trial

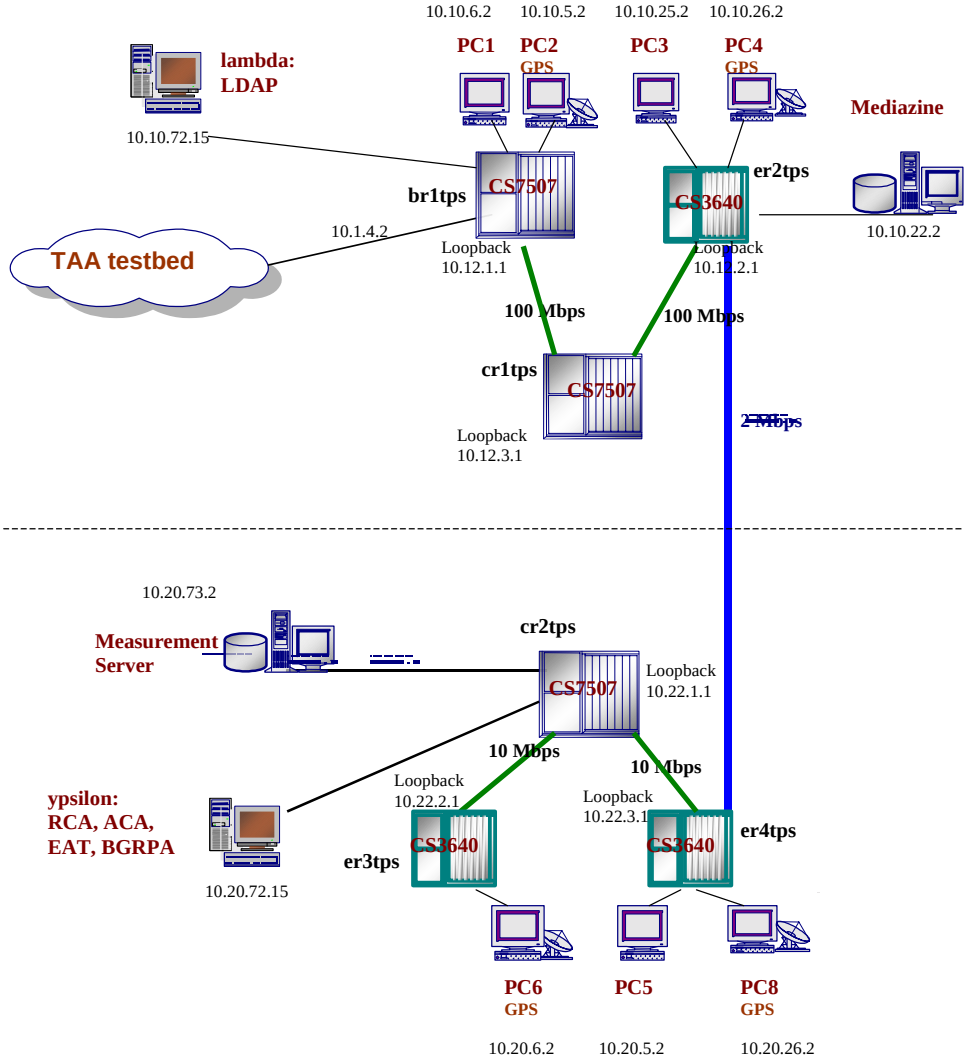


Figure 3-5: Two domains network topology

The addressing scheme is the following:

	Domain TPS 1	Domain TPS 2
	IP / Subnet	IP / Subnet
Autonomous System (AS)	65010	65020
LAN Segments	10.10.x.1 /24	10.20.x.1 /24

Loopback address	10.12.x.1 /30	10.22.x.1 /30

Table 3-12: Two domains addressing scheme

3.2.2 Hardware / Software equipment

The following routers will be used in the trial network:

- CISCO 7507 (3 routers).

IOS software release	IOS (tm) RSP Software (RSP-ISV-M), Version 12.1(4)E, EARLY DEPLOYMENT RELEASE SOFTWARE (fc1) rsp-isv-mz.121-4.e.bin
Router central processor	Cisco RSP4+ (R5000) processor with 131072K/2072K bytes of memory. R5000 CPU at 200Mhz, Implementation 35, Rev 2.1, 512KB L2 Cache
Interface processors	4 VIP4-50 RM5271 controllers

Table 3-13: CISCO 7507 description

- CISCO 3640 (3 routers).

IOS software release	IOS (tm) 3600 Software (C3640-IS-M), Version 12.1(2), RELEASE SOFTWARE (fc1) c3640-is-mz.121-2
Router processor	Cisco 3640 (R4700) processor (revision 0x00) with 36864K/12288K bytes of memory. R4700 CPU at 100Mhz, Implementation 33, Rev 1.0

Table 3-14: CISCO 3640 description

Other available equipment:

SUN workstations
2x Sun Ultra 60 with Solaris 8
PC computers

8 Pentium III 700MHz computers • 2 with Windows 2000/Linux SuSe 7.3 operating systems • 5 with Windows NT4.0/Linux SuSe 7.3operating systems • 1 with Linux only (Measurement Station) 1 Pentium II 450MHz • Linux SuSe 7.3 operating system
Cameras
2 USB Creative WebCam cameras
GPS equipment
1 antenna with distributor
4 GPS cards
Commercial measurement equipment
Agilent BSTS
Agilent Router Tester
InterWatch 95000

Table 3-15: Other available equipment description

3.3 Integration at Helsinki

3.3.1 Network topology

3.3.1.1 Intra-domain Testbed for signalling

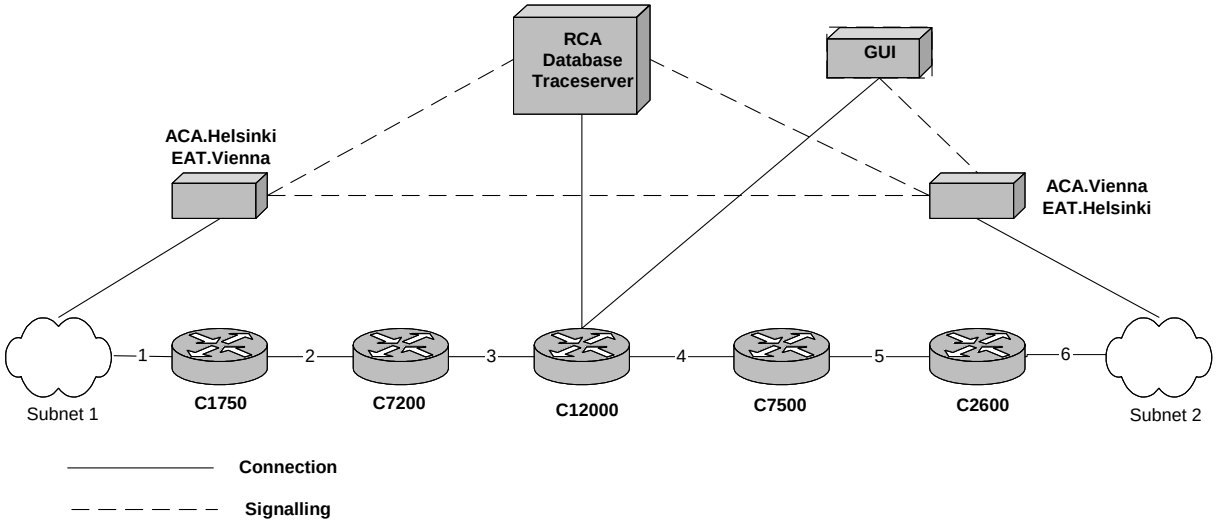


Figure 3-6: Intra-domain testbed for signalling

Router	Host-Name	Link / Interface	IP Address
Subnet 1		1	192.168.0.0/24
C1750	er1eli	1 / FastEthernet 0	192.168.0.1
C1750	er1eli	2 / Serial 0	192.168.1.102
C7200	cr2eli	2 / Serial 4/0	192.168.1.101
C7200	cr2eli	3 / POS 3/0	192.168.1.78
C12000	cr5eli	3 / POS 0/1	192.168.1.77
C12000	cr5eli	4 / ATM 3/0.40	192.168.1.29
C7500	cr3eli	4 / ATM 1/0.10	192.168.1.30
C7500	cr3eli	5 / Serial 0/1/0	192.168.1.113
C2600	er4eli	5 / Serial 0/0	192.168.1.114
C2600	er4eli	6 / FastEthernet 0/0	192.168.2.0/24
Subnet 2		6	192.168.2.1
RCA/DB/Traceserver	rca1eli		Pulivari
ACA.Helsinki	aca1eli		Paarma
ACA.Vienna	aca2eli		MSM

Table 3-16: IP addresses and names of the testbed devices

3.3.1.2 Inter-domain Testbed for signalling

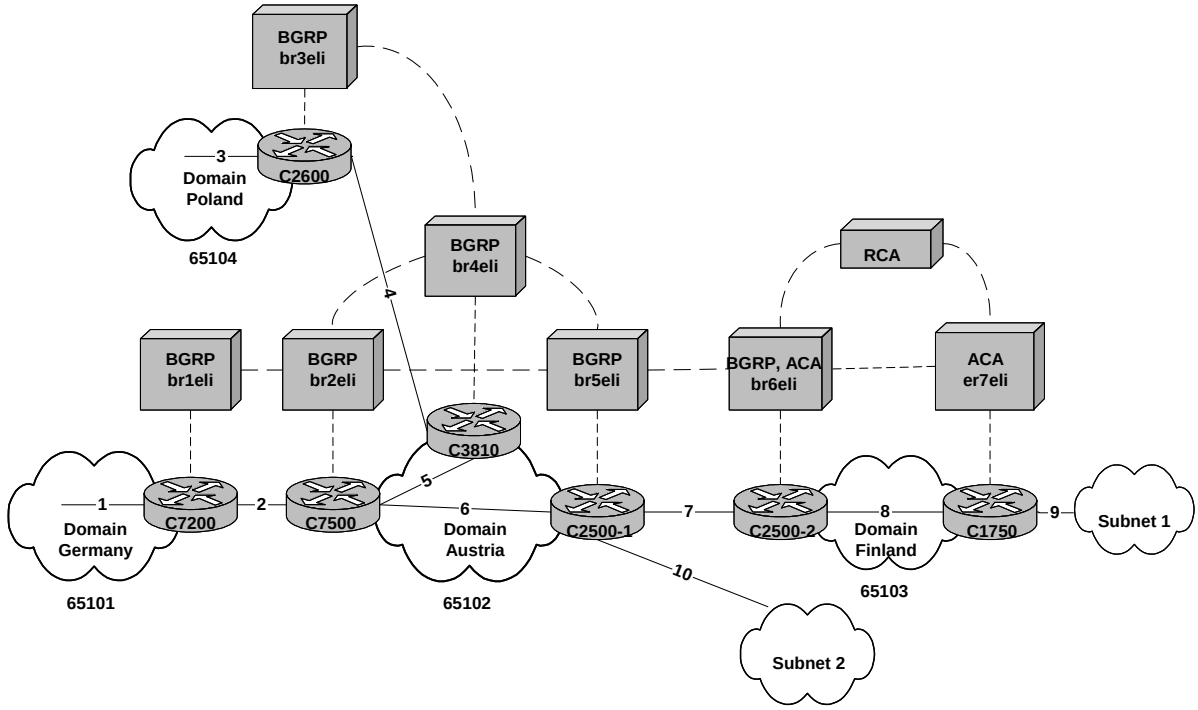


Figure 3-7: Inter-domain testbed for signalling

Router	Name	Link / Interface	IP Address	Loopback
Domain Germany		1	192.168.2.0/24	
C7200	br1eli	1 / FastEthernet 2/0	192.168.2.1/24	192.168.12.1
C7200	br1eli	2 / ATM 1/0.40	192.168.3.1/24	
C7500	br2eli	2 / ATM 1/0/0.30	192.168.3.2/24	192.168.12.2
Domain Poland		3	192.168.4.0/24	
C2600	br3eli	3 / FastEthernet 0/0	192.168.4.1/24	192.168.12.3
C2600	br3eli	4 / Serial 0/1	192.168.5.1/24	
C3810	br4eli	4 / Serial 0	192.168.5.2/24	192.168.12.4
C7500	br2eli	5 / Serial 0/1/1	192.168.6.1/24	
C3810	br4eli	5 / Serial 1	192.168.6.2/24	
C7500	br2eli	6 / Serial 0/1/2	192.168.7.1/24	

C2500-1	br5eli	6 / Serial 0	192.168.7.2/24	192.168.12.5
C2500-1	br5eli	7 / Serial 1	192.168.8.1/24	
C2500-2	br6eli	7 / Serial 0	192.168.8.2/24	192.168.12.6
C2500-2	br6eli	8 / Serial 1	192.168.9.1/24	
C1750	er7eli	8 / Serial 0	192.168.9.2/24	192.168.12.7
C1750	er7eli	9 / FastEthernet 0	192.168.10.1/24	
Subnet 1		9	192.168.10.0/24	
C2500-1	br5eli	10 / Ethernet 0	192.168.11.1/24	
Subnet 2		10	192.168.11.0/24	
RCA for all do-mains / DB			Pulivari	192.168.13.1
ACA br1eli, br2eli, br3eli, br4eli, br5eli, br6eli, br7eli			Pulivari	
BGRPA br1eli, br3eli, br5eli		br1eli / port 2001, br3eli / port 2003, br5eli / port 2005	Paarma	192.168.13.2
BGRPA br2eli, br6eli		br2eli / port 2002, br6eli / port 2006	MSM	192.168.13.3
BGRPA br4eli		br4eli / port 2004	Verkkolab 3	192.168.13.4
EAT br1eli, br3eli, er7eli			Pulivari	

Table 3-17: IP addresses and names of the testbed devices

3.3.1.3 Secondary Access Link Testbed

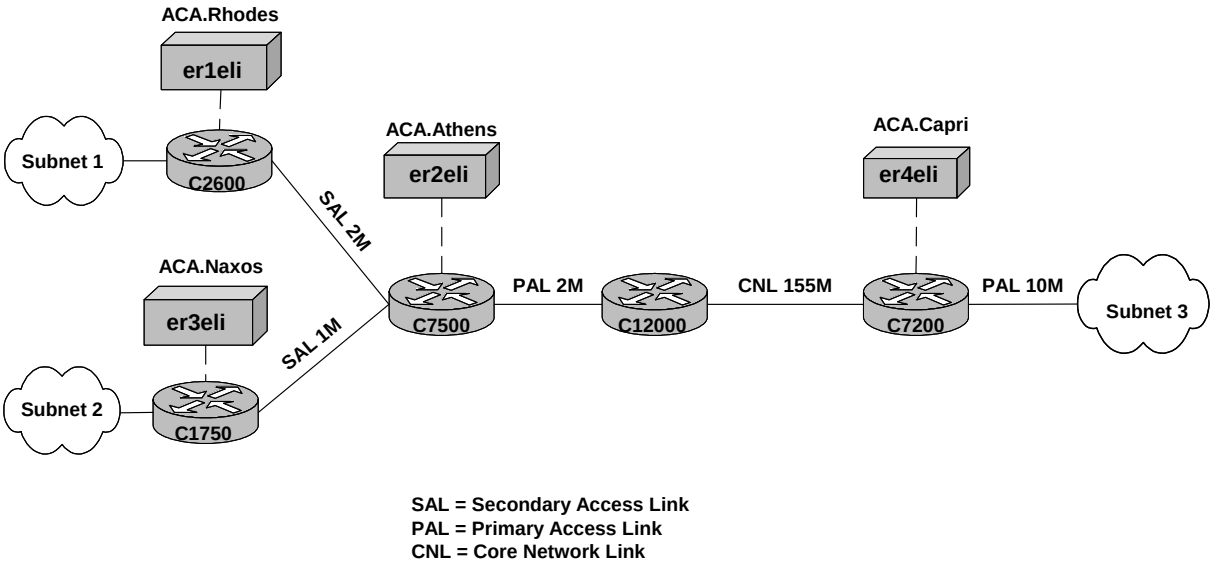


Figure 3-8: Secondary access link testbed

Router	Host-Name	Link / Interface	IP Address / Name
Subnet 1			192.168.2.0/24
C2600	er1eli	FastEthernet 0/0	192.168.2.1/24
C2600	er1eli	SAL 2M / Serial 0/0	192.168.3.1/24
C7500	er2eli	SAL 2M / Serial 0/1/0	192.168.3.2/24
Subnet 2			192.168.4.0/24
C1750	er3eli	FastEthernet 0	192.168.4.1/24
C1750	er3eli	SAL 1M / Serial 0	192.168.5.1/24
C7500	er2eli	SAL 1M / Serial 0/1/0	192.168.5.2/24
C7500	er2eli	PAL 2M / ATM 1/0/0.30	192.168.6.1/24
C12000	cr5eli	PAL 2M / ATM 3/0.40	192.168.6.2/24
C12000	cr5eli	CNL 155M / POS 0/1	192.168.7.1/24
C7200	er4eli	CNL 155M / POS 3/0	192.168.7.2/24
C7200	er4eli	PAL 10M / FastEthernet 2/0	192.168.8.1/24
Subnet 3		PAL 10 M	192.168.8.0/24
RCA / Database /			Pulivari

ACA.Rhodes, ACA.Naxos, EATA-			Paarma
ACA.Athens,			MSM

Table 3-18: IP addresses of the testbed devices

3.3.2 Hardware / Software equipment

3.3.2.1 Router feature information

Router	IOS	QoS	Interface
C1750	12.2T	Yes	Serial 0, FastEthernet 0
C2500-1	12.1T	No	Serial 0,1; Ethernet 0,1
C2500-2	12.1T	No	Serial 0,1; Ethernet 0,1
C2600	12.1T	Yes	Serial 0/0, 0/1; FastEthernet 0/0, 0/1
C3810	12.1T	No	Serial 0/1; Ethernet 0
C7200	12.2(4)T1	Yes	Serial 4/0, POS 3/0, FastEthernet 2/0
C7500	12.2T	Yes	Serial 0/1/0, 0/2/0, 0/3/0, 0/4/0, ATM
C12000	12.0(19)ST.2	Yes	POS 0/1, ATM 3/0.40

Table 3-19: Router feature information

4 Integration test cases / validation of the Implementation

4.1 Infrastructure test cases

The purpose of the infrastructure test cases is to validate the proper setup of the testbed devices (routers and computers). The test should verify the connectivity between all the end-terminals inside a particular testbed and between the interconnected testbeds. Additionally, the basic parameters (delay, packet loss ratio) of interconnection link between the TPS and TAA are measured.

4.1.1 Basic interconnectivity tests

Activity No.	Basic.interconnectivity.1
Title	Basic infrastructure tests, intra-domain TPS topology
Description	The test verifies the proper setup of DiffServ layer (configuration of routers: topology, addressing, OSPF routing). Test is performed with the use of 'ping' utility.
Preconditions	All routers and terminals are switched on.
Results	The ping confirms that the connectivity is established between all terminals (PC1-PC8, MS, SUN) in the TPS testbed. Result (passed/failed): passed
Notes	

Activity No.	Basic.interconnectivity.2
Title	Basic infrastructure tests, inter-domain TPS-TAA topology
Description	The test verifies the proper setup of interconnection link (tunnel configuration, BGP routing). Test is performed with the use of 'ping' and 'traceroute' utilities.
Preconditions	All routers and terminals in both testbeds are switched on.
Results	The ping confirms that the connectivity is established between terminals in TPS (PC1-PC8, MS, SUN) and terminals in TAA (CM1, CM2, MM1, MM2, MMS). The 'traceroute' program shows, that the path through interconnection networks (TPS-POL34-GEANT-ACONET) is tunnelled and treated as one hop in the testbed topology. Result (passed/failed): passed
Notes	

4.1.2 Advanced interconnectivity tests

Activity No.	Advanced.interconnectivity.1
Title	Tests the impact of parameters of TPS-TAA (direction Warsaw – Vienna) interconnection link on the end-to-end QoS.

Description	The test is performed with the AQUILA measurement tools. The measurement flow (probing) is established between PC2 in TPS and CM2 in TAA. The duration of the test was 1 hour. The measurement flow parameters: constant bit rate 1.12Mbps, packet size 1400B.
Preconditions	The interconnection link is not loaded with any additional traffic.
Results	Packet loss ratio: 0.019 Min delay: 25.58 ms Avg delay: 38.8 ms Max delay: 94.25 ms Mean IPDV: 0.1 ms Max IPDV: 55.51 ms
Notes	In should be noted, that measured packet loss ratio is quite high and this result should be taken into account during the inter-domain network services trials.

Activity No.	Advanced.interconnectivity.2
Title	Tests the impact of parameters of TAA-TPS (direction Vienna – Warsaw) interconnection link on the end-to-end QoS.
Description	The test is performed with the AQUILA measurement tools. The measurement flow (probing) is established between CM2 in TAA and PC2 in TPS. The duration of the test should was 1 hour. The measurement flow parameters: constant bit rate 1.12Mbps, packet size 1400B.
Preconditions	The interconnection link is not loaded with any additional traffic.
Results	Packet loss ratio: 0.0001 Min delay: 26.8 ms Avg delay: 37.36 ms Max delay: 63.49 ms Mean IPDV: 0.08 ms Max IPDV: 26.1 ms
Notes	In should be noted, that measured packet loss ratio is quite high and this result should be taken into account during the inter-domain network services trials.

4.2 Validation of implementation of Admission Control algorithms

The aim of the test if to verify, that the implementation of AC algorithms conforms to the specification in D1302. The test cases cover the new algorithms: MBAC for TCL1 and TCL2, two new algorithms specified for TCL3, as well as the concept of sharing resources on the access link by all classes (so-called Joint AC).

During the validation tests, parameter values for the AC algorithms should be configured according to Table 4-1.

PRA1	rho1Egress: 0.52 rho1Ingress: 0.52
MBAC1	k: 10 mAgeingWindow: 1 rho1Egress: 0.52 rho1Ingress: 0.52 tAgeing: 1 tAgeingConstant: 1 tSample: 2
EBA2	pLoss: 0.0001
MBAC2	k: 10 mAgeingWindow: 1 pLoss: 0.0001 tAgeing: 1 tAgeingConstant: 1 tSample: 10
PRA31	bufferSpace: 25 packetDropProb: 0.1 rho3Egress: 0.7 rho3Ingress: 0.7 rttInterDom: 310ms rttIntraDom: 310.ms
PRA32	lin: 10000000 bufferSpace: 300000 [bits] rttAvgInterDom: 0 rttAvgIntraDom: 0 rttMinInterDom: 100ms rttMinIntraDom: 100ms
EBA4	bufferSpace: 12500 [Bytes] rho4Egress: 1 rho4Ingress: 1 w4: 0.01 wLow: 0.03
Weight setting	TCL1: priority (0.66) TCL2: 0.3 TCL3: 0.01 TCL4: 0.01 TCL5: 0.01

Table 4-1 Parameter values for AC validation tests

4.2.1 Validation of MBAC implementation for TCL1 and TCL2

Test cases specified below correspond to validation of MBAC implementation for classes TCL1 and TCL2. MBAC takes into account actual traffic carried by the network, so in these tests measurement flows should be generated, corresponding to each accepted reservation request. The parameters of generated traffic should be in accordance with traffic descriptors submitted in the reservation requests. The time between the start of traffic generation and the last reservation attempt should be long enough to allow for stabilization of the measured mean rate estimation.

Activity No.	Ac.validation.mbac.case.1			
Title	Validation of MBAC implementation for TCL1			
Description				
Preconditions	MBAC1 algorithm should be configured for TCL1. Components aca_br1tps and aca_er3tps should be started and running without error. Reservation requests are submitted to PCBR service, for flows between PC1 and PC5. After each accepted reservation, new CBR flow (with rate equal to the declared PR) should be generated between PC1 and PC5.			
Results	Flow parameters	Expected no. of flows	Number of flows admitted	Result (pass/failed)
	PR=500	10	10	**passed**
	PR=300	17	17	**passed**
Notes				

Activity No.	Ac.validation.mbac.case.2			
Title	Validation of MBAC implementation for TCL2			
Description				
Preconditions	MBAC2 algorithm should be configured for TCL2. Components aca_br1tps and aca_er3tps should be started and running without error. Reservation requests are submitted to PVBR service, for flows between PC1 and PC5. After each accepted reservation, new ON-OFF flow (with peak rate equal to the declared PR and mean rate equal to the declared SR) should be generated between PC1 and PC5.			
Results	Flow parameters	Expected no. of flows	Number of flows admitted	Result (pass/fail)
	PR=940kbps SR=135kbps	11	11	**passed**
	PR=1900kbps SR=320kbps	2	2	**passed**
Notes				

4.2.2 Validation of new AC algorithms for TCL3

Test cases specified below correspond to validation of implementation of two new algorithms for TCL3. The algorithms are denoted as PRA31 (specified in D1302) and PRA32 (specified in 13wut2005c.doc).

Activity No.	Ac.validation.tcl3.case.1			
Title	Validation of implementation of algorithm PRA31.			
Description				
Preconditions	PRA31 algorithm should be configured for TCL3. Components aca_br1tps and aca_er3tps should be started and running without error. Reservation requests are submitted to PMM service, for flows between PC1 and PC5, with given requested rates (RR).			
Results	Requested rates	Expected no. of flows	Number of flows admitted.	Result (pass/fail)
	250kbps	9	9	**passed**
	250kbps 350kbps	6 2	6 2	**passed**
Notes				

Activity No.	Ac.validation.tcl3.case.2			
Title	Validation of implementation of algorithm PRA32.			
Description				
Preconditions	PRA32 algorithm should be configured for TCL3. Components aca_br1tps and aca_er3tps should be started and running without error. Reservation requests are submitted to PMM service, for flows between PC1 and PC5, with given requested rates (RR).			
Results	Requested rates	Expected no. of flows	Number of flows admitted	Result (pass/fail)
	288.022kbps	3	3	**passed**
	455.593kbps	3	3	**passed**
Notes				

4.2.3 Validation of Joint AC implementation on high speed links

The test cases specified below correspond to validation of Joint AC. In this test, requests are submitted to all network services, which share the resources of the access link.

Activity No.	Ac.validation.joint.case.1
Title	Validation of Joint AC implementation, case #1
Description	
Preconditions	Components aca_br1tps and aca_er3tps should be started and running

	without error. PRA1 algorithm should be configured for TCL1, EBA2 for TCL2, PRA31 for TCL3 and EBA4 for TCL4. Values of submitted flow parameters should be specified as follows: PCBR: PR=400000bps, BSP=2000B PVBR: PR=270000bps, BSP=2000, SR=160000bps, BSS=100000B PMM: RR=250000bps PMC: PR=100000bps, BSP=2000B, SR=50000bps, BSS=10000B Flows should be established between PC1 and PC5.					
Results		PCBR	PVBR	PMM	PMC	Result (pass/failed)
	No. of flows (expected/admitted)	13 / 13	0 / 0	0/ 0	0 / 0	**passed**
	No. of flows (expected/admitted)	5 / 5	22 / 22	0 / 0	3 / 3	**passed**
	No. of flows (expected/admitted)	5 / 5	10 / 10	3 / 3	12 / 12	**passed**
	No. of flows (expected/admitted)	0 / 0	18 / 18	5 / 5	20 / 20	**passed**
Notes						

4.2.4 Validation of Inter-domain AC implementation

This test validates the implementation of inter-domain admission control algorithms. Only two network services: PVBR and PMM are available in the inter-domain scope. The peak rate allocation scheme is used to perform AC on the inter-domain link.

Activity No.	Ac.validation.idom.case.1			
Title	Validation of Joint AC implementation, case #1			
Description				
Preconditions	PRA1 algorithm should be configured for TCL1 and PRA31 for TCL3. Flows should be established between PC1 and CM1. Components aca_br1tps, aca_er1taa, bgrp_br1tps, bgrp_br1taa should be started and running without error. Bandwidth configured in SLA for TCL1 is 500kbps. TCL1 flow parameters are: PR=100kbps, BSP=2000B Bandwidth configured in SLA for TCL3 is 1Mbps. TCL3 flow parameters are: RR=300kbps (resulting SR=87.33kbps)			
Results		PCBR	PMM	Result (pass/failed)
	No. of flows (expected /admitted)	5 / 5	3 / 3	**passed**
Notes				

4.2.5 Validation of secondary access link admission control implementation

4.2.5.1 Test environment

The test network consists of five Cisco routers with the topology shown in the picture. There are two secondary access links and one primary access link. The primary access link is the bottleneck.

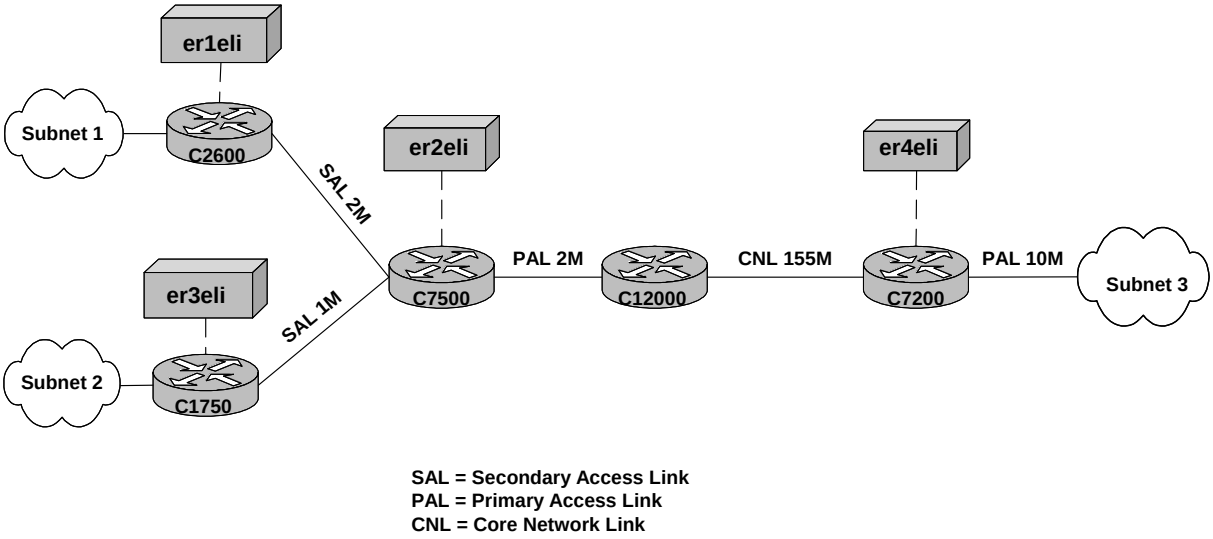


Figure 4-1: Test network

4.2.5.2 Declaration Based Admission Control – results

The validation of admission control for the secondary access links was done by calculating the expected number of reservations and making reservations in the testbed. The measured values were compared with the theoretical values.

The results are presented in the tables, where the first two columns present the traffic class and the guaranteed bandwidth for that traffic class. The next column gives the reservation size for one reservation. The following two columns contain the theoretical and measured number of reservations allowed by the admission control. The last column in the tables gives the equation defined in D1302, which fails when trying to do additional reservations.

Test scenarios from 1 to 4 are performed on the 1Mbit/s link between er3eli and er2eli.

Parameters used for the measurements are presented in Table 3.1. The WFQ weights are scaled so that the sum of the weights is one.

Traffic class	Rho	PQ weight	WFQ weight	P_loss	RTT
TCL1	1	0,1			
TCL2	1		0,81	1,00E-04	
TCL3	0,7		0,03		0,31
TCL4	1		0,03		

Table 4-2: Parameters used during measurements

4.2.5.3 Scenario 1: Reservations with different guaranteed bandwidth values

In the first test guaranteed bandwidth is set to zero for all traffic classes and maximum reservations are made to classes, one class at the time. The results are presented in the following table.

Traffic Class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL 1	0	10 k	102	102	Q1
TCL 2	0	50k / 25k	18	18	Q1
TCL 3	0	150k	1	1	Q2
TCL 4	0	10k / 5k	34	34	Q3

Table 4-3: Reservations without guaranteed bandwidth

Calculated and measured numbers of the reservations are equal for all traffic classes.

In the second test guaranteed bandwidth is set to test specification values for each traffic class and maximum reservations are made to classes one class at the time The results are presented in Table 3.4.

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL 1	100k	10k	32	32	P2
TCL 2	250k	20k / 10k	41	41	P1
TCL 3	150k	150k	1	1	P2
TCL 4	100k	10k / 5k	22	22	P3

Table 4-4: Reservations with guaranteed bandwidth

Also with guaranteed bandwidth values the expected and measured numbers of the reservations are equal.

4.2.5.4 Scenario 2: Maximum reservation for first traffic class

In this scenario the maximum number of flows is reserved in one traffic class and then reservations are made in other traffic classes to check that the guaranteed bandwidth exists. The test is repeated several times with different order of traffic classes.

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL 1	100k	10k	32	32	P2
TCL2	250k	20k / 10k	25	25	P2
TCL3	150k	150k	0	0	P2
TCL4	100k	10k / 5k	15	15	P3
Link utilisation	76,37 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL 2	250k	20k / 10k	41	41	P1
TCL1	100k	10k	11	11	Q1
TCL3	150k	150k	0	0	Q2
TCL4	100k	20k / 10k	8	8	P3
Link utilisation	72,95 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL 3	150k	150k	1	1	Q3
TCL1	100k	10k	13	13	P2
TCL2	250k	20k / 10k	***25***	***28***	P2
TCL4	100k	10k / 5k	***21***	***22***	P3
Link utilisation	88,09 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL 4	100k	10k / 5k	22	22	P3
TCL1	100k	10k	***12***	***11***	P3
TCL2	250k	20k / 10k	26	26	P3
TCL3	150k	150k	1	1	Q2
Link utilisation	83,69 %				

Table 4-5: Maximum reservation to first traffic class

Comparing the theoretical and measured values we can conclude that in most cases the admission control works exactly as specified. However in some cases small deviation from the calculated values was noticed. These deviating values are highlighted in the tables.

The results show that the three last classes will get at least the guaranteed bandwidth. In the case of TCL3 the reservation of 150kbit/s requires more bandwidth than 150kbit/s so no reservation is possible.

The link utilisation is always over 73% of the link capacity, which is reasonable in the case where no TCL3 reservations were possible and small amount of bandwidth is reserved for best effort traffic.

4.2.5.5 Scenario 3: Existence of guaranteed bandwidth

In this scenario maximum bandwidth allowed by AC was reserved for the three traffic classes. After this it was checked that the fourth traffic class did get its guaranteed bandwidth.

The first reservation was always made to traffic class three because only that way it can be guaranteed that one reservation can be made in that class. The minimum reservation size for TCL3 is 150kbit/s according to formulas in D1302 chapter 4.4.2.3.

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL3	150k	150k	1	1	Q2
TCL2	250k	40k / 20k	14	14	P2
TCL4	100k	10k / 5k	21	21	P3
TCL1	100k	10k	10	10	Q2
Link utilisation	84,08 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL3	150k	150k	1	1	Q2
TCL4	100k	10k / 5k	22	22	P3
TCL2	250k	40k / 20k	13	13	P3
TCL1	100k	10k	***11***	***10***	Q3
Link utilisation	82,71 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL3	150k	150k	1	1	Q2
TCL1	100k	10k	13	13	P2
TCL4	100k	10k / 5k	21	21	P3
TCL2	250k	40k / 20k	12	12	Q2
Link utilisation	82,32 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL3	150k	150k	1	1	Q2
TCL4	100k	10k / 5k	22	22	P3
TCL1	100k	10k	***12***	***11***	P3
TCL2	250k	40k / 20k	12	12	Q3
Link utilisation	81,35 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL3	150k	150k	1	1	Q2
TCL1	100k	10k	13	13	P2
TCL2	250k	40k / 20k	12	12	Q2
TCL4	100k	10k / 5k	21	21	Q3
Link utilisation	82,32 %				

Traffic class	Guaranteed BW	Reservation size	Calculated #	Measured #	Failure reason
TCL3	150k	150k	1	1	Q2
TCL2	250k	40k / 20k	14	14	P2
TCL1	100k	10k	10	10	Q2
TCL4	100k	10k / 5k	21	21	Q3
Link utilisation	84,08 %				

Table 4-6: Existence of guaranteed bandwidth

Comparing the theoretical and measured values we can conclude that in most cases the admission control works exactly as specified. However in some cases small deviation from the calculated values was noticed. These deviating values are highlighted in the tables.

The results show that the three last classes will get at least the guaranteed bandwidth. The link utilisation is slightly higher than in previous case because TCL3 is also using its guaranteed bandwidth.

4.2.5.6 Scenario 4: Minimum reservations

In this scenario minimum reservation was done in three traffic classes and then the unallocated bandwidth is reserved to one traffic class.

Traffic class	Guaranteed BW	Reservation size	Reservation # (calculated)	Failure reason
TCL 3	220k	150k	1	
TCL 2	250k	50k / 25k	5	
TCL 4	100k	10k / 5k	10	
TCL 1	100k	10k	11 (11)	P2
Link utilisation	59,57 %			

Traffic class	Guaranteed BW	Reservation size	Reservation # (calculated)	Failure reason
TCL 3	220k	150k	1	
TCL 1	100k	10k	10	
TCL 4	100k	10k / 5k	10	
TCL 2	250k	50k / 25k	10 (10)	Q2
Link utilisation	83,01 %			

Traffic class	Guaranteed BW	Reservation size	Reservation # (calculated)	Failure reason
TCL 3	220k	150k	1	
TCL 1	100k	10k	10	
TCL 2	250k	50k / 25k	5	
TCL 4	100k	10k / 5k	22 (22)	P3
Link utilisation	70,31 %			

Traffic class	Guaranteed BW	Reservation size	Reservation # (calculated)	Failure reason
TCL 1	100k	10k	10	
TCL 2	250k	50k / 25k	5	
TCL 4	100k	10k / 5k	10	
TCL 3	220k	150k	1 (1)	Q2
Link utilisation	58,59 %			

Table 4-7: Minimum reservations for three traffic classes

In this scenario the theoretical number for the reservations was equal to the measured values. The link utilisation was in same cases little lower because admission control is little conservative.

4.2.5.7 Scenario 5: Reservations over two links

In this scenario TCL1 reservations (10kbit/s) were done. First the maximum number of reservations was made from er3eli to er4eli. Then the maximum number of reservations was made

from er1eli to er4eli. Guaranteed bandwidth for TCL1 was 100kbit/s for er3eli and 200kbit/s for er1eli. Guaranteed bandwidth for er2eli was 200kbit/s. The results are summarised in the following table.

Source	Destination	Min link BW	Traffic class	Guarant. BW	Reserv. # (cal/meas)	Failure reason
er3eli	er4eli	1 Mbit/s	TCL1	100k	32 / 32	P2 / aca3eli
er1eli	er4eli	2 Mbit/s	TCL1	100k	32 / 32	P2 / aca2eli

Table 4-8: Maximum number of reservations from two sources with guaranteed BW

The results show that two-step admission control works as specified. The primary access link (er2eli) has no room for additional reservations although the secondary link (er1eli) would have capacity for additional reservation.

In this scenario TCL1 reservations (25kbit/s) were done. First the maximum number of reservations was made from er3eli to er4eli. Then the maximum number of reservations was made from er1eli to er4eli. All guaranteed bandwidth values were set to zero. The results are summarised in the following table.

Source	Destination	Min link BW	Traffic class	Guarant. BW	Reserv. # (cal/meas)	Failure reason
er3eli	er4eli	1 Mbit/s	TCL1	0	40 / 40	Q1 / aca3eli
er1eli	er4eli	2 Mbit/s	TCL1	0	41 / 41	Q1 / aca2eli

Table 4-9: Maximum number of reservations from two sources without guaranteed BW

The results show that two-step admission control works as specified. The primary access link (er2eli) has no room for additional reservations although the secondary link (er1eli) would have capacity for additional reservation.

4.3 Validation of the resource pool mechanisms

4.3.1 Reservations over 10 Mbit links

In order to check the functionality of the resource pools, i.e. to test the basic functionality of the RP (how it shares resources among the RPs, how it manages requests generated by them), an appropriate topology is necessary. Figure 4-2 shows the resource pool elements created in order to check and evaluate their operation as well as the implemented resource management algorithm.

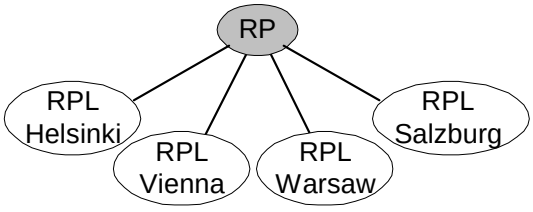


Figure 4-2: TAA RP structure for the resource pool mechanism validation

The goal is to test RP performance and the signalling load: efficiency of resource usage and frequency of resource shifts. During this test, reservations are created form host CM2 (10.0.8.1) to host CMS (10.0.1.1). The access links to the edge router are 100 Mbit links. The core network uses 10 Mbit links.

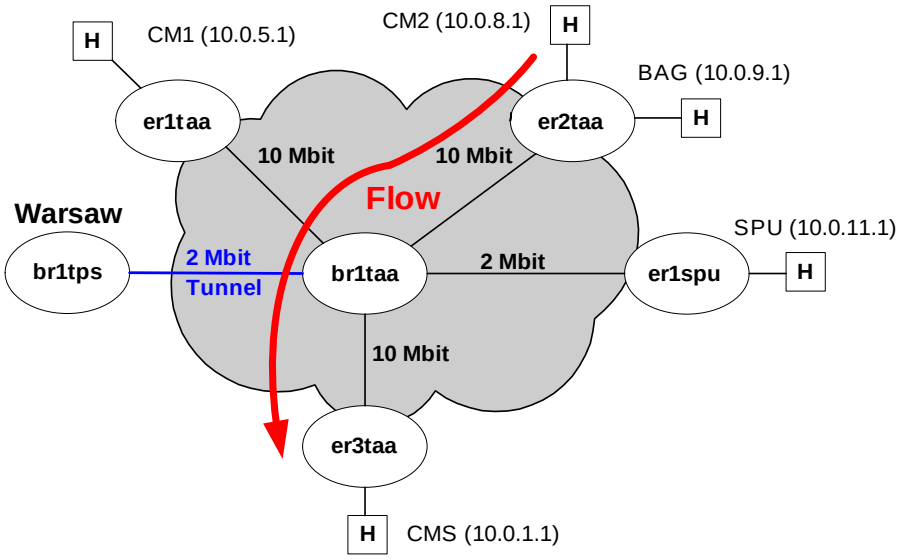


Figure 4-3: Configuration including 10 Mbit links

For the test, the following start-up configuration is used. Please note that in both, ingress and egress, the same start-up values are used.

	Root Pool		
TCL	Ingress / Egress	RS max [kbit/s]	RS Tot [kbit/s]
TCL1	Ingress	1900	1900
	Egress	1900	1900
TCL2	Ingress	2850	2850
	Egress	2850	2850
TCL3	Ingress	3800	3800
	Egress	3800	3800
TCL4	Ingress	950	950

	Egress	950	950

Table 4-10: RP initial configuration

The following Table 4-11 shows the parameter settings for the RP algorithm.

Parameter	Amax	Amin	WL	BlockSize	Counter	ReleasePeriod
Value	5	1	0,9	100kbps	10	5 min

Table 4-11: Parameter settings for the RP algorithm

The following separation of resources per traffic class for a 10 Mbit/s link is defined as:

TCL1	10%	500 kb/s	20 voice calls
TCL2	15%	750 kb/s	3 * 250kb/s video
TCL3	20%	1 Mb/s	2 * 500kb/s download
TCL4	5%	250 kb/s	50 sessions

Table 4-12: Reference load distribution among TCLs on a 10 Mbit link

4.3.1.1 Results for TCL 1

Twenty reservations were possible. Consequently this trial was successful.

4.3.1.2 Results for TCL 2

No problems occurred making TCL2 reservations.

4.3.1.3 Results for TCL 3

This trial also was successful: the third reservation was rejected. Worth mentioning here is that the bandwidth has be entered as sustainable rate and not as peak rate.

4.3.1.4 Results for TCL 4

The first trial was not successful, because P2A as connection type was chosen. After having solved this problem the system stopped to make reservation after the 31st one. With the help of the trial support people it was recognised that the minimum sustainable rate for TCL 4 is 8kbit/s. Considering this, the result of 31 possible reservations was correct.

4.4 End-to-end test cases

4.4.1 Main end-to-end tests

4.4.1.1 RCA

Activity No.	end-to-end.rca.1
Title	Allocation of a resource reservation
Description	Establishment of a reservation
Preconditions	RCA ready to receive requests. A user is logged in.
Postconditions	The RCA log file should contain a message indicating the bandwidth allocated to the successful reservation
Notes	The behaviour of the rcrule algorithm for the resource pools can be also tested with this test
Status	Done

Activity No.	end-to-end.rca.2
Title	De-allocation of a resource reservation
Description	Release of resource reservation
Preconditions	An appropriate request is already accepted
Postconditions	Resource is de-allocated. The log file indicates that the reservation has been released
Notes	The behaviour of the rcrule algorithm for the resource pools can be also tested with this test
Status	Done

Activity No.	end-to-end.rca.3
Title	Test the thresholds of each traffic class
Description	Via the portal, resources are requested from each traffic class and consume the available resources up to the limit. So each request of new resources to the different shares initiates a RCA request to the higher resource pool level.
Preconditions	Requests are done so that all resources at this ACA are consumed. Recourses are available at the higher RCA level.
Postconditions	The log file indicates the propagation of the request to the higher level of resource pools.
Notes	The behaviour of the rcrule algorithm for the resource pools can be also tested with this test
Status	Done

Activity No.	end-to-end.rca.4
Title	All resources are consumed up to the highest level
Description	All recourses are consumed up to the highest level of the RCA. Each new request has to be rejected
Preconditions	No resources available.

Postconditions	All the reservation requests are rejected
Notes	
Status	Done

4.4.1.2 ACA

The ACA test cases, described in the following, need three ACA only in order to test the third party model otherwise only two ACAs are required. For the inter-domain test case scenario two domains are needed.

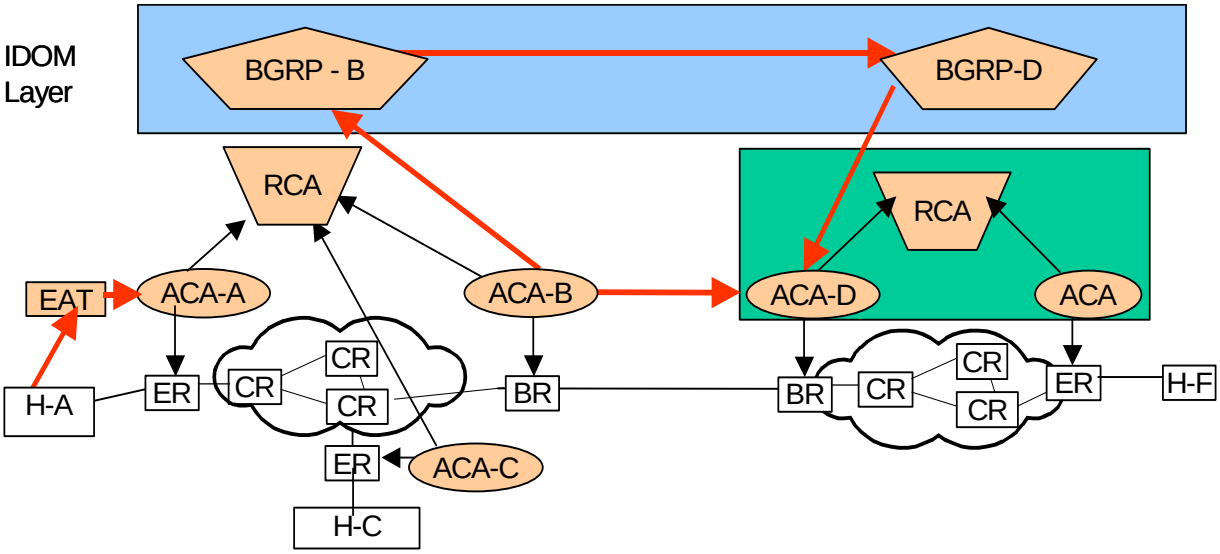


Figure 4-4: ACA Inter-domain test reference scenario

For all test cases the right configuration of the ACA settings is a prerequisite and also the router configuration has to fit. Each router device is controlled by one ACA and therefore for each ACA a separated configuration set of the router device has to be created in the QMTool under the router entry. The name of the router entry has to be the same name, as the ACA will get during the start up phase by a start up script. Only the same naming gives the ACA the change to find the right configuration settings

4.4.1.2.1 Login

Activity No.	end-to-end.aca.1
Title	Login

Description	The user logs in with the stored account name and the right password
Preconditions	All ERs (edge routers) have been configured correctly; the ACA responsible for these routers are also correctly configured and started up. The EAT has been started and the EAT is logically connected to ACA-A (see the EAT settings). A user has been configured with a right account, password and the needed services
Postconditions	The log file of the ACA-A must show that the login process has been successful
Notes	
Status	Done

4.4.1.2.2 Logout

Activity No.	end-to-end.aca.2
Title	Logout
Description	The “logged in” user logs out
Preconditions	A user has been logged in
Postconditions	The log file of the ACA-A must show that the logout process has been successful
Notes	
Status	Done

4.4.1.2.3 Request Enabled Reservation(s)

Activity No.	end-to-end.aca.3
Title	Request Enabled Reservation(s)
Description	Make a reservation from host-A to host-C
Preconditions	ACA-A and ACA-C are up and running
Postconditions	Analyse the log files of both ACAs, which are involved. ACA-A has to calculate the needed bandwidth amount and requests this amount from the RCA. After an acknowledgement the ACA-A forwards the request to the egress side that has to be found before. At the egress side a bandwidth calculation has to be done also. The amount has to be re-

	quested from the RCA and the acknowledgement has to be transferred back to ACA-A. The ACA-A sets the configuration in the router device and acknowledges the correct installation of this reservation request back to the EAT
Notes	
Status	Done

4.4.1.2.4 Request Disabled Reservation(s)

Activity No.	end-to-end.aca.4
Title	Request Disabled Reservation(s)
Description	Make a reservation from host-A to host-c
Preconditions	ACA-A and ACA-C are up and running
Postconditions	Analyse the log files of both ACAs. The same results as described in the chapter 4.4.1.2.3 but without the configuration of the router device. Only bandwidth resources are reserved
Notes	
Status	Done

4.4.1.2.5 Release Reservation

Activity No.	end-to-end.aca.5
Title	Release Reservation
Description	Release the requested “enabled” reservation
Preconditions	ACA-A and ACA-C are up and running
Postconditions	Analyse the log files of both ACAs. the configuration of the routers has to be deleted and a recalculation of the needed bandwidth amount has to be trigged; check if the bandwidth amount is at the same level before this released reservation request has been installed
Notes	
Status	Done

4.4.1.2.6 Bandwidth Calculation Check

Activity No.	end-to-end.aca.6
Title	Bandwidth Calculation Check
Description	Start an “enabled” reservation request into each class (tcl1, tcl2, tcl3, tcl4) on condition that no other reservation request is accepted in another traffic class. OR: Start an “enabled” reservation request into each class (tcl1, tcl2, tcl3, tcl4) on condition that other reservation requests are accepted in another traffic class.
Preconditions	ACA-A and ACA-C are up and running; Calculate by hand with the specified formulas the bandwidth amount that is needed for these reservation request parameters that should be sent in the especial test case
Postconditions	Analyse the log files of both ACAs and check if the calculated bandwidth amount is identical with the pre-calculated values
Notes	
Status	Done

4.4.1.2.7 Small Bandwidth Link Access

Activity No.	end-to-end.aca.7
Title	Small Bandwidth Link Access
Description	Make a reservation from host-A to host-C
Preconditions	A test scenario with cascaded ACAs is needed; therefore a router configuration has to be set that now ACA-A has to forward a reservation request to and new ACA-Y. See the router configuration file
Postconditions	Analyse the log files of both ACAs; an admission control has to be executed at the ACA-A and also at the ACA-X and only if both admission control checks are passed the request is accepted
Notes	
Status	Done

4.4.1.2.8 Inter-domain Reservation Request

Activity No.	end-to-end.aca.8
Title	Inter-domain Reservation Request
Description	Make a reservation from host-A to host-F
Preconditions	ACA-A and ACA-C are up and running
Postconditions	Analyse the log files of both ACAs, which are involved. In the first domain the same should happen as described in chapter 4.4.1.2.3 with the extension that ACA-B has to forward the received reservation request at the egress side to the inter-domain layer especially to the agent BGRP-B and after the acknowledgement of this inter-domain request from the BGRP layer a reservation request has to be done into the last domain of this domain chain in order to initiate also there an intra-domain reservation if a resource control layer exists there
Notes	
Status	Done

4.4.1.3 EAT

The end-to-end test cases regarding the EAT mainly concern the interaction between the different clients using the EAT API and the EAT itself. These clients are:

- the Complex Internet Service Mediazine as sample application,
- the AQUILA QoS Portal, which provides a GUI for the EAT,
- the EAT Script processor, which parses XML files containing reservation scenarios,
- and the applications, which do not directly interact with the EAT but through one of the EAT proxies.

The following main test scenarios can be identified:

4.4.1.3.1 Login and Logout

The login functionality ensures that only valid users can use the RCL's QoS. The login is the basis for an association between customers and reservations.

Login

Activity No.	end-to-end.eat.1
Title	Login
Description	Login via the AQUILA Portal, Mediazine, or the EAT Script with account name and password.
Preconditions	EAT and subscribers correctly configured.
Postconditions	The AQUILA Portal and Mediazine should show the next page, in the case of the portal that is the current reservation GUI. The EAT must print a success message on the console. Also the log file of the EAT must contain such a message.
Notes	
Status	Done

Logout

There are two kinds of logout: with (“hard” logout) and without (“close”) releasing the actual reservations.

Activity No.	end-to-end.eat.2
Title	Logout
Description	Logout via the AQUILA Portal, Mediazine, or the EAT Script.
Preconditions	An end-user must be logged in.
Postconditions	The AQUILA Portal and Mediazine should go back to the login page. The EAT must print a success message on the console. Also the log file of the EAT must contain such a message. In the case of a hard logout, there must also be some messages concerning released reservations, if some have exist.
Notes	
Status	Done

4.4.1.3.2 Reservation request on advanced level

The advanced reservation request is only for users which are familiar with the AQUILA architecture. A lot of technical parameters have to be entered in this mode.

Activity No.	end-to-end.eat.3
Title	Reservation request on advanced level
Description	Use the AQUILA Portal's advanced reservation GUI (or the EAT Script) to input the necessary reservation parameters, and then submit the form/request to the EAT.
Preconditions	An end-user must be logged in.
Postconditions	The AQUILA Portal must show a new entry on the current reservation GUI. The EAT must print a success message on the console. Also the log file of the EAT must contain such a message.
Notes	
Status	Done

4.4.1.3.3 Reservation request on regular level

The regular reservation request is the usual way for normal end-users. It is the way Mediazine uses.

Activity No.	end-to-end.eat.4
Title	Reservation request on regular level
Description	Use the AQUILA Portal's advanced reservation GUI (or the EAT Script) to input the necessary reservation parameters, and then submit the form/request to the EAT.
Preconditions	An end-user must be logged in. Application Profiles must correctly be configured.
Postconditions	The AQUILA Portal must show a new entry on the current reservation GUI. The EAT must print a success message on the console. Also the log file of the EAT must contain such a message.

Notes	
Status	Done

4.4.1.3.4 Reservation release

Reservation release can either be done automatically (by using the hard logout) or via the portal, Mediazine, or the script:

Activity No.	end-to-end.eat.5
Title	Reservation release
Description	Use the AQUILA Portal's current reservation GUI (or the script) to mark a reservation unit or group and then release it.
Preconditions	An end-user must be logged in. There must be one or more active reservations.
Postconditions	The reservation must be deleted from the current reservation GUI. The EAT must print a success message on the console. Also the log file of the EAT must contain such a message.
Notes	
Status	Done

4.4.1.3.5 Request and release of reservations by the Proxy

The proxies are used to automatically establish QoS reservations or to retrieve necessary flow information:

- The SIP Proxy can autonomously establish a reservation without using the AQUILA portal.
- The H.323 Proxy detects important flow information for an already (manually) established but still provisional reservation.

Consequently, there are two test scenarios:

SIP Proxy + SIP application:

Activity No.	end-to-end.eat.6
Title	SIP Proxy + SIP application reservation by Proxy

Description	Start a SIP session by using the SIP application (e.g. the SIP User Agent).
Preconditions	An end-user must be logged in. The SIP Proxy must be running and registered at the EAT. The SIP application must be configured to communicate via the SIP Proxy.
Postconditions	The AQUILA Portal must show a new entry on the current reservation GUI (after refresh). The SIP Proxy must show some print outs related to the SIP flow. The EAT must print a success message on the console. Also the log file of the EAT must contain such a message.
Notes	
Status	Done

H.323 Proxy + NetMeeting:

Activity No.	end-to-end.eat.7
Title	H.323 Proxy + NetMeeting reservations by Proxy
Description	Use the AQUILA Portal's advanced or regular reservation GUI to request for a NetMeeting reservation. (On the advanced GUI, choose "H.323 proxy". On the regular GUI choose "NetMeeting".) Start a NetMeeting session.
Preconditions	An end-user must be logged in. The H.323 Proxy must be running and registered at the EAT.
Postconditions	After the first test step, the AQUILA Portal must show a new entry on the current reservation GUI. This reservation is still provisional. After the second test step, the AQUILA Portal must show that the reservation is now active. The H.323 must show some print outs related to the NetMeeting flow. The EAT must print success messages for both steps on the console.

	Also the log file of the EAT must contain such a message.
Notes	
Status	Done

4.4.1.4 BGRP

The BGRP end-to-end test cases described in the following generally require a two-domain scenario. The final test case “Resource request at transit domain” requires three domains in sequence.

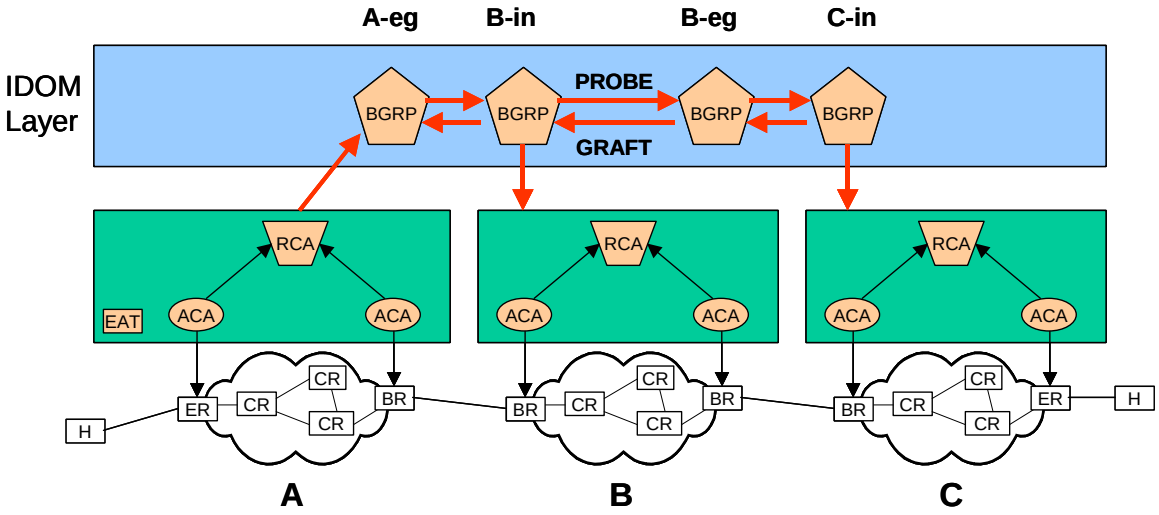


Figure 4-5: Inter-domain test reference scenario

The figure above shows the three-domain scenario and explains the notation used in the following chapter: The domains are named A, B and C. The BGRP agents at the ingress and egress border routers are named A-eg, B-in, B-eg and C-in. In the two-domain-scenarios it is assumed, that domain C is omitted.

4.4.1.4.1 Neighbour detection

Communication between neighboured BGRP agents is established using a neighbour detection mechanism. This test case verifies that two BGRP agents can set-up a communication path using this mechanism.

Activity No.	end-to-end.bgrp.1
Title	BGRP Neighbour detection

Description	Start BGRP agents A-eg and B-in
Preconditions	BGRP agents A-eg and B-in are configured correctly. It is important, that the neighbour configuration of both agents match each other
Postconditions	The log files of both BGRP agents must show an entry saying that the respective neighbour BGRP agent is up.
Notes	
Status	Done

4.4.1.4.2 Set-up along unknown sink tree

During the first reservation set-up, the BGRP agent is not able to identify the sink tree during PROBE processing. The PROBE message will therefore be forwarded to the final destination. The sink tree will be assigned there and communicated back with the GRAFT message.

Activity No.	end-to-end.bgrp.2
Title	Set-up along unknown sink tree
Description	Make a reservation with a source address within domain A and a destination address within domain B.
Preconditions	BGRP agents A-eg and B-in are up and running
Postconditions	Analyse the log files of both BGRP agents. BGRP agent A-eg must say, that it is unable to identify the sink tree during PROBE processing and must forward the PROBE message to BGRP agent B-in. BGRP agent B-in must find out, that this domain is the final destination and send a GRAFT message back. BGRP agent A-eg must receive this GRAFT message and install the sink tree and the associated NLRI in its database.
Notes	
Status	Done

4.4.1.4.3 Release

Release of resources will only have an immediate effect in the very first BGRP agent. Delayed resource release is tested in the next test case.

Activity No.	end-to-end.bgrp.3

Title	BGRP release
Description	Start BGRP agents A-eg and B-in
Preconditions	A reservation is established
Postconditions	The log files of BGRP agent A-eg must show an entry saying that the resources are released. BGRP agent B-in is not immediately affected.
Notes	
Status	Done

4.4.1.4.4 Delayed release

The resource cushions built during reservation release are decremented according to ruled determined by the RcRule configuration. Note, that the retain period is not specified in the RcRule, but rather in the “refresh” parameter of the BgrpAgent configuration.

Activity No.	end-to-end.bgrp.4
Title	Delayed release
Description	Wait at least three times the release period
Preconditions	A reservation was set up and released. The delayed resource release parameters of the RcRule is configured correctly.
Postconditions	The log files of both BGRP agents must show entries corresponding to REFRESH messages, which indicate delayed resource release as specified in the RcRule and BgrpAgent parameters.
Notes	
Status	Done

4.4.1.4.5 Set-up along known tree

When a reservation for an already known sink tree arrives, the BGRP agent can identify the sink tree and try to answer the request in advance.

Activity No.	end-to-end.bgrp.5
Title	Set-up along known tree
Description	Make a new reservation with more resources than the actual size of the resource cushion.

Preconditions	A reservation was set up and released. The delayed resource release has not yet released all resources.
Postconditions	The log files of BGRP agent A-eg must show an entry saying that the sink tree could be identified. The PROBE message must be forwarded to BGRP agent B-in.
Notes	
Status	Done

4.4.1.4.6 Quiet grafting

When a reservation for an already known sink tree arrives and the resource cushion is large enough to satisfy the request, a BGRP agent before the final destination can answer the request and sent a GRAFT.

Activity No.	end-to-end.bgrp.6
Title	Quiet Grafting
Description	Make a new reservation with less resources than the actual size of the resource cushion.
Preconditions	A reservation was set up and released. The delayed resource release has not yet released all resources.
Postconditions	The log files of BGRP agent A-eg must show an entry saying that the sink tree has been identified. The PROBE message is not forwarded to BGRP agent B-in. Instead, the reservation is already granted at A-eg.
Notes	
Status	Done

4.4.1.4.7 Intra-domain request at transit domain

When a transit domain is involved in the reservation, then the BGRP agent at the ingress of that domain is responsible to set-up a intra-domain reservation towards the egress border router.

Activity No.	end-to-end.bgrp.7
Title	Intra-domain request at transit domain
Description	Make a reservation with a source address within domain A and a destination address within domain C. Then release the reservation and wait,

	until all resource cushions are expired.
Preconditions	Three domains are up and running. All BGRP agents have recognised their respective neighbours.
Postconditions	The log files of BGRP agent B-in must show an entry saying that appropriate resources have been requested from the DomainResourceManager. The corresponding ACA must have an entry in its log file showing this reservation set-up. Later, the modifications and finally release of this intra-domain reservation must be visible in the log file.
Notes	
Status	Done

4.4.1.5 QMTool

4.4.1.5.1 Configuration

Activity No.	end-to-end.qmtool.1
Title	QMTool configuration management
Description	This test case verifies that a component can be re-configured. Therefore, if we change the configuration information of a component and update the database with it, we need to verify the success of this operation. View the configuration of the component and perform changes on it. Then, save and store the configuration information at the database as described in 5.4.1.2. Close the XMLOperator graphical interface. View again the configuration of the component.
Preconditions	The component should already have configuration information stored in that LDAP database
Postconditions	You should be able to view the updated configuration information.
Notes	
Status	Done

4.4.1.5.2 Failure Detection

Activity No.	end-to-end.qmtool.2
Title	Fault management
Description	The test verifies that if the failure detection has been enabled for a component, and the component stops running, its colour will turn to red. Start the failure detection mechanism for this component. Stop the execution of the component, i.e. of the BGRP Agent.
Preconditions	The component we are testing, for example a BGRP Agent, must have been started.
Postconditions	After some seconds, the color of the component should be red.
Notes	
Status	Done

4.4.1.6 Measurements

Preconditions of the following testcases are that all components of DMA are installed and operational according the instructions of chapter 5.3.

4.4.1.6.1 GUI & Database

For the GUI there doesn't exist standalone end-to-end tests. But because it is of cause a central point of the DMA and is used for the end-to-end test, it has to be operational. Therefore here are few test cases:

- The first test is the connection to the GUI via the browser. Fill in the IP-address of the server into the address field of your browser. If the connection works properly a login window is prompted.

- If this don't work properly, check the right address, check the right directory on the server, check if the apache web server works, check if you use unknown proxies in your browser. Ping will show if you can connect to the server via the net on the network level.

- If you fill in the login window with the pre-configured user “aquila” and password “aquila” you will see the navigation bar on the left side and the main window on the right side.

- Check if the graphics in the main window appear. Check every link in the left navigation bar. Every side must appear without an browser error. In case of errors check if the files are in the right directory and in the subdirectories and check the permissions of the files (readable for the server) .

- Try to start the server clock. Javascript is used and must be enabled. Stylesheets are necessary for an optimum view of the colors.

- Try to generate a new user (administration) to check if the connection to the database works properly. If a problem appears check if the user and password in the php files are set correctly.

- Configure the master, some measurement agents, and a few tests with the T-Nova Toolset.

- Try to configure a measurement for the T-Nova toolset. Choose “New test” and “new flow” with a preconfigured traffic. Start the flow and check if the start/stop field is set correct. If the DTA master is already working you will get feedback in the GUI after a few minutes. Check if the flowstate has changed in the start/stop menu. If you stop this flow, depending on the aggregation time, you configured in the flow menu, you will get some results. You can check the results in the result menu. Choose the test you have configured, choose the flow you have configured an try to see the results in graphics.

- If the graphics generate problems check if “gnuplot” is installed correct in the right directory for the flow monitoring.

4.4.1.6.2 CM Toolset (SPU)

Activity No.	end-to.end.measurement.cmt.1
Title	Login of the CMCaller via the EATScript
Description	When restarting the CMCaller and everything works ok, after some time (up to a minute) the line “Login ok!” will appear.
Preconditions	RCL must be up and running, All parts of the DMA must be installed.
Postconditions	CMCaller is running and has logged in to the RCL.
Notes	If the login fails, check the parameter values in the DMA configuration file “/etc/dma.rc” on the measurement server.
Status	Done

Activity No.	end-to-end.measurement.cmt.2
Title	Resource reservation / release using the EAT script
Description	When a flow with a reservation is configured, a reservation request is

	done before the flow starts and a reservation release is done when the flow is ready. For testing this, a flow with a duration of e.g. 5 minutes has to be configured. As long as the flow is running, the reservation has to be active.
Preconditions	Login from the CMCaller to the EAT was successful (see Activity end-to-end.measurement.cmt.1)
Postconditions	Flow has been started, no error message “reservation setup failed” in flow results (can be checked with the GUI). As long as the flow is running, the reservation can be viewed via the AQUILA portal (For the login the values for EAT_username and EAT_password from “/etc/dma.rc” have to be used)
Notes	If the reservation fails (due to misconfiguration or a lack of resources), it depends on the selected option, whether the flow will start or not.
Status	Done

Activity No.	end-to-end.measurement.cmt.3
Title	CMCaller operation
Description	A new flow has to be configured within the GUI. A test with the test-tool ‘SPU’ has to be selected for this flow. The starttime of the flow has to be within the past ten minutes. When the configuration of the flow has been confirmed with “submit”, the first bit of the flow state will change to ‘1’ (flow prepared). If the daemons are already running on the sending and receiving host, the cmcaller will contact the daemons and the flow state will further change to ‘receiver started’ and ‘sender started’. If the sending/receiving daemons are not running, the flow state will indicate an error.
Preconditions	GUI, Database & cmcaller running
Postconditions	Flow state changed to “flow prepared”
Notes	An error in the flow state does not imply improper operation of the caller, but can indicate, that the measurement daemons are not reachable.
Status	Done

Activity No.	end-to-end.measurement.cmt.4
Title	CMDaemon start-up
Description	The CMDaemon contacts the measurement server (CMCaller) and announces his operation.
Preconditions	GUI, database & CMCaller running, IP address of CMCaller host must be configured in the cmdaemon start-up script.
Postconditions	Measurement client is listed in the hop list on the GUI.
Notes	To follow the correct start-up, look at the messages on the terminal as well as in the log-file (/var/log/cmdaemon.log)
Status	Done

Activity No.	end-to-end.measurement.cmt.5
Title	Flow generation
Description	Same procedure as in end-to-end.measurement.cmt.3, after the end time of the configured flow has been reached or the specified number of packets have been sent, the flow state will change to 'flow ready'. Results are available and can be analysed with the GUI.
Preconditions	GUI, Database & cmcaller running
Postconditions	Flow state changed to "flow prepared"
Notes	An error in the flow state does not imply improper operation of the caller, but can indicate, that the measurement daemons are not reachable.
Status	Done

4.4.1.6.3 T-Nova Toolset (DTA)

Activity No.	end-to-end.measurement.tnova.1
Title	Integration of measurement clients into the system
Description	The configuration GUI is used for the integration of measurement clients into the system.
Preconditions	DMA are installed and operational
Postconditions	Measurement clients are known to the system and measurement tests/flows can be defined by the configuration GUI.
Notes	To create new flows it is necessary, that tests, traffic and hops are already available in the database.
Status	Done

Activity No.	end-to-end.measurement.tnova.2
Title	Distribution of flow information
Description	The master reads flow-information from the database and distributes it to the according measurement agents. Before flow-specific information is sent to the measurement agents, the state of the involved agents is queried (if is it up and Synchronisation is ok). After that, the master sends the flow-configuration data to the receiving agent and then to the sending agent.
Preconditions	end-to-end.measurement.tnova.1 and at least one flow is configured in the database and is to be started.
Postconditions	The involved measurement agents have the necessary information to start the specified flow. If everything happened successful, the flow will be started, else an according error message is stored in the event-log of the database.
Notes	
Status	Done

Activity No.	end-to-end.measurement.tnova.3
Title	Flow initialisation and start-up
Description	Measurement agents receive flow configuration data from the master and start with sending/receiving measurement packets. The measurement clients on the receiving sides of the measurement flows send the measurement results periodically to the master. The measurement results of the running flow(s) can be monitored e.g. with the flow monitor.
Preconditions	end-to-end.measurement.tnova.1+2
Postconditions	Running measurement flow(s). Measurement results are stored in the database.
Notes	
Status	Done

Activity No.	end-to-end.measurement.tnova.4
Title	Stop a running measurement flow
Description	The master detects that a measurement flows has to be stopped. The master sends the request to the sender agent and after the sender is stopped the request is send to the receiving measurement agent.
Preconditions	Measurement flow running end-to-end.measurement.tnova.3
Postconditions	Flow- and error state of the measurement flow is updated in the database
Notes	
Status	Done

Activity No.	end-to-end-measurement.tnova.5
Title	Resource reservation using the EAT script
Description	Configure a flow and a reservation for the used resources with the Configuration GUI (CfgGUI) Start the flow and verify with CfgGui that both sides of the flow are started Login to the AQUILA portal (Current reservations) and verify that the resource reservation has been made. (For the login the values for EAT_user and EAT_pw from the master configuration file “master.rc” in the directory /sbin on the masterstation has to be used)
Preconditions	See above, and at least two measurement agents are configured in the database, assigned to the DTA master and operational. The configuration and start / stop of a measurement flow has been tested.
Postconditions	The reservation part of the interworking between the T-Nova Toolset and the EAT is functional if the AQUILA portal (in Current reservations) shows that the reservation has been made successfully.

Notes	If the reservation fails, check the parameter values in the master configuration file “master.rc” in the directory /sbin on the masterstation
Status	Done

Activity No.	end-to-end.measurement.tnova.6
Title	Release of the reserved resources using the EAT script
Description	• Configure, start a flow and reserve the appropiate resources as described in end-to-end.measurement.tnova.5, verify that the flow is started and the resources are reserved (AQUILA portal). • Stop the flow • Login to the AQUILA portal (Current reservations) and verify that no resource are reserved furthermore.
Preconditions	end-to-end.measurement.tnova.5
Postconditions	The release part of interworking between the T-Nova Toolset and the EAT is ok.
Notes	If the release fails, check the parameter values in the master configuration file “master.rc” in the directory /sbin on the masterstation
Status	Done

4.4.1.6.4 Router QoS Monitoring Tool (ELI)

Activity No.	end-to-end.measurement.router.1
Title	Successful login to EAT at startup
Description	The Router monitoring tool uses EAT to retrieve statistics on a measurement flow. The tool needs to log in to EAT in order to be access the results.
Preconditions	EAT has been started
Postconditions	EAT accepts the login
Notes	
Status	Done

Activity No.	end-to-end.measurement.router.2
Title	Successful starting of monitoring
Description	The Router monitoring tool uses aquila.util.router.DMAMonAgent – interface for monitoring. When measurement flow start time is reached, the daemon requests starting monitoring from the router –object.
Preconditions	Flow information is available and start time has been reached
Postconditions	Router monitoring becomes active
Notes	
Status	Done

Activity No.	end-to-end.measurement.router.3
Title	Retrieving of results from Router –object.
Description	The Router monitoring tool retrieves monitoring results from router –object in user-defined intervals.
Preconditions	Router monitoring has been started in Router object
Postconditions	Monitoring results are correctly retrieved
Notes	
Status	Done

Activity No.	end-to-end.measurement.router.4
Title	Retrieving of results from EAT.
Description	The Router monitoring tool retrieves monitoring results from EAT in user-defined intervals.
Preconditions	A measurement flow has been started with an active reservation
Postconditions	EAT per-flow results are correctly retrieved
Notes	
Status	Done

Activity No.	end-to-end.measurement.router.5
Title	Saving of results to the measurement database
Description	The results of monitoring are saved to the DMA measurement database.
Preconditions	One result save interval has passed, and results were retrieved
Postconditions	Results are written to the database.
Notes	
Status	Done

5 Setup of the trial sites (Installation and user guide)

5.1 General instructions

The AQUILA resource control layer can be installed either on a Windows NT machine or Solaris. The path names specified in the following always use the UNIX-style syntax, separating the directories with a slash (/) character. A backslash (\) as the last character of a line is used as an indication, that the command will be continued on the next line.

For the following instructions, it is assumed, that a JAVA runtime JDK1.4.0 environment is already installed, and that the /bin directory of the JDK is in the search path.

The AQUILA RCL components themselves will be installed relative to a project directory. $project will be used as a placeholder for this directory in the following.

The set-up described in the following allows, that software components belonging to different domains may run on a single machine. To keep a common set-up, the specified procedure should be followed, even if this is not the case.

5.1.1 Fetching the AQUILA components

The components of the Resource Control Layer are all written in Java. You should make sure that JDK 1.4.0 is installed in the PCs / Workstations. The Java classes that make up the AQUILA components are grouped in Java packages, according to which package / module they belong to. For the AQUILA project, a general package structure has been selected, with the aim to ease the work of developers, as well as group the different Java files. The package structure used in all AQUILA files is the following:

```
aquila/
     idom/
       bgrp/
       gwks/
       neighb/
     manag/
       qmtool/
     rcl/
       rca/
           network/
       aca/
       eat/
           eatManager/
           eatPersistence/
           api/
           appProfile/
           proxy/
           converter/
           gui/
           script/
       service/
       subscriber/
```

```
tc/
util/
    alive/
    corba/
    event/
    inet/
    main/
    persist/
    prisig/
    rcrule/
    router/
    trace/
```

For example, all files of the EAT belong to the aquila.rcl.eat package. Especially those that belong to the Proxy are grouped in the aquila.rcl.eat.proxy package. All Java packages are compressed in ZIP files and stored in the AQUILA FTP server, using the following convention:

nameofComponent-xx.yy.zz.zip

where:

- nameOfComponent is the name of the AQUILA module (e.g. aca or rca)
- xx / yy / zz is the year / month / day of distribution

5.1.2 Installation and compilation

The compilation process uses a script, which performs the following steps:

- unpacks the ZIP files containing the AQUILA components
- runs the IDL compiler (for CORBA) and the XJC compiler (for JAXB)
- automatically corrects some errors in the XJC generated files (the XJC compiler is an early access release, which contains some bugs)
- runs the JAVA compiler to generate the class files
- packs the class files into a JAVA archive (JAR file)

Before the script can be started, a complete set of zip files has to be collected in the production directory. The settings, especially for the variables “jdk” and “aquila”, have to be adapted to the actual environment.

The script is listed in the appendix of this deliverable.

5.1.3 Running a Java class

In general, in order to run a Java class that is part of a package structure, the following command should be used (from the *classes* directory):

```
java aquila.nameOfPackage.nameOfComponent.nameOfClass
```

where:

- NameOfLayer is the name of the layer the component belongs to (e.g. idom or rcl)
- nameOfComponent is the name of the package (e.g. aca or rca)
- nameOfClass is the name of the Java class to be run (e.g. Aca or Rca)

5.1.4 Property file

All AQUILA components, which are based on the aquila.util.main.Main class, require a property file. The software offers possibilities to specify the name and location of this file. This facility will be used, if components for several domains are running on a single machine. The location and name of the property file is:

Property file path and name: $project/as<asnumber>.rc

Example for AS 65000: $project/as65000.rc

The content of the property file is as follows:

```
org.omg.CORBA.ORBInitialHost:   <nameserverhost>
org.omg.CORBA.ORBInitialPort:   <nameserverport>
java.naming.factory.initial:    com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:
                ldap://<ldaphost>:<ldapport>/cn=as<asnumber>,o=aquila,c=eu
```

For the proper entries in line 1, 2 and 4, please refer to the following chapters. If the AQUILA component, the LDAP database and the CORBA name server are running on the same machine, a typical property file for AS 65000 could look like:

```
org.omg.CORBA.ORBInitialHost:   localhost
org.omg.CORBA.ORBInitialPort:   5000
java.naming.factory.initial:    com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:       ldap://localhost:389/cn=as65000,o=aquila,c=eu
```

5.1.5 Name Server

For each domain, a CORBA name server must be running. We use the transient name server “tnameserv” contained in the JDK for this purpose.

To distinguish several name servers running on the same machine, a unique port number must be used. Assuming that the AS numbers are in the range 65000..65535, it is proposed to derive this port number from the AS number by removing the leading digit “6”. So port 5000 will be used for AS 65000 and so on.

The name server is started with the following command:

```
tnameserv –ORBInitialPort <port>
```

The same port number must be specified in the corresponding property file (see previous chapter).

On a UNIX system, the server process can be started in background by appending a trailing ampersand "&" to this command line.

The name of the host running the name server and the port number must be entered in all property files for components, which will access this name server.

5.1.6 LDAP Server

In contrast to the name server, a single LDAP server instance is used, even if multiple domains share the same machine. Different name contexts are used to separate the entries for multiple domains. The default LDAP port number 389 is used.

Netscape directory server 4.1 or higher is used as LDAP server. During installation, the following DN should be specified as the root:

```
o=aquila,c=eu
```

After installation, write access for everybody must be enabled on the root context.

There is a number of configuration-independent items in the LDAP database, e.g. for trace configuration, network services, etc. To set up this information, collect the following XML files in the directory $project/xml:

```
gwks.xml
service.xml
tc.xml
alive.xml
prisig.xml
trace.xml
```

Prepare a list file named staticxml.lst with the following content, and store it in the $project directory:

```
cn=gwks,cn=idom          xml/gwks.xml    aquila.idom.gwks.GlobalServices
cn=service,cn=rcl        xml/service.xml aquila.rcl.service.NetworkServices
cn=tc,cn=rcl             xml/tc.xml      aquila.rcl.tc.TrafficClasses
cn=alive,cn=util         xml/alive.xml   aquila.util.alive.KeepAlive
cn=prisig,cn=util        xml/prisig.xml  aquila.util.prisig.SigDSCP
cn=trace,cn=util         xml/trace.xml   aquila.util.trace.Tracing
```

Prepare a property file for each autonomous system this LDAP server will handle, as described in chapter 0:

Be sure, that both the name server and the LDAP server are running. Make sure, that the aquila.jar file is stored in the directory $project/lib. Then store the information in the LDAP database with the following command:

```
cd $project
java –cl lib/aquila.jar
 –Daquila.util.main.propertyfile=support/rc/as<asnumber>.rc \
 aquila.util.persist.StoreXml –l staticxml.lst
```

Repeat this step for each domain using the appropriate property file name.

Additionally, create an XML file for each user you want to add to the subscriber database. You can store the users in the database either by following a similar procedure with a list file as shown above, or by using individual StoreXml calls:

```
cd $project
java –cl lib/aquila.jar
 –Daquila.util.main.propertyfile=support/rc/as<asnumber>.rc              \
 aquila.util.persist.StoreXml                                           \
 cn=<user>,cn=subscriber,cn=rcl                                         \
 xml/<xmlfile.xml>                                                      \
 aquila.rcl.subscriber.Subscriber
```

5.1.7 Prisig proxy

The AQUILA components use DSCP marking for packets used for communication between the components. However different approaches are used for different types of components:

- JAVA components use the JDK-1.4.0-provided API to mark signalling packets
- The routers use pre-marked packets by default for a telnet connection.
- For the LDAP server and the CORBA naming server, a proxy is used.

To set-up prioritised signalling, the following steps have to be performed:

- Enable prioritised signalling using the QMTool, and set the DSCP to “48”.
- For each non-JAVA server (LDAP, CORBA naming), select and configure a pair of port numbers for the prisig proxy. The following port numbers are just a proposal:
    - forward the LDAP port 389 to 6389
    - forward the CORBA naming service port 5000 to 6000
- Use the forwarded port numbers in the rc files for all components, ***except*** the prisig proxy itself. Note, that you cannot use the forwarded port numbers for the proxy!
- Start the prisig proxy on the same machine as the LDAP server and the CORBA naming service. If these servers run on different machines, you have to start the prisig proxy on each of them.

5.1.8 Trace Server

Start a trace server for each domain with the following command:

```
cd $project
java –cl lib/aquila.jar
 –Daquila.util.main.propertyfile=support/rc/as<asnumber>.rc              \
 aquila.util.trace.TraceMain
```

For performance reasons, disable all trace information, which is not actually needed. Especially logging to the global trace file involves communication with the trace server and therefore affects the overall performance.

5.2 RCL layer

5.2.1 RCA

5.2.1.1 Required software (packages, xml files) and installation procedure

All classes belonging to the RCA agent are located in the aquila.jar library.

For each RCA agent, a configuration entry in the LDAP database is necessary, as described in the next chapter. The main functionality of the RCA agent is the initialisation of the Resource Pool Tree. Moreover, one RCA agent is defined per domain.

5.2.1.2 User guide

To set up a RCA agent, the corresponding Resource Pools need to be created. This information will be stored in the LDAP database and used by the agent during initialisation and operation.

- An RCA agent is composed of a number of Resource Pools, which in sequence are comprised of the corresponding Resource Shares one for TCL and direction. A name is assigned to the RCA agent.

- Each Resource Share interacts with the algorithms implemented in the util.rcrule package in order to increase or reduce their assigned resources. An identifier for the proper algorithm must is specified. The identifier specifies the algorithms used both for increase and reduce of resources. Each identifier determines that a corresponding xml file is specified by the rcrule package with the same name.

- The Resource Pools belonging to the lower level of the RPT are regarded as Resource Pool Leaves.

- To initialise the RPT, the consumed shares of each ACA should also be specified in the appropriate xml file. Since the name of each ACA is also included in the xml file for the RCA, it should be the same with the one specified in the ACA xml file. Therefore, the number of the ACAs specified in the RCA xml, should be equal to the number of xml files determined for each ACA agent.

For each RCA agent, the number of resource pools is specified, where each one is described by the following information:

- Name of the resource pool (the root of the tree is characterised by the name “root”).

- The name of his “father” resource pool (this field is “null” for the root of the RPT).
- Whether or not this resource pool is a leaf of the RPT.
- The identifier of the algorithm used from the rcrule package.
- The names of the “children” Resource Pools (this field is optional, since the resource pools belonging to ACA have no “children”).

Moreover, for each Resource Share, the following fields are required:

- Maximum possible assigned bandwidth.
- Initially assigned bandwidth.
- The traffic class, which this Resource Share belongs to.
- The direction, ingress or egress.

In the following example XML file, one resource pool (”root”) belonging to RCA is configured, which is responsible for two ACAs (“aca1” and “aca2”):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RcaSetting SYSTEM "..\rca.dtd">
<RcaSetting rcaName="rca">
  <ResourcePools>
    <ResourcePool fatherID="null" isLeaf="true" poolID="root" rcRuleName="rule1">
      <ResourceShares>
        <ResourceShare>
          <RshareInfoSetting  ingress="true" maxBW="200" tcID="TCL1" totBW="100">
             </RShareInfoSetting>
        </ResourceShare>
        <ResourceShare>
          <RShareInfoSetting ingress="true" maxBW="200" tcID="TCL2" totBW="100">
          </RShareInfoSetting>
        </ResourceShare>
        </ResourceShares>
      <ChildID poolID="aca1"></ChildID>
      <ChildID poolID="aca2"></ChildID>
    </ResourcePool>
    <ResourcePool fatherID="root" isLeaf="false" poolID="pool1" rcRuleName="rule1">
       <ResourceShares>
        <ResourceShare>
          <RShareInfoSetting ingress="true" maxBW="100" tcID="TCL1" totBW="60">
          </RShareInfoSetting>
        </ResourceShare>
        <ResourceShare>
          <RShareInfoSetting ingress="true" maxBW="100" tcID="TCL2" totBW="60">
          </RShareInfoSetting>
        </ResourceShare>
      </ResourceShares>
    </ResourcePool>
    <ResourcePool fatherID="root" isLeaf="false" poolID="pool2" rcRuleName="rule1">
      <ResourceShares>
        <ResourceShare>
          <RShareInfoSetting ingress="true" maxBW="100" tcID="TCL1" totBW="60">
          </RShareInfoSetting>
        </ResourceShare>
        <ResourceShare>
          <RShareInfoSetting ingress="true" maxBW="100" tcID="TCL2" totBW="60">
```

```
          </RShareInfoSetting>
        </ResourceShare>
      </ResourceShares>
    </ResourcePool>
  </ResourcePools>
</RcaSetting>
```

Store the XML file in the LDAP database under the following DN:

```
cn=<rcaname>,cn=rca,cn=rcl
```

and start the RCA agent with the following command:

```
cd $project
java –cl lib/aquila.jar
 –Daquila.util.main.propertyfile=as<asnumber>.rc \
 aquila.rcl.rca.Rca
```

5.2.2 ACA

5.2.2.1 ACA startup

- An example of a needed “file.rc” file:

```
org.omg.CORBA.ORBInitialHost:   host-name
org.omg.CORBA.ORBInitialPort:   port-number
java.naming.factory.initial:    com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:       ldap://hostname:389/cn=as-name,o=AQUILA,c=EU
```

- Needed external packages beside the aquila.jar package:

```
jaxb-rt-1.0-ea.jar
jta20.jar
```

- Start command in order to start the ACA:

```
java -classpath "class-path-name" -Daquila.util.main.propertyfile=”file.rc”
aquila.rcl.aca.Aca aca_name
```

5.2.2.2 ACA Configuration

With the QMTool an ACA entity has to be created and linked to the configured part of the RCA.

The ACA properties only include the information which bgrp agent is logically connected to this ACA. This value is optional and if the value hasn’t been set, no binding to a bgrp agent is supported.

5.2.2.3 Router configuration

A well-configured router information entry is expected and needed for the startup of the ACA. Therefore all router interfaces including the logical loopback interfaces have to be configured at each router component in the “router access entry”. A very important thing is, that the name of the router entry is identical with the aca_name, used in the start script (see above).

5.2.3 EAT

5.2.3.1 EAT Installation

The EAT is included in the package **aquila.rcl.eat**. It contains of the following sub-packages:

- **api** - EAT API containing the by idlj generated java files from api.idl
- **appProfile** - Profile Manager code and the by the jaxb compiler generated java files from ApplicationProfile.dtd and ServiceComponentProfile.dtd
- **converter** - TechnicalCharacteristicConverter code and the by idlj generated files from converter.idl
- **eatManager** - EAT Manager java code, the by idlj generated files from eatManager.java, and the by jaxb generated files from eat.dtd
- **eatPersistence** - Reservation History java code and the by idlj generated files from eatPersistence.idl
- **gui** - AQUILA portal jsp and java files, Sample Client
- **proxy** - Proxy Manager and Application Proxies and the by idlj generated files from proxy.idl (see also 5.2.3.2)
- **script** - EAT Script

Additionally, the following EAT files are required in the aquila project directory:

IDL: **api.idl, appProfile, converter.idl, eatManager.idl, eatPersistence.idl, proxy.idl**

DTD: **eat.dtd, proxy.dtd, ApplicationProfileV10.dtd, ServiceComponentProfileV1.dtd**

XJS: **eat.xjs, proxy.xjs, ApplicationProfileV10.xjs, ServiceComponentProfileV1.xjs**

XML: **eat_name.xml[1], sip_name[1].xml, h323_name[1].xml, NetMeeting_3_01_v2.xml, NetMeeting_3_01_Speech_v2.xml, NetMeeting_3_01_Video_v2.xml** and other application profiles

The following external libraries are required for the EAT:

JAXB:	**jaxb-rt-1.0-ea.jar, jaxb-xjc-1.0-ea.jar**	
JAXP:	**jaxp.jar, crimson.jar, xalan.jar**	(for EAT Script)
JDBC for MySQL:	**mm.mysql-2.0.11-bin.jar**	(for Persistence Layer)

[1] "name" is a placeholder. The files will have names like "eat_dresden".xml, "sip_athens.xml", etc.

The following external libraries are required for the AQUILA Portal:

JAXP:	**jaxp.jar, crimson.jar, xalan.jar**	
Servlets:	**servlet.jar**	(in Tomcat)

The file eat_name.xml is for the EAT settings: It will have the following content (example):

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE EATSettings SYSTEM "eat.dtd">
<EATSettings ACAName="aca_name">
    <Database URL="mysql://localhost/reservation_history"/>
    <ApplicationProfiles Path="cn=appProfile,cn=eat,cn=rcl">
        <Profile>NetMeeting_3_01_v2</Profile>
        <Profile>...</Profile>
    </ApplicationProfiles>
</EATSettings>
```

where “aca_name” is the name of the associated ACA;
the Database tag refers to the URL of the MySQL database (this tag is optional); and
the ApplicationProfile tag contains the “path” to the LDAP context (should not be changed) as well as the several available application profiles (not the service component ones).

Installation procedure:

1. Make sure that the file idl.config contains the following entries:

```
PkgPrefix.api=aquila.rcl.eat
PkgPrefix.appProfile=aquila.rcl.eat
PkgPrefix.converter=aquila.rcl.eat
PkgPrefix.eatManager=aquila.rcl.eat
PkgPrefix.eatPersistence=aquila.rcl.eat
PkgPrefix.proxy=aquila.rcl.eat
```

2. Compile the IDLs:

```
idlj –fallTIE api.idl
idlj –fallTIE appProfile.idl
idlj –fallTIE converter.idl
idlj –fallTIE eatManager.idl
idlj –fallTIE eatPersistence.idl
idlj –fallTIE proxy.idl
```

3. Compile the DTDs:

```
java -jar jaxb-xjc-1.0-ea.jar eat.dtd eat.xjs

java -jar jaxb-xjc-1.0-ea.jar proxy.dtd proxy.xjs

java -jar jaxb-xjc-1.0-ea.jar
       ApplicationProfileV10.dtd ApplicationProfileV10.xjs

java -jar jaxb-xjc-1.0-ea.jar
       ServiceComponentProfileV1.dtd ServiceComponentProfileV1.xjs
```

4. Download and compile the EAT's JAVA source code of the above mentioned packages.

5. Start tnameserv and store the XML files:

```
java aquila.util.persist.StoreXml
        "cn=eat_name,cn=eat,cn=rcl"
        eat_name.xml
        aquila.rcl.eat.eatManager.EATSettings

java aquila.util.persist.StoreXml
        "cn=sip_name,cn=proxy,cn=eat,cn=rcl"
        sip_name.xml
        aquila.rcl.eat.proxy.ProxySettings

java aquila.util.persist.StoreXml
        "cn=h323_name,cn=proxy,cn=eat,cn=rcl"
        h323_name.xml
        aquila.rcl.eat.proxy.ProxySettings

java aquila.util.persist.StoreXml
        "cn=NetMeeting_3_01_v2,cn=appProfile,cn=eat,cn=rcl"
        NetMeeting_3_01_v2.xml
        aquila.rcl.eat.appProfile.ApplicationProfile

java aquila.util.persist.StoreXml
        "cn=NetMeeting_3_01_Speech_v2,cn=appProfile,cn=eat,cn=rcl"
        NetMeeting_3_01_Speech_v2.xml
        aquila.rcl.eat.appProfile.ServiceComponentProfile

java aquila.util.persist.StoreXml
        "cn=NetMeeting_3_01_Video_v2,cn=appProfile,cn=eat,cn=rcl"
        NetMeeting_3_01_Video_v2.xml
        aquila.rcl.eat.appProfile.ServiceComponentProfile

... (other profiles)
```

6. Install the MySQL DBMS on the local host or any other computer and create an empty database "reservation_history", for example.

7. Create the database tables:

```
java aquila.rcl.eat.eatPersistence.CreateTables
        mysql://localhost/reservation_history
```

8. Run the EAT (based on the settings in aquila.rc):

```
java aquila.rcl.eat.eatManager.EAT <eat_name>
```

9. Run the application proxies (see below).

5.2.3.2 Proxy Installation

For the second trial, the main topic as far as the Proxies are concerned, is the support of the Session Initiation Protocol (SIP). However, the H.323 proxy developed during the first trial can also be used in parallel.

5.2.3.2.1 Common information

The java package **aquila.rcl.eat.proxy** contains:

- The generated java files from idlj and jaxb.
- The java files for the Proxy Manager and the two Proxies: SIP and H.323

This package also contains another package: **aquila.rcl.eat.proxy.sipserver**, a modified version of the SIP server provided by Coritel.

Both Proxies communicate with the EAT through CORBA, so they can be located in a different process or workstation from the EAT. This means that they need a separate initialisation file (aquila.rc) that will contain the following lines:

```
org.omg.CORBA.ORBInitialHost:     <hostname>
org.omg.CORBA.ORBInitialPort:     <port number>
java.naming.factory.initial:      com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:          ldap://ip_address:389/o=aquila,c=EU
```

For the initialisation of the SIP Proxy, the following command line should be used:

```
java aquila.rcl.eat.proxy.SIPProxy
        -Daquila.util.main.propertyfile=$HOME\aquila.rc
        <sip_name>
```

and for the H.323 Proxy:

```
java aquila.rcl.eat.proxy.H323Proxy
        -Daquila.util.main.propertyfile=$HOME\aquila.rc
        <h323_name>
```

The names “sip_name” and “h323_name” are also used for the retrieval of the initialisation information from the LDAP server. The server must have entries for each Proxy in its domain, and they should have the following structure:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ProxySettings SYSTEM "proxy.dtd">
<ProxySettings EATName="eat_name">
    <ProxyID>1</ProxyID>
    <FullName>SIP Proxy</FullName>
    <ControlPort>4545</ControlPort>
</ProxySettings>
```

where, “eat_name” is the name of the EAT Manager (used for JNDI lookup) and “ProxyID” is the ID of each proxy (used in the Application Profiles). The Control port tag is not used by any of the two proxies and is ignored.

5.2.3.2.1.1 Specific information

SIP Proxy

For the SIP Proxy, an additional initialisation file is required. It is called “sipusers.txt” and is placed inside the home directory of the AQUILA structure. This file contains information

about the users that exist in the domain controlled by the SIP Proxy. All those users must be listed in the file, in order for the SIP Proxy to locate them and forward to them the SIP messages.

First of all, the exact number of users must be given in the first line of the file. Each user is characterised by the following 5 strings:

- User name
- Full name
- SIP name
- IP address of his/her workstation
- Port number that the SIP User Agent is listening to

The third and fourth element, when concatenated, make up the full sip address of the individual (e.g. sip:sip_name@my.workstation.com)

Here is an example file:

```
4
pluto
Pluto di Topolino
pluto
galileo.coritel.it
5060
luca
normal user
luca
acer.coritel.it
5060
donald
normal user
mac
gauss.coritel.it
5060
UserA
test_seat user
UserA
eulero.coritel.it
5060
```

H.323 Proxy

For the H.323 Proxy to operate properly on Windows platforms, a DLL file is required. It is named "**filter.dll**" and should be placed in the directory where the java process is started. This means that, if the Proxy is started from the command line, then the file should be placed in the AQUILA home directory. If JBuilder or another tool is used for running the Proxy, then the DLL should be located in the "bin" directory of the JDK.

5.2.3.3 User guide

The EAT's user guide is described in detail in [D2203].

5.3 Measurements

5.3.1 Overview

The distributed measurement system to be used in AQUILA mainly consists of a measurement server (or measurement management station) and several measurement client stations. This chapter is structured as follows: First, a section containing a description of the basic installation procedure of SuSE Linux 7.3 and the installation of the GPS drivers. This should be use for both, the measurement server and the clients. After that a section follows, which contain the installation instructions of the software components, which need to be installed on the measurement server. The third part contains the installation instructions which have to be installed at the measurement clients.

The following list contains an overview on the software to be installed on the different machines:

Measurement Management Station with

- SuSE Linux 7.3 (If another Version is used the directory structure might be a little bit different and therefore changes in the installation and start/stop scripts are necessary)
- NTP daemon
- Apache Web-Server extended with: PHP/Perl script support, authentication module for the MySQL database
- MySQL database
- Unix ODBC driver + the MySQL specific MyODBC driver
- Additional support software modules: gnuplot, gd library
- DTA masterstation software module (called "Master") and the DTA measurement master monitor program (called "mmon")

- SPU measurement management process (called “cmcaller”)
- ELI router QoS monitoring tool
- The DMA GUI

Measurement Client Stations with:

- SuSE Linux 7.3 (If another Version is used the directory structure might be a little bit different and therefore changes in the installation and start/stop scripts are necessary)
- Meinberg GPS card and device driver
- NTP daemon
- DTA measurement agent (called “magent”) and the DTA measurement agent monitor program (called “mamon”)
- SPU measurement agent (called “cmdaemon ”)

In the following the files are listed, which are needed to follow the installation instructions below besides the SuSE Linux 7.3 CD-ROMs. These files are located at the ftp-server of the AQUILA project in the directory “software/measurement_tools”. [yy.mm.dd] is for the date of the version. Be sure to take the most current! The contents of the tar.gz-files can be displayed by using the command: “tar -tzf *filename*”.

dma-setupkit-yy.mm.dd.tar.gz: contains the necessary files for the measurement server (Apache, PHP, MySQL, AQUILA Measurement Database, GUI, Router Monitor, CMCaller)

dma-dta-master-yy.mm.dd.tar.gz: contains the binary file of the master and a monitor program. It also contains a install- and a start/stop-script.

dma-dta-magent-yy.mm.dd.tar.gz: contains the binary file of the measurement agent, a monitor program an the new measurement agent controller program “MACON”. It also contains a install- and a start/stop-script.

dma-cmdaemon-yy.mm.dd.tar.gz: contains the source-code and the start/stop-script for the cmdaemon (SPU measurement agent).

support/dma-gps-ntp-yy.mm.dd.tar.gz: contains the Meinberg GPS-Driver and NTP-binaries for this driver.

support/mbgfl101.exe: contains a windows application for upgrading the flash ROM of the Meinberg GPS card.

support/GPCI416.ZIP: contains the code for the Meinberg GPS card flash ROM, version 4.16.

5.3.2 SuSE Linux 7.3 Installation on Measurement Machines

5.3.2.1 Basic Installation

This chapter contains a short reference list for the installation of SuSE Linux 7.3 from CD-ROM. Installing via network should be similar with the difference that the network configuration set-up is done as the first step. The bold items mark the most important points, which are differing from a standard installation.

- **Change clock in BIOS approximately to UTC**
- **Disable the power saving options in the BIOS**
- **Set PNP O/S to NO in BIOS**
- Insert CD1 and boot from CD
- Choose “Installation” from the lilo boot menu
- Select Language “English (GB)” => NEXT
- Select your Keyboard Layout
- **Select Timezone “Global/GMT”**
- **Select Hardware clock set to “GMT” => NEXT**
- Select “New Installation” => NEXT
- Select Partitioning (Here no recommendations are made, as this is very dependent on the specific system. Usually you can accept the suggestion from Linux) => NEXT
- Select Software: We strongly recommend using “Minimum graphical system (without KDE)” because of performance reasons. => NEXT
- System Boot Configuration: Usually just => NEXT
- Choose a password for the root user => NEXT
- Configure a user as follows:
  - First name: Aquila; Last name: User; User login: aquila; Password: choose yourself
- Confirm the installation => NEXT
- Follow the installation instructions
- Set-up graphical display

- Login as root
- Do the network configuration:
- Start yast2 (e.g. from an xterm)
- Choose Network/Advanced => Expert network configuration
- Activate Network Card and configure IP address, subnet mask, hostname, name server and routing
- Remove CD and reboot, if the hostname has been changed

5.3.2.2 Installation of supporting software packages

The SuSE installation, which is now on your system, excludes some useful software packages, which should be installed now. Therefore the SuSE setup tool called "YaST2" should be used.

- Start yast2 (e.g. from an xterm)
- choose "Software" => Install/Remove software
- From Group Networking/Utilities: select **traceroute, tcpdump, nmap**
- From Group System Environment/Libraries: select **libelf**
- From Group System Environment/Daemons: select **xntp**
- From Group Applications/Engineering: select **gnuplot** (only necessary if the machine is dedicated as measurement server)

Optional, the following packages are recommended:

- From Group Development: select **gcc, gpp**
- From Group Development/Building: select **make**
- From Group Applications/Networking: select **ngrep**
- From Group X11/Applications/Internet: select either **Netscape6, netscape *or* opera**

Confirm the installation with 'OK' and close YaST2. If the machine is *not* equipped with a GPS card, you have to configure ntp by editing `/etc/ntp.conf`. In Line 41 remove the leading '##' and enter the IP address of the nearest NTP-Server (e.g. one of your hosts with GPS equipment). To start NTP during system start-up the START_XNTPD variable has to be set to "yes" in `/etc/rc.config` . The value of XNTPD_INIT_NTPDATE should be set to "`/usr/sbin/ntpdate x.x.x.x`", where x.x.x.x has to be replaced with the next NTP server

(e.g. a machine where GPS-equipment is installed). Start the NTP-Daemon with `/etc/init.d/xntpd start`. If the machine itself has GPS equipment installed, please install the GPS drivers as described below.

If you think, it's necessary to have any other software to be installed on your system from the SuSE distribution (e.g. Firewall, etc.), please perform the installation **now**, as SuSEconfig (which is executed almost every time when leaving YaST2), can be very dangerous to the software, which will be installed in the following chapters. Be also sure, that the software mentioned below (e.g. apache, mysql, etc.) is NOT installed by using YaST2!!!

5.3.2.3 Installation of Meinberg GPS Hardware Driver (GPS167PCI, if applicable)

When a Meinberg GPS card is installed in a measurement station, you should check, whether the hardware was detected by the Plug & Play BIOS and an interrupt for the hardware was assigned. To check this, type (as "root"):

```
lspci
```

In the Meinberg entry (e.g. Meinberg Funkuhren Unknown Device (rev 0). Vendor id=1360. Device id=201) an interrupt must be listed.

If not, check the BIOS. The entry "Plug & Play O/S" (or similar) must be set to "NO"! This switch decides who initializes the PCI bus, BIOS or OS. For a Linux PC it has to be the BIOS!

Before install the Meinberg GPS device driver be sure NTP is installed as described above

Get the Meinberg device driver and the NTP daemon for Meinberg GPS clocks from the AQUILA server (/software/measurement_tools/support/dma-gps-ntp-yy.mm.dd.tar.gz). Copy this file to the measurement PCs.

- Login as root
- Unpack the file with : `tar –xvzf dma-gps-ntp-yy.mm.dd.tar.gz`
- Change into the new directory : `dma-gps-ntp-yy.mm.dd`
- Make the "install" script executable : `chmod u+x ./install`
- Run the install script : `./install`

```
mc-7-14:~/dma-gps-ntp-02.03.26 # ./install

install Meinberg device driver
make devices in /dev
copy Meinberg programs
install new NTP programs
load Meinberg GPS device driver
set PC clock via GPS before NTP start
mbgtime v1.0 (c) Meinberg 2001
Date/time set to 26.03.02 12:43:21.94
                                                          done
Starting network time protocol daemon (NTPD)              done
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 GENERIC(0)      .DCFa.           0 l    -   64    0    0.000    0.000 4000.00
 LOCAL(0)        LOCAL(0)        10 l    -   64    0    0.000    0.000 4000.00

mc-7-14:~/dma-gps-ntp-02.03.26 #
```

Figure 5-1: GPS and NTP install script

The script copies the device driver, programs and start/stop scripts. Then the install script loads the Meinberg device driver, start the NTP daemon and run the NTP test program “ntpq -p”. One of the printed lines of the ntpq output must start with “GENERIC.

```
mc-7-14:~/dma-gps-ntp-02.03.26 # ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*GENERIC(0)      .DCFa.           0 l   21   64  377    0.000    3.795   0.336
 LOCAL(0)        LOCAL(0)        10 l    5   64  377    0.000    0.000   0.008
```

Figure 5-2: NTP test

After some minutes run “`ntpq -p`” again. Now an asterisk (*) must be printed in front of the GENERIC line. This means that NTP synchronises the PC clock on the GPS time. It can take some hours before the synchronisation is good enough for an exact measurement. Therefore the “offset” and “jitter” value has to be less than “1.0”.

5.3.2.3.1 Upgrade of the Flash ROM of the Meinberg GPS-card (if applicable)

This step has to be done, when the version of the flash is below v4.15. The current version is 4.16. Please check, whether a more recent version is available. You can check the flash version by running the “mbgtest” (included in `dma-gps-ntp-yy.mm.dd.tar.gz`) program.

The file “support/mbgfl101.exe” has to be installed on a MS Windows machine. The file “support/GPCI416.ZIP” has to be copied and extracted to this machine as well. (Most likely a laptop)

In the next step a serial cable that connects on the one hand to the COM port of the win-machine and on the other hand to the serial port at the back of the Meinberg card is needed.

(It is a simple cable, where only the TX on the sender side connects to the RX on the receivers side and vice versa)

- Connect the cable.
- Stop NTP, if already running ('/etc/init.d/[x]ntpd stop').
- Stop the kernel modules working with this hardware (e.g. mbgclock), if already installed ('lsmod' is used for displaying the running modules, 'rmmod *modulename*' removes the module from the kernel).
- Start MBG-flash application on win machine.
- Change to the right processor of the GPS card (most likely C166/167).
- Open the flash file (GPSPCI.416, which was packed in GPCI416.ZIP).
- Punch the little switch at the back of the Meinberg-card with a sharp thing (pen or so). The card is set into bootstrap-mode.
- Begin to upgrade the flash ROM (Simply click "FLASH" in the mbgflash application).
- Disconnect cable when ready.
- Reboot the computer to bring the card back to normal mode.

5.3.3 Installation procedures for the Measurement Management Station

5.3.3.1 Java2 Installation

Login as root and download Java2 Platform Standard Edition (Linux GNUZIP Tar shell script) from the SUN homepage (java.sun.com). Depending on the version, the filename will be e.g.: **j2sdk-1_3_1_03-linux-i386.bin**

Copy the file into the `/root/` directory and execute

```
cd /usr/local/
/root/j2sdk-1_3_1_03-linux-i386.bin
```

After the license agreement, Java2 will be installed into `/usr/local/jdk1.3.1_03/` . A link has to be added:

```
ln –s /usr/local/jdk1.3.1_03/ /usr/local/j2sdk
```

The binaries (e.g. `java`) can then be found in `/usr/local/j2sdk/bin/` , which has to be added to the path.

Note: Possibly Java2 has been already installed from the SuSE Linux distribution (e.g. necessary if Netscape6 was installed), but both Java installations can run on the system in parallel.

5.3.3.2 Installation of the additional software including AQUILA DMA components

In contrast to the first trial a single installation script for all parts of the measurement management station is provided. The installation contains:

- MySQL
- UnixODBC, MyODBC
- Apache, PHP, gd, Gnuplot
- SPU Measurement Management Process (cmcaller)
- ELI Router Monitor
- AQUILA measurement database
- AQUILA DMA Graphical User Interface

The installation script installs all necessary files using absolute path names. It is therefore strongly recommended to use a fresh installed SuSE Linux 7.3 standard installation (if necessary with X-Server). Updated files which probably come along with bug-fixes can be installed (overwritten) separately afterwards. Furthermore, a de-installation script is produced during the installation process, which can be executed to remove exactly the installed files. *Warning:* Files, which are already on the machine before the installation and part of the installation file tree will be overwritten.

- Login as root
- copy the most recent dma-setupkit-yy.mm.dd.tar.gz to your machine (e.g. by scp to /root/)
- extract the shell script dma-setup from the setupkit and execute the script as follows:

```
tar –xzf dma-setupkit-yy.mm.dd.tar.gz dma-setup
./dma-setup dma-setupkit-yy.mm.dd.tar.gz
```

Reboot the computer or start the services as described on the screen

5.3.3.3 Configuration of the SPU CMCaller

The CMCaller comes up with a standard configuration file, which can be found in `/etc/dma.rc`. The parameters that can be configured are described within this file. If the installation was done like described above, usually only the variables EAT_name, EAT_username and EAT_password have to be adapted to configure, which EAT will be contacted by the CMCaller for resource reservations. Note that the CMCaller has to be restarted (`/etc/init.d/cmcaller stop; /etc/init.d/cmcaller start`) after changing the configuration file.

5.3.3.4 Installation of the DTA masterstation software module (master) and the master monitor program (mmon)

Login to the Masterstation as user “root”

Copy the archive “dma-dta-master-yy.mm.dd.tar.gz” to the desired place (e.g. /home/master/

Unpack the archive with tar -xzvf dma-dta-master-yy.mm.dd.tar.gz. The unpacked files will be stored in the new subdirectory (e.g. /home/master/install)

Change to the new subdirectory

One enhancement of the master software module is that it is now able to make resource reservations for the measurement flows (if configured in the measurement database). The master realizes the resource reservation via generating an XML-File, according the specified reservation parameters in the measurement database and executing this file by calling the EATScript command in tool. For calling this command line tool several parameters are necessary. These parameters are defined in the “master.rc”.

Open the Masterstation configuration file "master.rc" in an editor and adjust the following parameters (**This are the valid parameters for the testbed in Vienna**) :

MY_IP_ADDR=10.0.1.1 (IP address of the masterstation)

DB_IP=127.0.0.1 (IP address of the PC where the measurement database is located)

DB_USER=aquila

DB_PASSWORD=aquila

DB_NAME=aquila

DB_PORT=3306

MODE=394248 (This value should not be modified)

JVM_JavaPath=/usr/lib/java/bin/java (Path of the JVM)

EAT_enable=1 (Enable (1) / Disable (0) resource reservation)

EAT_user=rfr (EAT username)

EAT_pw=geheim (EAT password)

EAT_aquila_rc_path=/usr/local/aquila/as65000.rc (Path of the AQUILA resource file)

EAT_script_path=aquila.rcl.eat.script.EATScript (Path of the EAT script)

EAT_name=eat_er1spu (Name of the used EAT)

EAT_aquila_jar_path=/usr/local/aquila/aquila.jar:/usr/local/aquila/crimson.jar (Pathes of the necessary JAR file)

EAT_dtdPath=../usr/local/aquila/EATScript.dtd (Path of the EAT document type description file. **Attention !!!**: **This has to be the relative path**)

Store the config file

Start the install script with ./install

After the script is finished (The message " Reboot please !!!" is displayed), reboot the Masterstation

During the rebooting process verify that the Masterstation software starts correct (The message "T-NOVA Master .... (started)" will be displayed)

Login as root

Call the masterstation monitor with mmon

Depending on the shown mode (see Figure 5-3) perform one of the following actions:

Connecting: Master cannot connect to the database. Check the parameters in the configuration file "/root/master.rc". After changing the parameters, stop the software with "/sbin/init.d/master_s stop" and start the software again with "/sbin/init.d/master_s start"

Read_ID: Master is connected to the database but couldn'd find his entry in the hop table. Make a new entry with the Configuration-GUI in the database (Menu Hop/New Hop) with the following values:

- **Master ID**: NULL (required, must be NULL)
- **Hoptype**: Master Station (required)
- **IP Adress**: IP-Address of this master station (required). **Must be the same as in the master configuration file "master.rc" in the directory /sbin.**
- **NetMask**: Netmask of the master station's subnet (required)
- **Hopname**: Desired name for this master station (required)

Hop description: Description (optional)

Running: The master station is running correct.

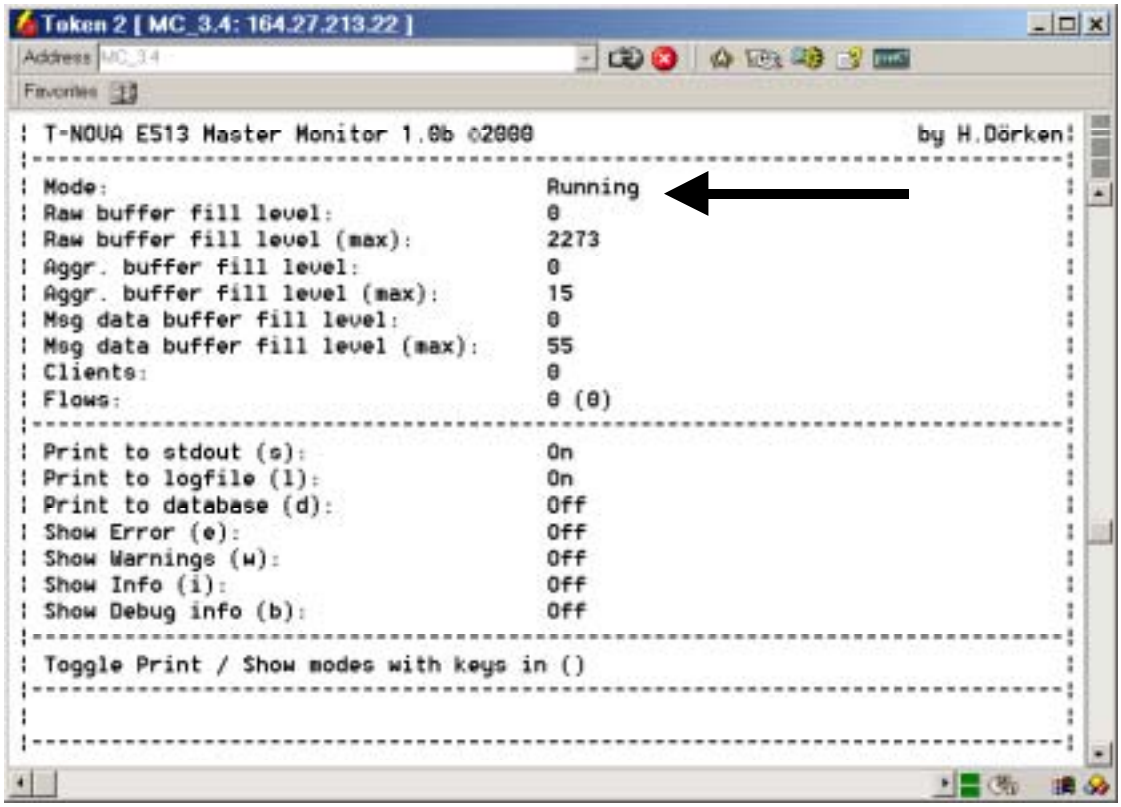


Figure 5-3: Masterstation monitor (mmon)

5.3.4 Installation of the DTA measurement agent controller (MACON)

MACON is a easy to use GUI for DTA measurement agents. The program is a Java jar file and is executable on every PC (Windows, Linux, Solaris) with JAVA2 (but there are some problems with the fonts on Linux-PCs).

- Copy the archive "dma-dta-macon-yy.mm.dd.zip" to the PC where normally all configuration is done (e.g. master station).
- Unpack the archive to a desired directory
- Start with : `java –jar macon.jar`

5.3.5 Installation of the DTA measurement agent controller (MACON)

MACON is a easy to use GUI for DTA measurement agents. The program is a Java jar file and is executable on every PC (Windows, Linux, Solaris) with JAVA2 (but there are some problems with the fonts on Linux-PCs).

- Copy the archive "dma-dta-magent-yy.mm.dd.tar.gz" to the folder of the PC

- Unpack the archive to a desired directory (with tar –xzvf or WINZIP on a Windows-based PC)
- Start with : `java –jar macon.jar`

5.3.6 Installation Procedures for the Measurement Client Stations

5.3.6.1 Installation of the DTA measurement agent (magent)

The following description shows the installation of a DTA measurement agent on a Linux PC.

Get the measurement agent archive from the AQUILA server (/software/measurement_tools/dma-dta-agent-yy.mm.dd.tar.gz). Copy this file to the measurement PCs.

- Login as root
- Unpack the file with : `tar –xvzf dma-dta-magent-yy.mm.dd.tar.gz`
- Change into the new directory : `dma- dta-magent-yy.mm.dd`
- Make the "install" script executable : `chmod u+x ./install`
- Run the install script : `./install`

```
mc-7-14:~/dma-dta-magent-02.03.20 # ./install
install magent in dirctory /usr/sbin
install start script

wait a moment ....
Starting T-Systems Nova TZ E513 MAgent

T-Systems E513 ©2002  measurement agent 3.150   by J.Mende
MAgent              NTP loaded
MAgent              NTP exact
MAgent              max raw data:50000
MAgent              standard kernel scheduler
MAgent              sender list installed
MAgent              timer installed
MAgent              aggregation started
MAgent              analyse started
MAgent              server started

mc-7-14:~/dma-dta-magent-02.03.20 #
```

Figure 5-4: DTA measurement agent install script

The script copies the programs and start/stop script. Then the install script starts the measurement agent automatically. Start and stop the measurement agent manually with command:

- `/etc/init.d/magent start`
- `/etc/init.d/magent stop`

5.3.6.2 Installation of the DTA measurement agent monitor (MAMON)

MAMON is a monitor for DTA measurement agents. The program is a Java jar file and is executable on every PC (Windows, Linux, Solaris) with JAVA2 (but there are some problems with the fonts on Linux-PCs).

- Copy the archive "dma-dta-mamon-yy.mm.dd.zip" to the PC where normally all configuration is done (e.g. master station).
- Unpack the archive to a desired directory
- Start with : `java –jar mamon.jar`

5.3.6.3 Installation of the SPU measurement daemons (CMDaemon)

5.3.6.3.1 Installing the SPU agent

The SPU agent (called cmdaemon) is distributed as a tar-zipped binary.

To install the CMDaemon, simply extract the dma-cmdaemon-yy.mm.dd.tar.gz file:

```
tar –xzPf dma-cmdaemon-yy.mm.dd.tar.gz
```

The -P option is necessary to force using the absolute path names. The following files will be extracted to your system:

```
/usr/local/cmtoolset/bin/cmdaemon
/etc/init.d/cmdaemon
/etc/init.d/rc3.d/K99cmdaemon
/etc/init.d/rc3.d/S99cmdaemon
/etc/init.d/rc5.d/K99cmdaemon
/etc/init.d/rc5.d/S99cmdaemon
/var/log/cmdaemon.log
cmd_uninstall
```

Change the RESULTSERVER variable (default is 10.0.1.1, which is the CMS in Vienna) by editing the cmdaemon startscript (`/etc/init.d/cmdaemon`) to your CMCaller IP address (measurement management station). Be sure that the CMCaller is up and running, when the daemons will be started, so that they can register themselves at the management station.

WARNING: Please beware, that the CMDdaemons will report the results to the configured measurement server (RESULTSERVER). As in the current trial set-up 2 measurement servers are installed (Warsaw, Vienna), the CMDaemons have to be **re-configured for the inter-domain scenarios**, so that they all report to **ONE SINGLE measurement server** (either all to Warsaw or all to Vienna)!!!

5.3.6.3.2 Start and Test

The CMDaemon can be controlled by executing `/etc/init.d/cmdaemon start|stop|status`.

After starting the CMDaemon (`/etc/init.d/cmdaemon start`), some start log messages, like the version info, GPS location info (if available) and status of start-up will be displayed on the terminal (stdout).

The logging messages for errors and events of the CMDaemon are in `/var/log/cmdaemon.log`. They can be checked e.g. with `tail -f` after start, i.e.:

```
/etc/init.d/cmdaemon start
tail -f /var/log/cmdaemon.log
```

The number of logging messages can be configured within the cmdaemon startscript (`/etc/init.d/cmdaemon`) by changing the variable “debug”.

With `/etc/init.d/cmdaemon status` one can get all running CMDaemon processes.

5.3.6.3.3 Update

To update the CMDaemon download the newest version of the dma-cmdaemon-yy.mm.dd.tar.gz file, reinstall and restart the CMDaemon like described above.

5.3.6.3.4 Uninstall

Simply execute `cmd_uninstall` (located in the same directory from where you extracted dma-cmdaemon-yy.mm.dd.tar.gz) to delete the files related to CMDaemon.

5.3.7 Users Guide

5.3.7.1 Description

After the successful installation you can start the Configuration GUI. Start your webbrowser, enable Style Sheets, Java and Javascript and insert the IP address and the subdirectory of the webserver in the address field of the browser (`http://servername:8089/aquiladma/`). After authentication you will see the graphical user interface for the AQUILA measurement database. There is a standard administrator with username and password “aquila”. Change the password as soon as possible.

The interface is divided in three groups (Output, Configuration and Administration). The output interface shows the results and the information which is generated by the measurement system. In the configuration part the measurement is defined. The administration part manages the users which handle the measurement system.

The configuration and administration of the measurement follows the way of the navigation tree in the left side of the browser window from top to bottom.

To configure a measurement you need a user first. This user must generate a test, which contains parameters of the measurement. Next step is the definition of hops and routers as there are in your network. You have to use exactly the same parameters (IP-Address, etc.) you’re using at your network equipment.

Then you have to define distributions and traffic which describes how your measurement packets behave in the testscenario. It is possible to use preconfigured traffic parameters. This parameters will be generated when you use the sql script as described above.

The central page is the flow page. With a new flow you connect all the parameters which you have defined before to a measurement flow. If you are ready you can start this flow. The master looks in the database and finds your new flow. The master request the chosen measurement clients to generate the chosen traffic packets and send them through the network. If this flow runs, you can monitor the results online or wait until the flow stopped and take a look at the results. A lot of pages help you to manage the database with all the trimmings.

5.3.7.2 Administration

To configure a measurement you need a user first. Click “Administration” in the navigation tree. If you check the box “New user” this will lead you to a mask, where you can generate a new user. The accesslevel "Administrator" allows you to change user parameters and every parameter in the database. With accesslevel "User" you can only work with your own flows and tests. Non used user can be deleted from the database. If an user has a test, you have to delete the tests first, and maybe other parameters, which belong to this test.

5.3.7.3 Configuration

There is an easy way in a few steps to configure and start a measurement test. Change to the menu "Configuration" and fill in the fields. When submitting the first page the system guides you through the rest of the configuration of all necessary parts of a test. For the configuration it is also necessary to configure at least one user and more than two hops for the measurement. Afterwards the mask switch automatically to start or stop tests. This part is not necessary for the SPU tool, because these tools are controlled by time.

5.3.7.4 Advanced Config

This mask handles the different parameters for your measurement system. You can generate new parameters, modify non used parameters and delete non used parameters.

Also it is possible to make copies, which is an easy way ore than one parameter with nearly the same values. Every field of the masks is described with a help text at the end of the mask.

5.3.7.5 Start / Stop

This navigation point has two options. You can start or stop all flows within a test, or start or stop one or more tests. The first point is very useful if a test handles a lot of flows. The start/stop masks are not necessary if you use the tools from ELI or SPU.

5.3.7.6 Monitor

This navigation option has some useful masks to get more information about the measurement system. The hopstatus for example shows the status of the synchronisation which is helpful to know if the timebase is correct and if you will get valid data at the end of your measurement

Events are generated by the measurement tools. There are also helpful for error detection.

The management info shows the information generated by the ELI router monitoring.

The flowmonitor is an online monitoring of some dedicated measurement flows. Within an time interval, which can be defined, the results of the measurement are presented in a graph. Only measurement flows that produce aggregated results can be viewed with the flow monitor!

The flow monitor (available at Monitor → Flowmonitor) lists all running flows, which have an aggregation time specified. With the flow monitor it is possible to view the aggregated results as a graph, which was produced during the runtime of the flow. This is especially useful for probing flows. The aggregation time of results can be selected during the flow configuration and specifies the constant intervals, where the receiving daemons are reporting a result aggregate over the last aggregation period.

From the list of running flows, the monitor can be either configured within the current frame (integrated) or within an extra window (popup).

The flow monitor automatically updates the graph in specified intervals, which defaults to the specified aggregation period.

5.3.7.7 Results

The results of the measurements can be selected browsed either grouped by tests or single flows can be searched.

- **DTA Tests (probing):** Only the tests performed with the DTA T-Nova Toolset of the current user will be listed.
- **SPU Tests (appl-like):** Only the tests performed with the SPU CMToolset of the current user will be listed.
- **All Tests:** All tests of the current user will be listed.
- **Running Tests:** Only running tests of the current user will be listed.
- **Error Tests:** Only tests with at least one error flow of the current user will be listed
- **Search Flows:** Single Flows can be selected by choosing different parameters. More than one parameter can be specified by selecting ‘new criteria’. All criteria will be AND-connected. The following figure describes the hierarchy of the search function:

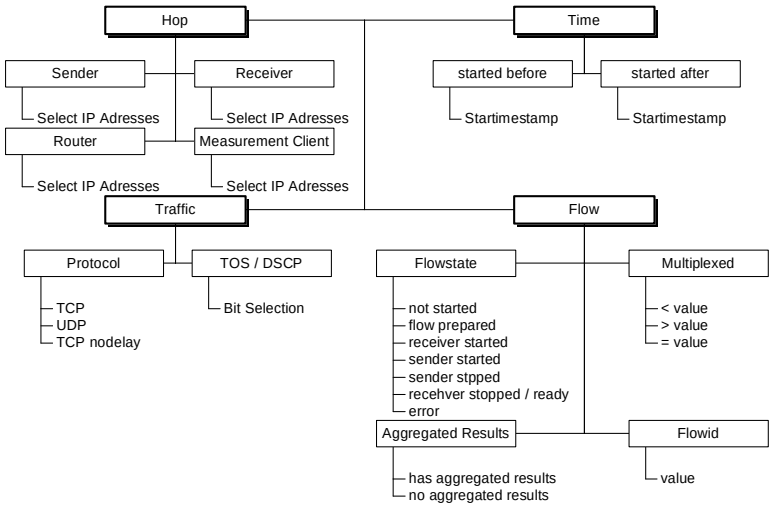


Figure 5-5 Structure of Flow Search Function

Using one of the functions Results → DTA / SPU / All / Running / Error Tests, the selected tests of the current user will be displayed in a table. From here the single flows of the tests can be viewed by using the '+' button at a specific test, which lists all the containing flows. With a further '+' the according overall results to the flow are shown. Now links are provided to view the graphs or the data lists (as comma separated value list) of the flows. Also statistics can be calculated and viewed, if raw data is available for this flow.

In principle the function Results → Search Flows shows the same results as above, with the difference, that the flows are not grouped by their according tests, but can be selected individually. The list of matching flows is then displayed in a table. Each flow can be selected by using the checkbox. Afterwards the following functions can be applied:

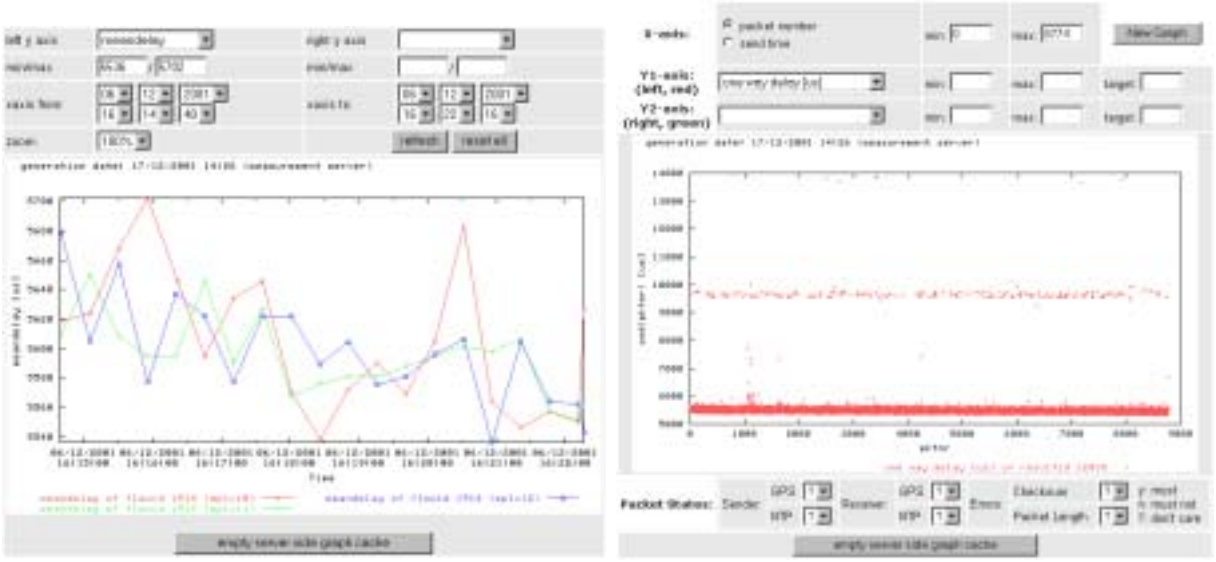
- **Zoom to selected:** With this function, the same table is displayed but only the selected flows are listed.
- **Show selected:** This function displays the aggregated results of the selected flows as graph.
- **Delete selected:** This function deletes the selected flows with all according results and the path information.

Three types of measurement results are produced by the active measurement part of the measurement system:

- **Overall results** are produced to have a final result for the measurement flow. A flow has usually one overall result, except the flow has been specified as a multiplexed flow. Multiplexed flows have several overall results, one for each single flow.
- **Aggregated results** are produced to get an overview about the measurement results within specific time periods.

- **Raw data results** are produced to get a detailed view of the measurement results. Raw data means, that the sending and receiving timestamp as well as the packet size and a packet state are reported for every single measurement packet.

E.g. Figure 5-6 shows two graphs of the delay of a measurement flow. The measurement flow consisted of three multiplexed UDP flows, each with mean a sending rate about 150 kbps. Figure 5-6 (a) displays the mean delay of the aggregated data of the three UDP flows. Figure 5-6 (b) displays the according raw data (per packet) of one of the three multiplexed UDP flows.



(a) aggregated data result display (b) raw data result display

Figure 5-6: Screenshot of DMA results

The results are only produced if they are requested by the user.

For overall and aggregated results the reported results are:

- Number of Packets in this Aggregate
- Throughput
- Packet Loss
- Packet Loss Patterns (maximum loss burst, maximum loss distance, number of loss periods)
- Delay (mean delay, maximum delay, minimum delay)
- Delay variation (mean delay variation, maximum delay variation)

Except by using the flow monitor the results can be viewed when the measurement flows are ready. In general, the results of the active measurements can be browsed grouped by tests or by searching single flows.

5.3.7.8 DTA measurement agent controller (MACON)

The DTA MACON is an easy to use GUI for a maximum of 10 DTA measurement agents. It is possible to set up a fully mashed measure with a maximum of 5 traffic classes. The results are shown online in separate windows and they can be saved in csv files (comma separated values).

Figure 5-7: DTA MACON

1. IP addresses of all PC where the DTA measurement agents are running. Check the field if you what the use the agent form a measurement. After starting a test, the MA-number has the colour “green” if the agent is reachable, else the colour is “red”. The “status” shows the synchronisation of the PC. The status must be “NTP” and the colour can be red (NTP offset/jitter > 2ms; not good enough for measurement), yellow (NTP offset/jitter < 2ms) or green (NTP offset/jitter <1ms).

2. Configuration of the traffic classes. The checkbox enables this traffic class for the next measurement. You can adjust the ToS-Byte, the distribution and the parameters for every traffic class.

3. Switch the windows (6, 7, 8, 9, 10) on/off and choose which measurement flow is displayed in the table 5 and the graphics 7, 8, 9, 10.

4. Enable the writing of results in csv-files.

5. Result table of one measurement flow and the opposite direction.

6. Window for all agent messages and errors.

7. Graphic of the aggregated one-way delay.

8. Graphic of the aggregated delay variation.

9. Graphic of the throughput.

10. Graphic of the raw data (one-way delay of every measurement packet).

To start a measurement press the start/stop button on top of the GUI. All given value were read and the agents are initialised with them.

5.3.7.8.1 DTA measurement agent monitor (MAMON)

The DTA MAMON is a monitor for the DTA measurement agents. The program checks the status of an agent and shows all running flows.

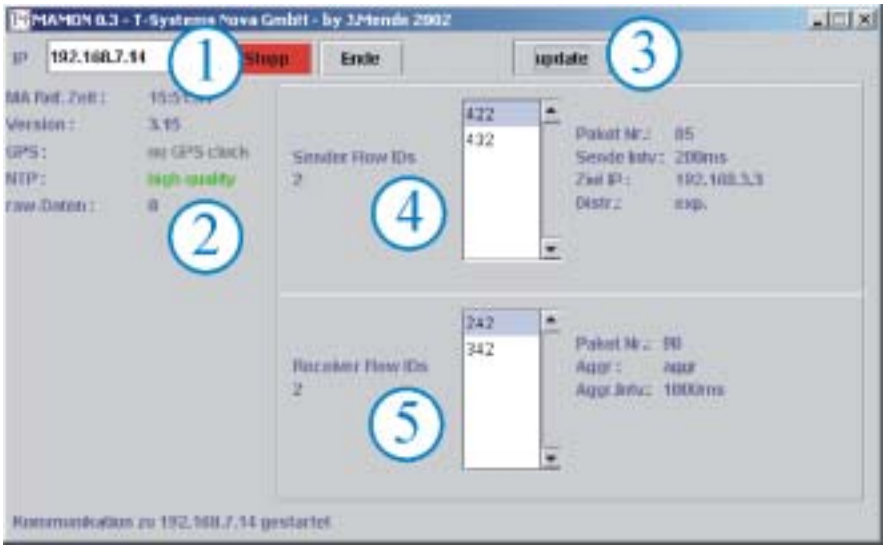


Figure 5-8: DTA MAMON

1. Enter the IP address of a measurement agent and press the start/stop button

2. In this section is the time of the clock, the agent version number an the synchronisation status displayed
3. If you press the update button all running flows are listed in the panels 4 +5
4. List of all running senders on this agent.
5. List of all running receivers on this agent.

5.4 Manag layer

5.4.1 QMTool

5.4.1.1 Required software and installation procedure

All classes belonging to the QMTool are located in the aquila.jar library. In particular, these classes are located in two separate directories within the aquila.jar file due to the fact that QMTool has been integrated with the XMLOperator for the needs of configuration (see 5.4.1.2). The pure QMTool class files reside under the package "aquila/manag/qmtool/" whereas the XMLOperator class files reside under the package "org/xmloperator/". Accordingly, the package structure used for QMTool files is the following:

```
aquila/
    manag/
        qmtool/
org/
    xmloperator/
```

Additionally, the following external libraries are required for QMTool:

JAXP:	**xerces.jar, xalan.jar**
JAXB:	**jaxb-rt-1.0-ea.jar, jaxb-xjc-1.0-ea.jar**
JBCL:	**jbcl3.1.jar**

Moreover, the “xmloperator.xin” file is needed for QMTool and it should be placed under its *current* directory.

The QMTool has one difference with respect to the other aquila components that has resulted from its integration with the XMLOperator code. The *current* directory for starting qmtool is not the $project directory, as it is the case for the other components, but one directory within this hierarchy. We have chosen the $project/xml for this purpose.

QMTool uses the installation and compilation procedure performed for all Aquila components as described in 5.1.1.2.

For running QMTool, the following steps should be taken:

```
cd $project/xml
java –cl ../lib/aquila.jar
 –Daquila.util.main.propertyfile=as<asnumber>.rc \
 org.xmloperator.Tool
```

To be noted that the NameServer has to be running before starting QMTool.

5.4.1.2 User guide

The QMTool is used for performing 3 tasks, i.e. configuration, failure detection and monitoring. In the sequel, a guide for using these features will be given.

Configuration

The configuration of the Aquila components comprises the maintenance of their corresponding configuration XML files at the LDAP database. For that purpose, QMTool provides a graphical interface (XMLOperator is used) for enabling the user to view the configuration information of each component and to perform changes on this information which is finally stored at the LDAP.

However, in the case of the components that belong to the RCL layer (Rpools, ACAs, EATs, Proxies, BGRPAgents), the configuration is eased through the use of an additional graphical interface where the visualization of these components is performed.

Therefore, for configuring the aforementioned components, the following steps should be performed:

- Select the menu “Database/Import”. This will have as a result the visualization of the resource pools, ACAs, EATs, Proxies and BGRP Agents as it can be seen in Figure 5-9.

- By right-clicking on each component to be configured (actually to view the existing configuration information), a menu appears. Select “Properties…” from this menu. The XMLOperator graphical interface (see Figure 5-10) appears where the configuration information of this component can be viewed.

- If needed, perform the appropriate changes on this configuration information. Then, in order to update the configuration of this component at the LDAP database, select “File/Save&Store to database…”.

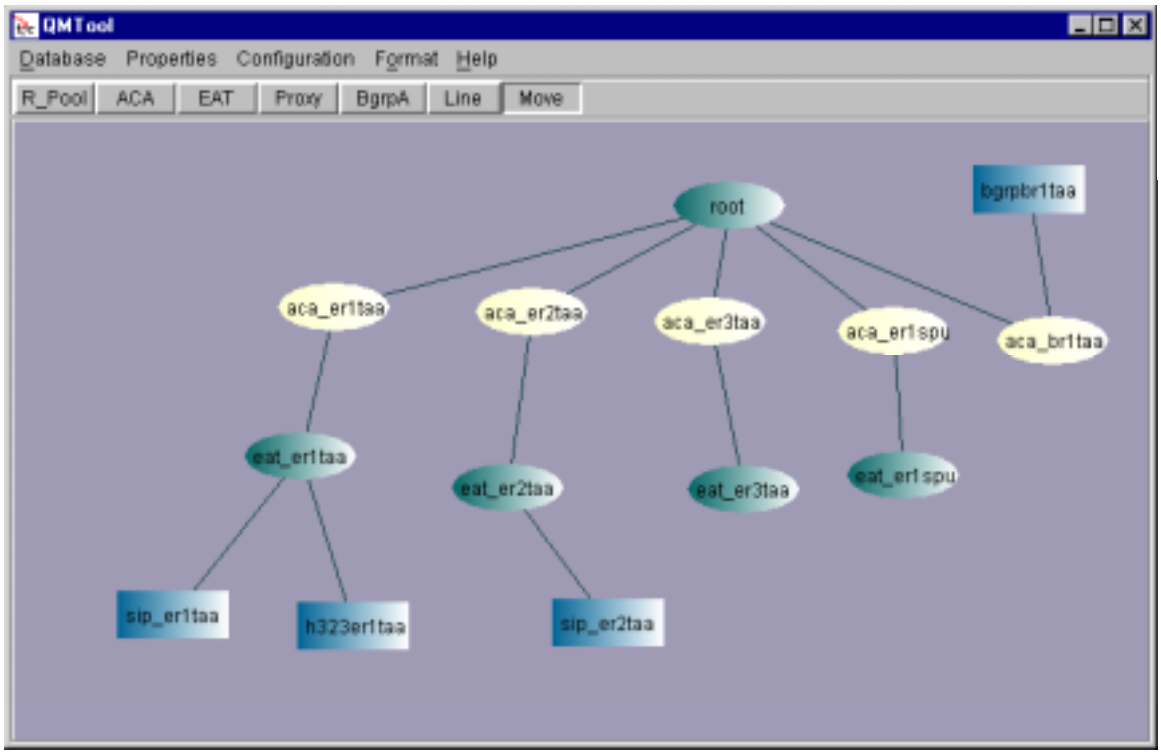


Figure 5-9: Visualization of RCL components

Moreover, other type of configuration information requires different steps. This includes the configuration of the KeepAlive mechanism, the Tracing mechanism, the RcRules, the Prioritised Signalling, the Routers, the IOS version of the routers, the Services, the Traffic Classes, the Users, the Application Profiles, the Service Component Profiles and the GWKS. The following steps should be performed:

- Select the menu “Configuration” and from there, the component that you want to configure. The names of the menus obviate their corresponding component.

- For some of the components, the XMLOperator graphical interface will appear where the configuration information can be viewed. For others (e.g. users), a dialog box (see Figure 5-11) appears that enables the addition, removal of components, i.e. addition/removal of users in our example. For viewing the configuration information for each component (user), select the component from the table and then press the “Edit…” button. The XMLOperator appears again with the configuration information of the selected component. For adding a component, press the “Add…” button. Another dialog box appears where you have to enter the name of the component and then press “Save”. Then another dialog box appears where one drop down list and a text field can be viewed (see Figure 5-12). At the “Document type”, you select the dtd type of the component (remember, its component is characterized by its own dtd type). At the “File name” text field, you put the complete pathname of the xml configuration file. You can either select an existing xml configuration file for this component (“Browse…” button) or simply put an xml pathname that does not exit. In the latter case, by pressing “Ok”, the XMLOperator interface will not have the configuration information of the component but the user has to create it.

- At the time you can view and change the configuration information via the XMLOperator, select “File/Save&Store to database…” in order to update the data at the LDAP database.

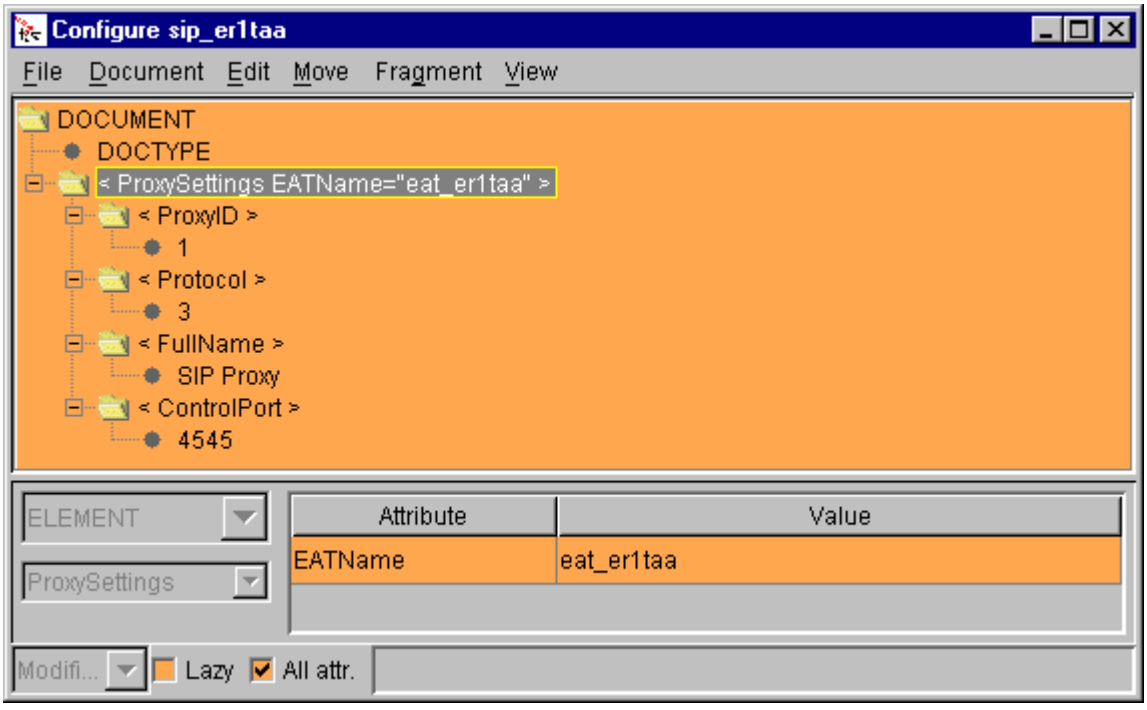


Figure 5-10: XMLOperator graphical interface for configuration of components

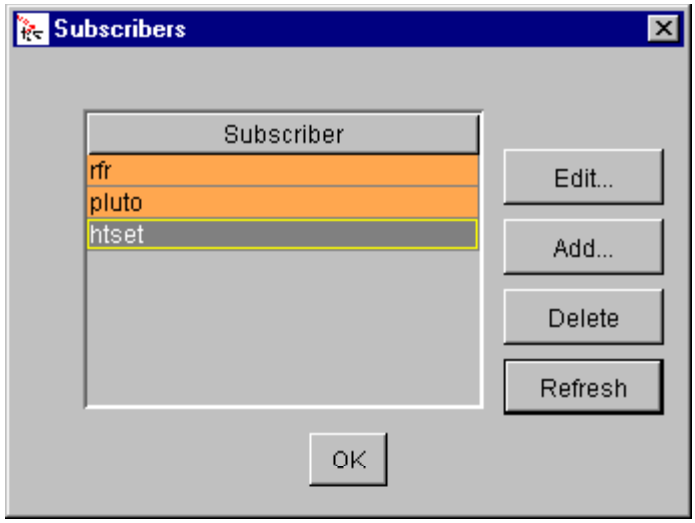


Figure 5-11: Dialog Box for the configuration of certain components

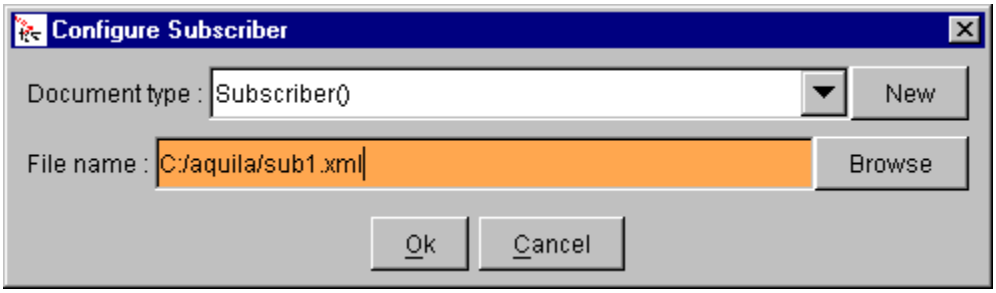


Figure 5-12: Dialog Box for defining the dtd type of the configuration component

Failure Detection

Failure detection is possible for the components of the Resource Control Layer and in particular the resource pools, the ACAs, the EATs and the BGRP Agents. The following steps should be performed.

- In the visualization area (see Figure 5-9), right click on the component for which you want to enable the failure detection mechanism. From the menu that it appears, select

"Failure Detection" and the mechanism is automatically enabled. In case something goes wrong, a dialog box appears in order to inform you for the error.

In the case that the mechanism detects a failure in the component, the latter is turned to color red. At that time, the mechanism is automatically disabled and therefore the user has to enable it again if he wishes to continue detecting this component.

Monitoring

The monitoring functionality is possible for the resource pools and the BGRP Agents. We will describe the monitoring of each type separately.

When we refer to the monitoring of ***resource pools***, we refer to the monitoring of their resources with respect to the several Traffic Classes. Resource Pools' Monitoring necessitates the following steps:

- In the visualization area (see Figure 5-9), right click on the resource pool that you want to monitor and select "Monitoring". From the menu that it appears (all Traffic Classes are listed), select the traffic class that you want to monitor. Then, the monitoring window appears (see Figure 5-13) unless the user is notified of a possible error.

The monitoring window, as it can be seen from its title, corresponds to a specific resource pool and a specific Traffic Class. The "Y_max" and "DY" values refer to the maximum value of the Y_Axis and to the step on the same axis respectively. The "Y_max" value as well as the time interval of monitoring can be set from the "Properties/Monitoring…" menu (see Figure 5-14). The selected values will influence the monitoring (monitoring windows) that will start from that time on.

Below the graphical area of the monitoring window, the current monitored values can be seen in text format. These correspond to the Maximum, Reserved and Total resources that are handled by the resource pool algorithm. The current values are the ones depicted at the graphical area whereas the max values correspond to the maximum value that the Maximum, Reserved and Total resources have reached.

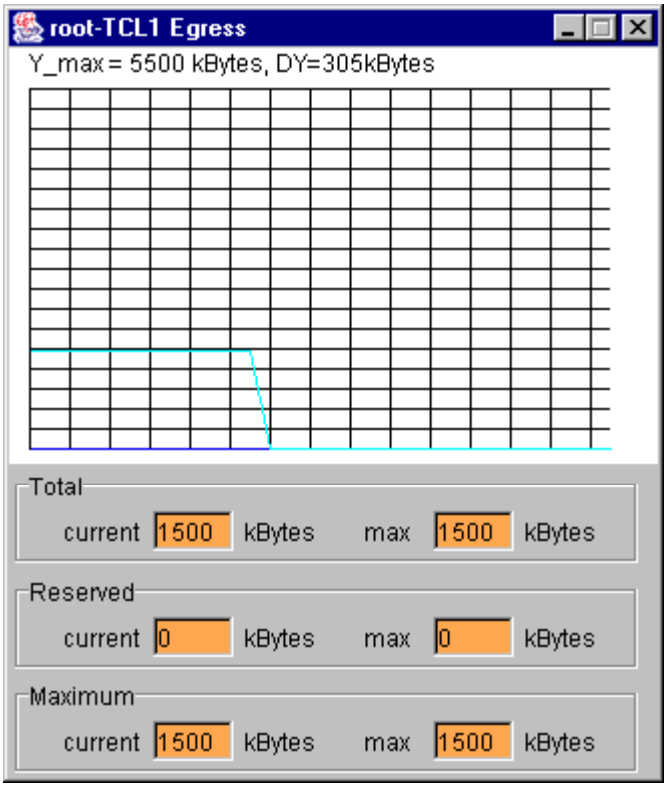


Figure 5-13: Monitoring window of resource pools for a selected Traffic Class

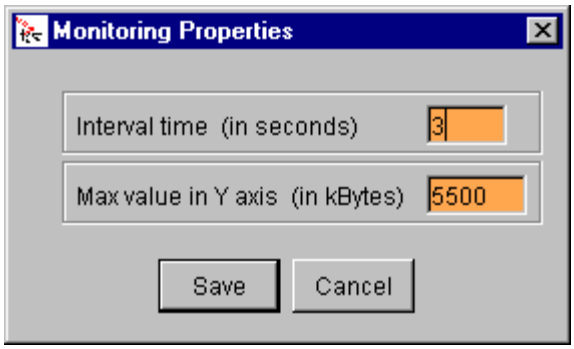


Figure 5-14: Dialog Box for changing the monitoring properties

When we refer to the monitoring of ***BGRP Agents***, we refer to the monitoring of their resources with respect to the several Globally Well Known Services. BGRP Agents' Monitoring necessitates the following steps:

- In the visualization area (see Figure 5-9), right click on the Bgrp Agent that you want to monitor and select “Monitoring…”. The frame that is depicted in Figure 5-15 will appear unless the user is notified of a possible error.

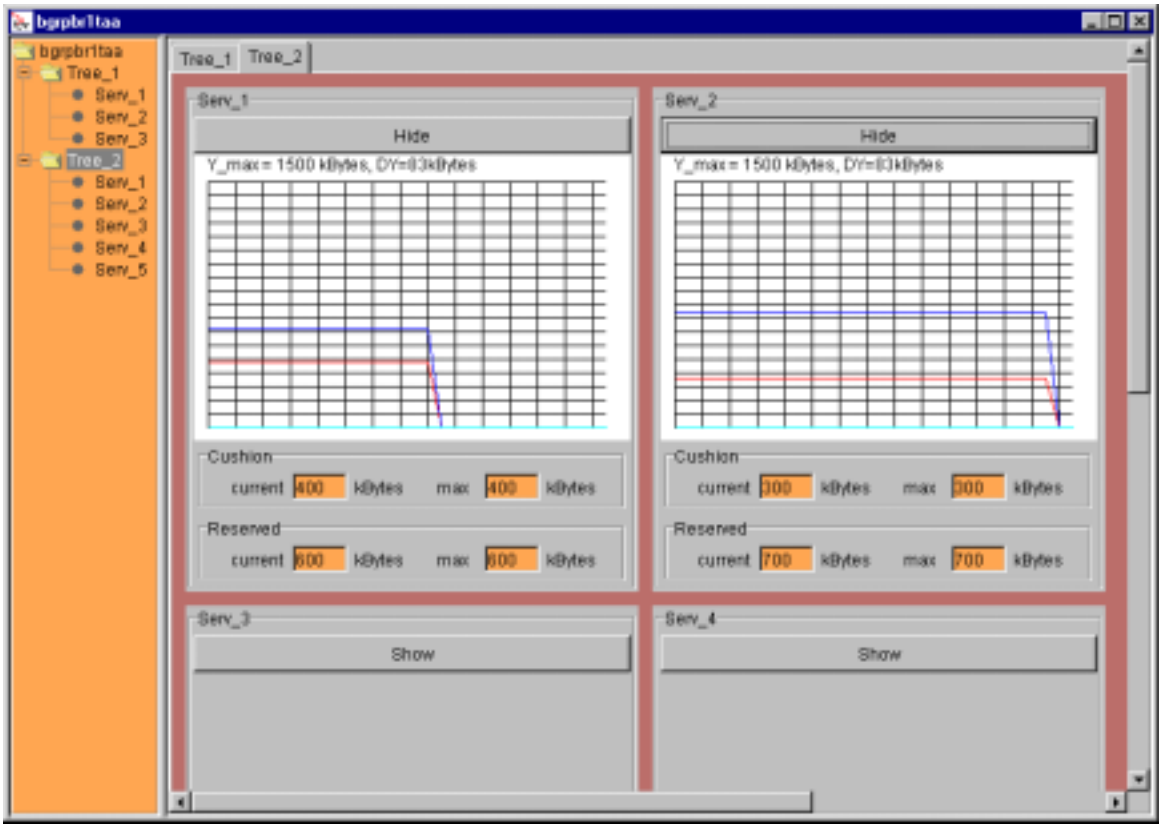


Figure 5-15: Monitoring window for BGRP Agents

The illustrated monitoring window corresponds to one BGRP Agent and, as it can be seen, it monitors the GWKS’ resources in each sink tree of the agent. Here again, the areas “Cushion” and “Reserved” correspond to the resources handled by the Bgrp Agents. There is the possibility to hide/show the monitoring panel devoted to one service by pressing the “Hide”/”Show” buttons respectively. In addition, the “Y_max” value can be set from the same dialog box that has been described above for the needs of the resource pool’s monitoring. This value is the same for both the bgrp agent and the rpool monitoring.

5.5 IDOM layer

5.5.1 BGRP

5.5.1.1 Required software (packages, xml files) and installation procedure

All classes belonging to the BGRP agent are located in the aquila.jar library.

For each BGRP agent, a configuration entry in the LDAP database is necessary (see next chapter).

5.5.1.2 User guide

To set up a BGRP agent, you have to collect information about the agent and its neighbours. This information will be stored in the LDAP database and used by the agent during initialisation and operation.

- A BGRP agent is always associated to an ACA and a border router. To establish the connection to the ACA, the name of the ACA must be known.
- To release resources, the BGRP agent uses an algorithm from the util.rcrule package. An identifier for the proper algorithm must be specified. As the preferred way for resource release is a delayed resource release mechanism, the identifier of this algorithm should be specified.
- To identify the border router, an IP address of the router must be known. Preferably, this should be the (an) address at the inner side of the border router.
- A BGRP agent belongs to a domain. The AS number of this domain must be known.
- A REFRESH interval can be specified, if it is necessary to change the default of 300 sec.

Similar to BGP, also the BGRP agent needs information about all neighbours, both, within the same domain or in a neighboured domain. For each neighbour, the following information must be collected:

- Name of peering border router. This name points to an entry of the bgrp.neighb package. See there for details.
- List of addresses of peering border router. The first address should be the address of the loopback interface. More addresses might be listed, if BGP NEXT_HOP may return other addresses.
- AS number of peering domain.

Typically ALL other BGRP agents of the same domain are neighbours to a BGRP agent, resulting in a full mesh of BGRP neighbours.

To each neighbour (internal and external!) a SLA has to be specified, which specifies a bandwidth limit and possible packet remarking properties for each supported GWKS. So collect the following information for GWKS1 and GWKS3 (the two, which will be supported in the second trial):

- Maximum incoming bandwidth
- Maximum outgoing bandwidth
- Incoming DSCP

The first SLS entry in a PerServiceSla is always for the incoming direction, the second specifies the outgoing direction. If you want to disable SLS checking for internal links, specify a very high bandwidth limit.

An example XML file could look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE BgrpConfig SYSTEM "../dtd/bgrp.dtd">
<BgrpConfig as="20" domainRM="aca-20-1" ipAddr="192.168.20.1" rcRule="0">
  <NextHop as="10" ipAddr="192.168.10.2 192.168.100.1" name="bgrp-10-2">
    <Sla>
      <PerServiceSla gwks="GWKS1">
        <Sls dscp="8" maxBw="1000000"></Sls>
        <Sls maxBw="1000000"></Sls>
      </PerServiceSla>
      <PerServiceSla gwks="GWKS3">
        <Sls dscp="32" maxBw="1000000"></Sls>
        <Sls maxBw="1000000"></Sls>
      </PerServiceSla>
    </Sla>
  </NextHop>
  <NextHop as="20" ipAddr="192.168.20.2 192.168.200.1" name="bgrp-20-2">
    .. data for other neighbour
  </NextHop>
</BgrpConfig>
```

Store the XML file in the LDAP database under the following DN:

```
cn=<bgrpname>,cn=bgrp,cn=idom
```

and start the BGRP agent with the following command:

```
cd $project
java –cl lib/aquila.jar
 –Daquila.util.main.propertyfile=support/rc/as<asnumber>.rc \
 aquila.idom.bgrp.BgrpMain <bgrpname>
```

5.5.2 GWKS

5.5.2.1 Required software (packages, xml files) and installation procedure

All classes belonging to the GWKS package are located in the aquila.jar library.

The GWKS package has just descriptive purposes to specify the name of the available GWKS. A single configuration entry in the LDAP database is necessary to specify all GWKS. The following XML file can be used:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GlobalServices SYSTEM "../dtd/gwks.dtd">
<GlobalServices>
  <Gwks id="GWKS1"></Gwks>
  <Gwks id="GWKS3"></Gwks>
</GlobalServices>
```

5.5.2.2 User guide

Store the XML file in the LDAP database under the following DN:

```
cn=gwks,cn=idom
```

5.5.3 Neighb

5.5.3.1 Required software (packages, xml files) and installation procedure

All classes belonging to the neighb package are located in the aquila.jar library.

For each possible neighbour in a domain, a configuration entry in the LDAP database is necessary (see next chapter). Possible neighbours are

- All BGRP agents in the own domain
- All peering BGRP agents in neighbour domains

5.5.3.2 User guide

For each BGRP agent in the own domain and each peering BGRP agent in neighbour domains collect the following information:

- IP address of host running the BGRP agent (not IP address of router!)
- Port number identifying the BGRP agent on this host. This number can be chosen arbitrarily and can be used to distinguish several BGRP agents running on the same host.
- Additionally, the timeout values for the neighbour detection can be specified, if the default values are not sufficient.

For each neighbour, create an XML file containing this information and store it under the following DN in the LDAP database:

```
cn=<bgrpname>,cn=neighb,cn=idom
```

An example XML file could look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GlobalServices SYSTEM "../dtd/neighb.dtd">
<Neighbour ipAddr="192.168.2.3" port="2002"></Neighbour>
```

5.6 Utilities package

5.6.1 RcRule

5.6.1.1 Required software (packages, xml files) and installation procedure

All classes belonging to the rcrule package are located in the aquila.jar library.

A configuration entry in the LDAP database is necessary for the rcrule package, where the different xml files will be stored, one per algorithm identifier.

5.6.1.2 User guide

The rcrule package aims at defining the algorithms necessary by the BGRP agent, the RCA agent and the ACA agent. There is the possibility to define, as many xml files are necessary for those three components.

- Concerning the RCA agent, one xml file can be written for each Resource Pool, in case a different set of algorithms is used or a different parameter configuration is required.
- For each BGRP agent, one xml file is defined, where the parameters for the delayed release algorithm are initialised.
- Finally, each ACA agent also requires one xml file.
- Those xml files should be named after the identifier determined in the xml files of the corresponding components.
- Moreover, in each xml file is defined the class name of the actual set of algorithms used by each component.

Depending on the algorithms, two structures should be initialised:

- The LeakyShareSetting structure, where the parameters for the delayed resource release or the counter based algorithm are initialised:
    - counter
    - release block size
    - release period
- The ResourceRequestSetting structure, where the parameters for the calculation of the increased assigned resources are initialised:
    - Amax

- Amed
- Amin
- wHigh
- wLow

In the following examples, one xml file named “DelayedRelease” is defined for a BGRP agent, and one for a resource pool named “Pool1”:

DelayedRelease xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Algorithm SYSTEM "rcrule.dtd">
<Algorithm RcRuleSet="RcRuleSet3">
  <LeakyShareSetting blockSize="50" releasePeriod="120">
  </LeakyShareSetting>
</Algorithm>
```

Pool1 xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Algorithm SYSTEM "rcrule.dtd">
<Algorithm RcRuleSet="RcRuleSet1">
  <LeakyShareSetting blockSize="50" counter="10">
  </LeakyShareSetting>
  <ResourceRequestSetting Amax="5" Amed="3" Amin="1" wHigh="0.8" wLow="0.2">
  </ResourceRequestSetting>
</Algorithm>
```

Store each XML file in the LDAP database under the following DN:

```
cn=<rcrulename>,cn=rcrule,cn=util
```

5.6.2 Router

5.6.2.1 Required software (packages, xml files) and installation procedure

All classes belonging to the router package are located in the aquila.jar library.

For each router (edge router, core router, border router), a configuration entry in the LDAP database is necessary.

5.6.2.2 User guide

The router package configuration mainly serves two aspects:

- Configuration of the communication to the router
- Configuration of the QoS features of the router itself

In the following, a sample configuration of a border router is shown. We first focus on the model, message and transport layer of the router package:

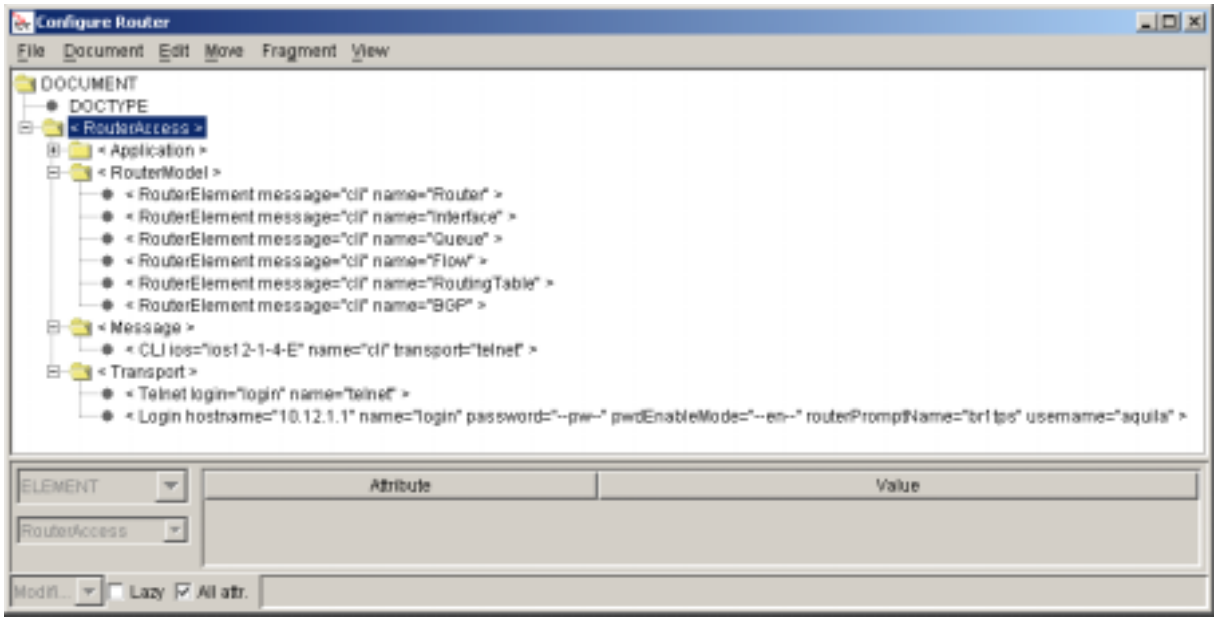


Figure 5-16: Configuration of model, message and transport layer

The snapshot above shows the following configuration items from bottom to top:

- a Login, specifying the host, username and passwords
- a Telnet session referring to this login. Note: advanced configurations could use multiple telnet sessions for parallel processing of commands.
- a CLI Message layer configuration specifying the IOS version of the router and referring to a telnet session.
- configuration of six types of router elements referring to a message layer entity. Note, that each router configuration MUST contain six router element entries of the above-shown types!

The application layer always contains the configuration for the six applications RouterConfig, RouteLookup, FlowManager, BGPSearcher, MBACMonAgent and DMAMonAgent:

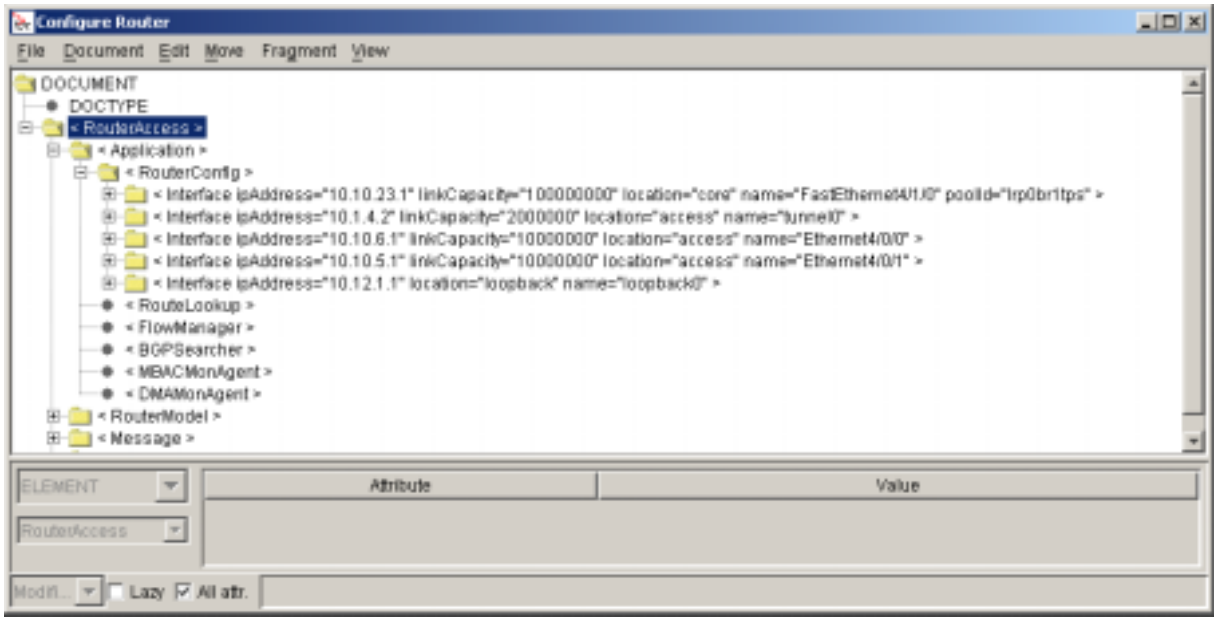


Figure 5-17: Configuration of the application layer

The configuration of most of the applications is pretty straight-forward (there is just the possibility to specify an alternate class name for the application). The RouterConfig application however contains the QoS configuration of all interfaces.

As you can see from the snapshot above, three types of interfaces have to be configured:

- core interfaces for all type of routers
- access interfaces for edge and border routers
- a loopback interface for edge and border routers

The amount of details varies for each type of interface, as shown below:

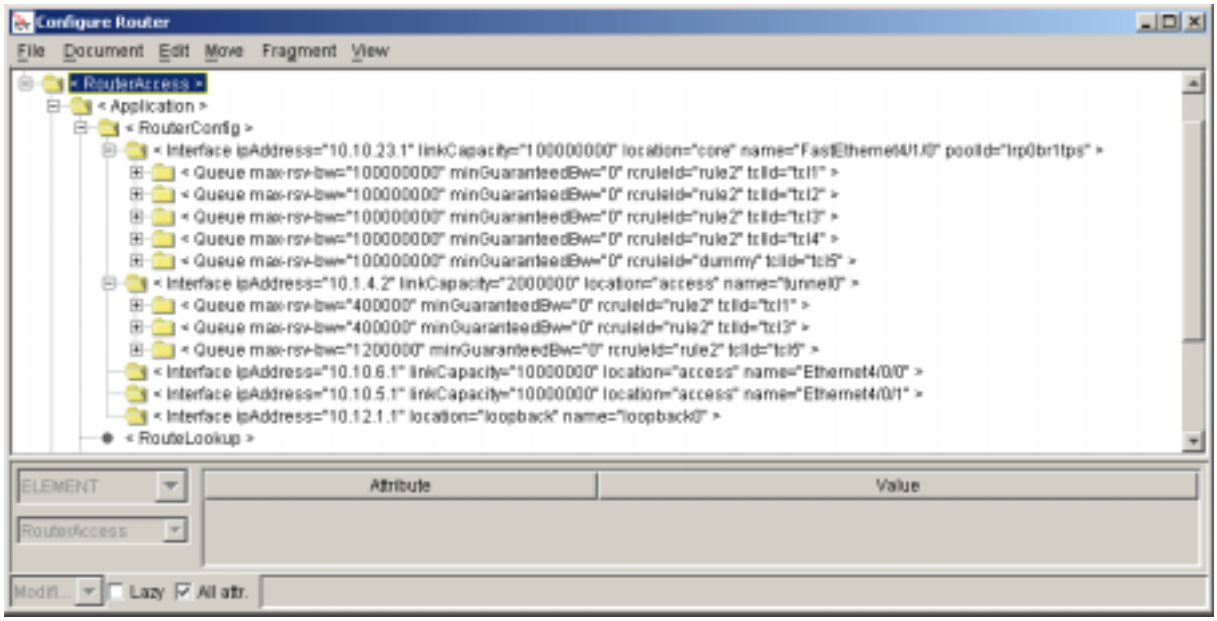


Figure 5-18: Configuration of the interfaces

For a core interface, five queues for the traffic classes tcl1 to tcl5 have to be configured. A pool ID must be specified to connect the admission control to the proper RCL resource pool.

For an access interface leading to another domain, only queues for tcl1, tcl3 and tcl5 have to be configured.

It depends on the actual properties of an access interface, whether queuing should be configured there as well. In the snapshot shown above, no queuing was configured, as these interfaces have a rather high bit rate and are not subject to congestion.

The loopback interface has just to be configured with its name and IP address.

The following snapshot shows more details on queue configuration for a core interface:

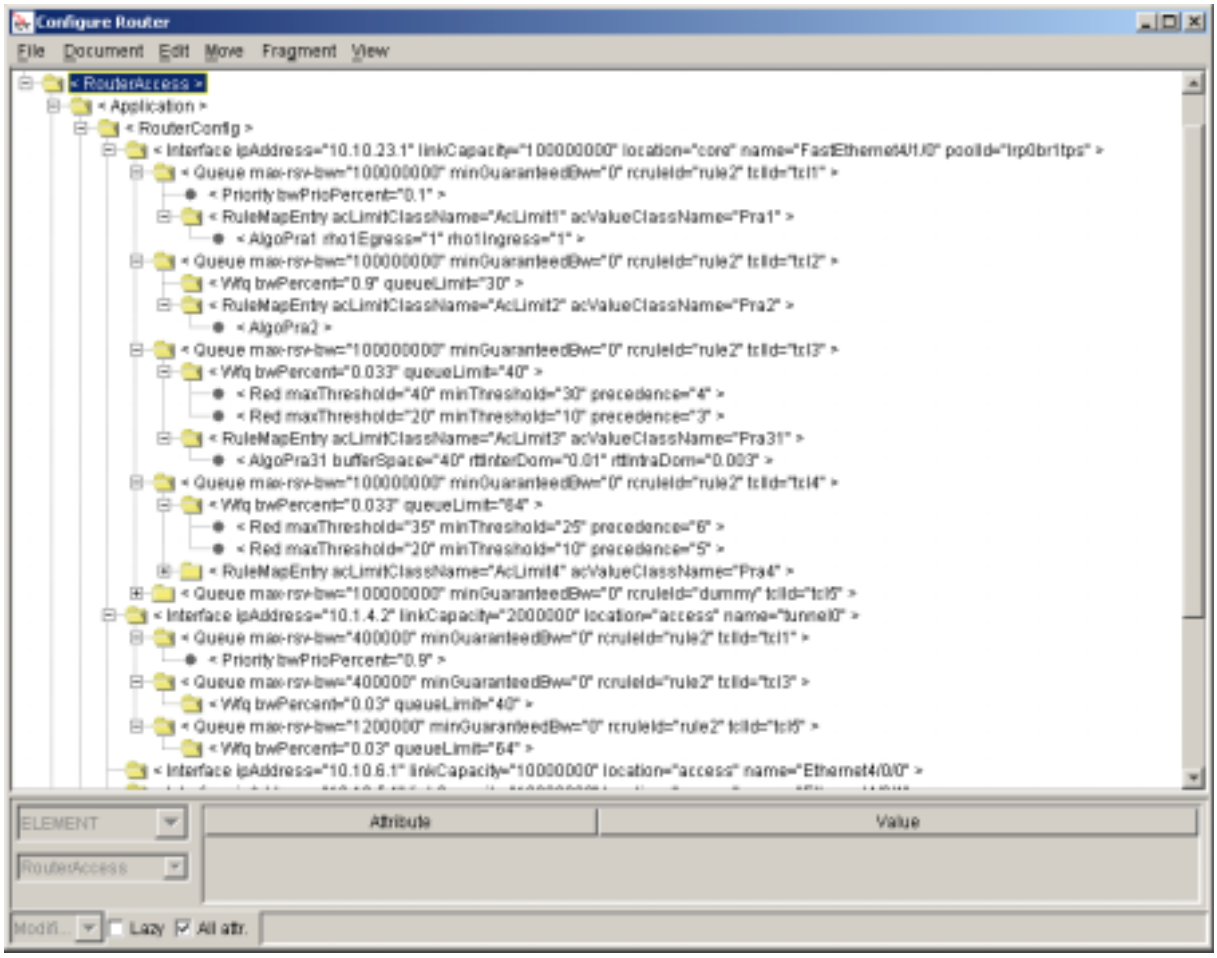


Figure 5-19: Detailed configuration of core interface

The queue for TCL1 is configured as a Priority queue. The rule map entry points to the Limit and Value classes for the TCL1 algorithms.

The queue for TCL2 is configured for weighted fair queuing.

The queue for TCL3 is configured for weighted fair queuing. For a core interface, two RED droppers are attached to this queue. No RED droppers are used, if the queue is on an inter-domain link.

The configuration of the queue for TCL4 is similar to TCL3.

Details of queue configuration and the meaning of the parameters can be extracted from [D1302].

6 Results and experiences

For the integration one meeting was held. It took place in Vienna, which hosts one of the trial sites of the AQUILA project. The integration continued the work of the pre-integration phase, by testing the RCL functionality over a real topology. Also, the Measurement Tools were integrated into the AQUILA network. Finally, the resulting prototypes were installed to the other two trials sites, in Warsaw and Helsinki.

The outcome of the integration meetings is considered by all partners to be very successful and that the objectives of the integration were fulfilled.

The success of the AQUILA integration was the result of good co-operative work. This is the major factor for the success of such a complex task. It should be noted here that the integration of the second trial required the interconnection of the two trial sites and therefore external assistance was needed. The AQUILA consortium would like to thank the people operating the GEANT network (SEQUIN). Many experiences were gained from the first trial integration which were applied to the second trial. The most important points are outlined in the following paragraphs.

It is of critical importance to plan carefully the targets of the integration, the time plan to be followed, the network topology to be used and the tests to be performed. The maintenance of a software library will also ensure the integrity and consistency of the produced software components.

For the planning of integration meetings, it is very important to have all the equipment available and all the necessary software already installed. In the case of multiple trial sites the remote configuration is a good approach to minimize the number of the integration meetings. Firstly, the integration takes place in the local site and after the successful setup the remote integration procedure can be started. However, enough time should be foreseen for this work, and also the interconnection between the sites should have be established along with the needed software/hardware equipment. Such an integration procedure assures that the same responsible people make all the appropriate configurations on all trial sites ensuring the consistency of the testbeds.

7 ABBREVIATIONS

ACA	Admission Control Agent
API	Application Programming Interface
BSP	Bucket Size for Peak Rate
BSS	Bucket Size for Sustainable Rate
BGP	Border Gateway Protocol
BGRP	Border Gateway Reservation Protocol
CAR	Committed Access Rate
CBQ	Class-Based Queuing
CBWFQ	Class-Based Weighted Fair Queuing
CLI	Command-Line Interface
COPS	Common Open Policy Service
CORBA	Common Object Request Broker Architecture
DiffServ	Differentiated Services
DSCP	DiffServ Code Point
DTA	T-Nova Systems
EAT	End-user Application Toolkit
EDAdapter	Edge Device Adapter
ELI	Elisa communications
GUI	Graphical User Interface
GPS	Global Positioning System
GWKS	Globally well known services
IDOM	Inter-domain Layer
IDL	Interface Definition Language

LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LLQ	Low Latency Queuing
MBAC	Measurement based admission control
PCBR	Premium Constant Bit Rate
PMC	Premium Mission Critical
PMM	Premium Multimedia
PR	Peak Rate
PVBR	Premium Variable Bit Rate
RCA	Resource Control Agent
RCL	Resource Control Layer
RED	Random Early Detection
RP	Resource pool
RPL	Resource pool leaf
RUT	Router Under Test
SLS	Service Level Specification
SPU	Salzburg Research
SR	Sustainable Rate
TAA	Telekom Austria
TCL	Traffic class
TPS	Polish Telecommunication Company
TC	Traffic Class
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Detection
WUT	Warsaw technical university

XML	Extensible Mark-up Language

8 REFERENCES

[D2203] IST-1999-10077-WP2.2-TUD-2203-PU-R/b0, User Guide for End-user Application Toolkit

[D1302] IST-1999-10077-WP1.3-COR-1302-PU-O/b0, Specification of traffic handling for the first trial

9 APPENDIX

9.1 Production script

```
#!/bin/csh -x
#
# set the CLASSPATH ...
#
set jdk=/usr/local/j2sdk1.4.0/bin
set aquila=/home/aquila/production
set src=$aquila
set classes=$aquila/classes
set support=$aquila/support
set external=/home/aquila/2ndTrialReleases/external
set lib=$aquila/lib
set classpath=$lib/aquila.jar':'$external/jta20.jar':'$external/crimson.jar': \
        '$external/xalan.jar':'$external/jaxp.jar':'$external/servlet.jar': \
        '$external/mm.mysql-2.0.11-bin.jar':'$external/jaxb-rt-1.0-ea.jar': \
        '$external/jbcl3.1.jar':'$external/xerces.jar
set qmtoolpath=$external/xerces.jar':'$classpath': \
        '$external/jaxb-xjc-1.0-ea.jar':'$external/jbcl3.1.jar

setenv CLASSPATH $qmtoolpath

cd $aquila

# clean up
echo Cleaning up...
rm -rf aquila classes org aquila.jar *.idl *.dtd *.xjs

# extract zip files
foreach fn ( *.zip )
    echo unzip $fn
        $jdk/jar xf $fn
end

echo Moving DTDs and XJSs to the dtd directory
cd $aquila
mv -f *.dtd *.xjs dtd

# compile IDL files
foreach fn ( *.idl )
    echo Compiling $fn
    $jdk/idlj -oldImplBase -fallTIE $fn
end

# compile DTD/XJS files
foreach fn ( dtd/*.dtd )
    echo Compiling $fn
    $jdk/java -classpath $external/jaxb-xjc-1.0-ea.jar com.sun.tools.xjc.Main \
        $fn dtd/`basename $fn .dtd`.xjs
end

foreach fn ( `grep -l "import aquila.*VPatcher" \
        aquila/*/*/*.java aquila/*/*/*/*.java` )
    echo "Correcting VPatcher problem in" $fn
    grep -v "import aquila.*VPatcher" $fn > $fn.tmp
    rm -f $fn.bak
    mv $fn $fn.bak
    mv $fn.tmp $fn
end

set fn=aquila/util/router/message/ParseGroup.java
```

```
echo "Correcting self-import problem in" $fn
grep -v "import aquila.util.router.message.ParseGroup" $fn > $fn.tmp
rm -f $fn.bak
mv $fn $fn.bak
mv $fn.tmp $fn

# compile JAVA files
echo Compiling JAVA files...
mkdir classes

$jdk/javac -classpath $classpathProduct -d classes \
        aquila/*/*/*.java aquila/*/*/*/*.java aquila/*/*/*/*/*.java \
        aquila/*/*/*/*/*/*.java org/xmloperator/*.java org/xmloperator/*/*.java \
        org/xmloperator/*/*/*.java org/xmloperator/*/*/*/*.java
$jdk/javac -classpath $classpathProduct -d classes java/net/*.java

echo Compile Gui
cd $aquila

$jdk/javac -classpath $classpathProduct -d gui/WEB-INF/classes \
        gui/WEB-INF/src/*.java

# create JAR file
echo Creating JAR archive...
cp -r $aquila/aquila/manag/qmtool/pics $classes
cd $classes

set aquilaDate=aquila-`date "+%y.%m.%d-%H.%M"`

$jdk/jar cf $lib/$aquilaDate.jar aquila org pics

cd $lib
chmod ug+rw aquila*.jar
chmod ug+rwx .
echo JAR file generated

#create the new upload-version
echo create the new upload-version
cd $aquila
cp classes/java/net/*.class java/net/
rm java/net/*.java

$jdk/jar cMf $aquilaDate.zip lib/$aquilaDate.jar gui xml dtd java

rm java/net/*.class

echo Cleaning up...
cd $aquila
rm -rf aquila classes/aquila classes/org org *.idl *.xjs

echo Finished
```