




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-3101-PU-R/b0
Contractual Date of Delivery to the CEC:	March 31, 2001
Actual Date of Delivery to the CEC:	March 29, 2001
Title of Deliverable:	First Trial Integration Report
Workpackage contributing to the Deliverable:	WP 3.1
Nature of the Deliverable:	R – Report
Editor:	Charilaos Tsetsekas (NTU)
Author(s):	Reinhard Frank, Martin Winter, Axel Hof (SAG), Falk Fünfstück, Anne Thomas (TUD), Wojciech Burakowski, Marek Dabrowski, Andrzej Bak, Andrzej Beben (WUT), Zbigniew Kopertowski (TPS), John Karadimas (QSY), Natalia Miettinen, Tero Kilkanen (ELI), Felix Strohmeier, Thomas Pfeiffenberger, Martin Herfurt (SPU), Heinz Doerken (DTA), Petros Sampatakos, Lila Dimopoulou, Sotiris Maniatis, Charilaos Tsetsekas (NTU)

Abstract:	This document describes the results of the 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, phase, testbed, Internet, test

Executive Summary

The present document is the first 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 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. Finally, the results of the integration are presented.

Table of Contents

1	INTRODUCTION.....	10
2	INTEGRATION METHODOLOGY AND TIME PLAN	11
3	INTEGRATION SITE TOPOLOGY	12
3.1	INTEGRATION AT DRESDEN	12
3.2	INTEGRATION AT WARSAW	13
3.2.1	Terminals.....	13
3.2.2	Routers	13
3.2.3	Measurement equipment.....	15
3.2.4	Network topology	15
4	PHASE 1: STANDALONE INTEGRATION.....	17
4.1	COMMON MODULES	17
4.2	RCA INTEGRATION	18
4.2.1	Available functionality for 1 st trial.....	18
4.2.2	Testing.....	19
4.2.2.1	Start up.....	19
4.2.2.2	Main functionality.....	19
4.3	ACA INTEGRATION.....	20
4.3.1	Available functionality for 1 st trial.....	20
4.3.2	Testing.....	21
4.3.2.1	Start up.....	21
4.3.2.2	Interaction with EAT.....	21
4.3.2.3	Ingress ACA – Egress ACA Interworking	25
4.3.2.4	Keep alive mechanism tests	26
4.3.2.5	Interaction with the Edge Device	27
4.4	EAT INTEGRATION.....	29
4.4.1	Available functionality for 1 st trial.....	29
4.4.2	Tests	30

4.4.2.1	EAT Manager.....	30
4.4.2.2	GUIs.....	32
4.4.2.3	Proxy.....	33
4.5	MEASUREMENT TOOLS	34
4.5.1	Available functionality	34
4.5.2	Tests	34
4.5.2.1	Measurement database	34
4.5.2.2	CM Toolset	34
4.5.2.3	T-Nova Toolset	35
4.5.2.4	Router QoS Monitoring Tool.....	35
4.5.2.5	GUI	35
4.6	ROUTER TESTING	36
4.6.1	QoS mechanisms implemented in the routers	36
4.6.2	Scope of the tests	37
4.6.3	Test configuration and required measurement equipment	37
4.6.4	Router performance tests	38
4.6.5	Tests of QoS mechanisms implemented in the routers	40
4.6.5.1	Packets classifier tests	40
4.6.5.2	Traffic conditioner tests	43
4.6.5.3	Queue management tests	49
4.6.5.4	Scheduler tests	50
4.6.5.5	Other tests	52
4.6.6	Test results.....	53
4.6.6.1	CISCO 7507 router tests	53
4.6.6.2	CISCO 3640 router tests	74
4.6.6.3	CISCO 1605 router tests	84
4.6.6.4	CISCO 2600 Router Tests.....	88
4.6.6.5	CISCO 827 and 1750 Router Tests.....	101

4.6.6.6	Summary	103
5	PHASE 2: RCL-ONLY INTEGRATION	104
5.1	RCA – ACA INTERWORKING	104
5.2	EAT – ACA INTERWORKING.....	105
5.3	EAT - SERVICE INTERWORKING.....	108
5.4	END - TO - END TESTS	108
5.5	FAILURE SCENARIOS.....	110
5.5.1	Recovery Scenarios	110
5.5.1.1	RCA Breakdown	110
5.5.1.2	ACA Breakdown.....	110
5.5.1.3	EAT Breakdown.....	111
6	PHASE 3: OVERALL INTEGRATION.....	112
6.1	RCL – DIFFSERV LAYER INTERWORKING.....	112
6.2	END – TO – END TESTS	113
6.2.1	RCA.....	113
6.2.2	ACA	113
6.2.3	EAT	113
6.2.3.1	EAT Manager.....	113
6.2.3.2	Proxy.....	114
6.2.3.3	GUIs.....	114
6.2.4	Measurement tools.....	114
6.2.4.1	CM Toolset	114
6.2.4.2	T-Nova Toolset	116
6.2.4.3	Router QoS Monitoring Tool.....	117
6.2.4.4	GUI	117
7	SOFTWARE MAINTENANCE AND BUG REPORT.....	119
8	PROCEDURES FOR SET-UP OF NEW TRIAL SITES.....	120
8.1	INSTALLATION AND USE OF RCL COMPONENTS	120

8.1.1	General Instructions.....	120
8.1.1.1	Fetching the RCL components.....	120
8.1.1.2	Installation and compilation.....	121
8.1.1.3	Running a Java class	122
8.1.1.4	Property files.....	122
8.1.1.5	Name Server.....	123
8.1.1.6	LDAP Server.....	123
8.1.1.7	Trace Server.....	129
8.1.1.8	QMTool	130
8.1.1.9	Operation of the Resource Control Layer	132
8.1.2	RCA.....	132
8.1.2.1	Required packages and property files.....	132
8.1.2.2	Operation	133
8.1.3	ACA	133
8.1.3.1	Required packages and property files.....	133
8.1.3.2	Operation	134
8.1.3.3	EDAdapter	135
8.1.3.4	System requirements	140
8.1.4	EAT	140
8.1.4.1	EAT Manager, API and EAT Script.....	141
8.1.4.2	Reservation GUI and Tomcat.....	142
8.1.4.3	Proxy.....	144
8.1.5	Measurement tools.....	146
8.1.5.1	Creation of the AQUILA database and loading some initial data	146
8.1.5.2	Update of web server configuration	147
8.1.5.3	Installation of the GUI to the measurement tools.....	147
8.1.5.4	Installation of the DTA masterstation software module	147
8.1.5.5	Installation of the SPU CMCaller software module.....	148
8.1.5.6	Installation of the DTA measurement agents	148

8.1.5.7	Installation of the SPU measurement daemons	150
8.1.5.8	Installation of the ELI router QoS monitoring tool	151
8.2	INSTALLATION AND USE OF APPLICATIONS	152
8.2.1	Siemens WinSIP	152
8.2.2	Unreal Tournament Client Epic Games & Digital Extremes	153
8.2.3	NetMeeting	154
8.2.4	RealSystem	155
9	RESULTS AND EXPERIENCES	157
10	ABBREVIATIONS	158
11	REFERENCES	160
12	APPENDIX	161
12.1	LDAP INSTALLATION	161
12.1.1	Evaluation of Netscape Directory Servers	161
12.1.1.1	Netscape Directory Server 4.11	161
12.1.1.2	Netscape Directory Server 4.12	162
12.1.1.3	Conclusion	162
12.1.2	Installation Procedures	162
12.1.2.1	Netscape Directory Server Software Components	163
12.1.2.2	Basic Configuration Information	165
12.1.2.3	Common Problems during installation	170

Table of Figures

FIGURE 1. TOPOLOGY OF THE DRESDEN INTEGRATION SITE	13
FIGURE 2. TOPOLOGY OF THE WARSAW INTEGRATION SITE	16
FIGURE 3. RECOMMENDED CONFIGURATION FOR ROUTER TESTING	37
FIGURE 4. CONFIGURATION FOR 7507 ROUTER TESTING	54
FIGURE 5. THROUGHPUT OF THE CISCO 7507 ROUTER	55
FIGURE 6. FRAME LOSS OF CISCO 7507 ROUTER IN THE CEF MODE	56
FIGURE 7. FRAME LOSS OF CISCO 7507 ROUTER IN THE DISTRIBUTED CEF MODE	56

FIGURE 8. FRAME LOSS OF CISCO 7507 ROUTER IN THE DISTRIBUTED CEF MODE WITH WFQ	57
FIGURE 9. FRAME LATENCY OF CISCO 7507 ROUTER	57
FIGURE 10. NUMBER OF DROPPED PACKETS	61
FIGURE 11. PACKET DROPPING RATE	61
FIGURE 12. CONFORMING BURST SIZES	62
FIGURE 13. PACKET LOSS RATIO WITH WRED MECHANISM USED	64
FIGURE 14. CBWFQ SCHEDULER PERFORMANCE ON FAST ETHERNET	66
FIGURE 15. FAIRNESS INDEX OF CBWFQ SCHEDULER ON FAST ETHERNET	66
FIGURE 16. PACKET LATENCY IN CBWFQ SCHEDULER ON FAST ETHERNET	67
FIGURE 17. CBWFQ SCHEDULER PERFORMANCE ON ETHERNET	68
FIGURE 18. FAIRNESS INDEX OF CBWFQ SCHEDULER ON ETHERNET	68
FIGURE 19. PACKET LATENCY IN CBWFQ SCHEDULER ON ETHERNET	69
FIGURE 20. LATENCY INTRODUCED BY THE LLQ SCHEDULER AS A FUNCTION OF BACKGROUND TRAFFIC LOAD	70
FIGURE 21. LATENCY INTRODUCED BY THE LLQ SCHEDULER AS A FUNCTION OF FOREGROUND TRAFFIC LOAD....	71
FIGURE 22. LLQ LATENCY AS A FUNCTION OF BG PACKET LENGTH.....	71
FIGURE 23. LLQ LATENCY AS A FUNCTION OF FG PACKET LENGTH	72
FIGURE 24. THROUGHPUT AS A FUNCTION OF NUMBER OF FLOW RESERVATIONS.....	73
FIGURE 25. CONFIGURATION FOR 3640 ROUTER TESTING.....	74
FIGURE 26. THROUGHPUT OF THE CISCO 3640 ROUTER	75
FIGURE 27. FRAME LOSS OF CISCO 3640 ROUTER IN THE RSP (NO CEF) MODE.....	76
FIGURE 28. FRAME LOSS OF CISCO 3640 ROUTER IN THE CEF MODE	76
FIGURE 29. FRAME LATENCY OF CISCO 3640 ROUTER	77
FIGURE 30. NUMBER OF DROPPED PACKETS	79
FIGURE 31. PACKET DROPPING RATE.....	79
FIGURE 32. CONFORMING BURST SIZES	80
FIGURE 33. CBWFQ SCHEDULER PERFORMANCE ON ETHERNET	82
FIGURE 34. FAIRNESS INDEX OF CBWFQ SCHEDULER ON FAST ETHERNET	82
FIGURE 35. THROUGHPUT AS A FUNCTION OF NUMBER OF FLOW RESERVATIONS.....	83
FIGURE 36. CONFIGURATION FOR ROUTER TESTING.....	84
FIGURE 37. THROUGHPUT AS A FUNCTION OF NUMBER OF FLOW RESERVATIONS.....	85
FIGURE 38. FRAME LOSS OF CISCO 1605 ROUTER IN THE RSP (NO CEF) MODE.....	85
FIGURE 39. FRAME LOSS OF CISCO 1605 ROUTER IN THE CEF MODE	86
FIGURE 40. FRAME LATENCY OF CISCO 1605 ROUTER	86
FIGURE 41. THROUGHPUT AS A FUNCTION OF NUMBER OF FLOW RESERVATIONS.....	87
FIGURE 42. CONFIGURATION FOR 2620 ROUTER TESTS	88
FIGURE 43. THROUGHPUT [PACKETS] AS A FUNCTION OF FRAME SIZE	89
FIGURE 44. THROUGHPUT [BYTES] AS A FUNCTION OF FRAME SIZE	90

FIGURE 45. ROUTER PERFORMANCE FRAME LOSS.....	91
FIGURE 46. ROUTER PERFORMANCE LATENCY	92
FIGURE 47. NUMBER OF DROPPED PACKETS IN ONE-MINUTE MEASUREMENT INTERVAL.....	96
FIGURE 48. PACKET DROPPING RATE.....	96
FIGURE 49. RATE OF RECEIVED AND LOST PACKETS IN THREE FLOWS.....	99
FIGURE 50. LATENCY FOR THE THREE FLOWS.....	100
FIGURE 51. MAXIMAL BIT RATE FOR DIFFERENT NUMBER OF FLOWS	101
FIGURE 52. THROUGHPUT [BYTES] AS A FUNCTION OF FRAME SIZE	102

Table of Tables

TABLE 1. NUMBER OF DROPPED PACKETS IN ONE-MINUTE MEASUREMENT INTERVAL.....	60
TABLE 2. PACKET DROPPING RATE	60
TABLE 3. CONFORMING BURST SIZES	62
TABLE 4. WFQ WEIGHTS	64
TABLE 5. TRANSMISSION RATES IN WFQ SCHEDULER ON FAST ETHERNET	65
TABLE 6. TRANSMISSION RATE IN WFQ SCHEDULER ON ETHERNET	67
TABLE 7. PERFORMANCE OF LLQ SCHEDULER.....	69
TABLE 8. MAXIMUM BIT RATE WITH 10 AND 50 FLOW RESERVATIONS.....	73
TABLE 9. NUMBER OF DROPPED PACKETS IN ONE-MINUTE MEASUREMENT INTERVAL.....	78
TABLE 10. PACKET DROPPING RATE	78
TABLE 11. CONFORMING BURST SIZES	80
TABLE 12. WFQ WEIGHTS	81
TABLE 13. TRANSMISSION RATES IN WFQ SCHEDULER ON ETHERNET	81
TABLE 14. MAXIMUM BIT RATE WITH 10 AND 50 FLOW RESERVATIONS.....	83
TABLE 15. MAXIMUM BIT RATE WITH 10 AND 50 FLOW RESERVATIONS.....	87
TABLE 16. NUMBER OF DROPPED PACKETS IN ONE-MINUTE MEASUREMENT INTERVAL.....	94
TABLE 17. PACKET DROPPING RATE.....	95
TABLE 18. MAXIMUM BIT RATE WITH DIFFERENT NUMBER OF FLOWS	100
TABLE 19: RANGE OF PARAMETERS FOR DIFFERENT TRAFFIC CLASSES.....	140

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. It also provides detailed instructions for the establishment of new trial sites.

The document is structured according to the phased integration methodology adopted by the project, as described in section 2. In section 3, the topology and the equipment of the two integration sites (Dresden and Warsaw) are described. In sections 4, 5 and 6, the test cases used for the verification of the produced software and the functionality of the routers are presented, according to the phase they belong to. In section 7, procedures for the establishment of a software library are described and in section 8, detailed instructions for the establishment of new trial sites can be found. In section 9, we present some results and experiences of the integration procedure, while in the appendix the installation of an LDAP server is described.

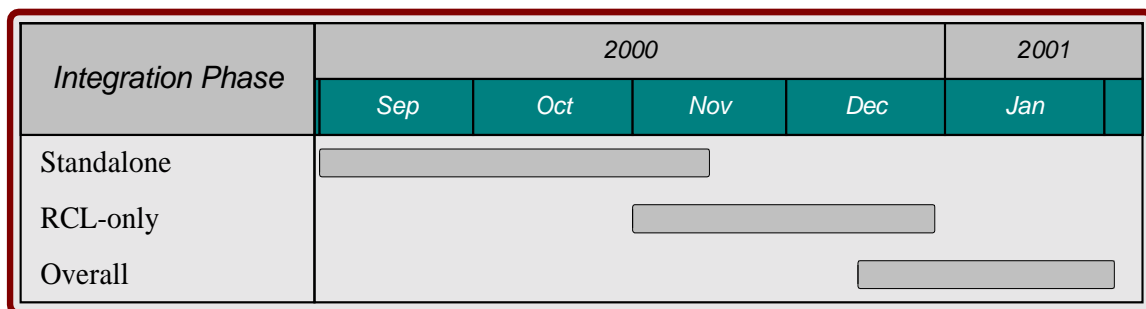
2 Integration methodology and time plan

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

Three distinct phases have been identified:

- **Standalone integration:** the functionality of individual modules was tested. In some cases this was achieved with the use of dummy modules. With the term “dummy”, we refer to modules that only implement the interface to the module under test and not the complete functionality.
- **RCL-only integration:** the functionality of the Resource Control Layer was tested. This was mainly performed during the integration session in Dresden. End-to-end reservations were accomplished, without using any routers.
- **Overall integration:** Routers were incorporated to the already tested Resource Control Layer and end-to-end reservation tests were conducted, using applications or traffic generators.

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



As can be deduced from the above figure, the integration phases were overlapping. This enabled the project to gradually introduce and test different features of a specific component. For instance, after a version of a component passed the standalone tests, it could be incorporated in the RCL-only testing. However on the same time, additional features of this component that had not been tested before could take part in standalone testing.

In the above figure, it can also be seen that the final deadline for the production of a working AQUILA prototype was the end of January. In the subsequent two months until the submission of this document, the WP 3.1 team was involved in assisting the installation to the other trial sites and the running of the trials, making corrections to the AQUILA prototypes, when needed.

3 Integration site topology

As mentioned in the previous paragraph, the integration process was broken down in three major phases. Standalone integration took place at each partner's premises. During this phase, individual components were tested. For the end-to-end testing of the Resource Control Layer, an integration meeting was held in Dresden. In this session, only the RCL functionality was tested and no routers were used. Finally, the RCL as well as the Measurement Tools were tested in an overall testing session in Warsaw.

In the following paragraphs, the topology and the equipment used at the two integration sites are presented.

3.1 Integration at Dresden

TUD hosted the second integration step, the RCL-only integration. For that, TUD organised an integration meeting at its laboratory with the following equipment:

- Sun Enterprise 450 server, Solaris 7, 2 processors, 400 MHz, providing NIS including a common home directory on all workstations
- 4 Sun Ultra 60 workstations, Solaris 7, 2 processors, 360 MHz, integrated PC card, all with the same environment including Java 2 SDK, JBuilder 3.5, Together 4.1, Netscape Communicator
- 2 Sun Ultra 5 workstations, Solaris 7, 270 MHz

Additionally, some Windows NT 4.0 workstations were available. Here is a simple figure of the topology for the integration session at the TUD lab:

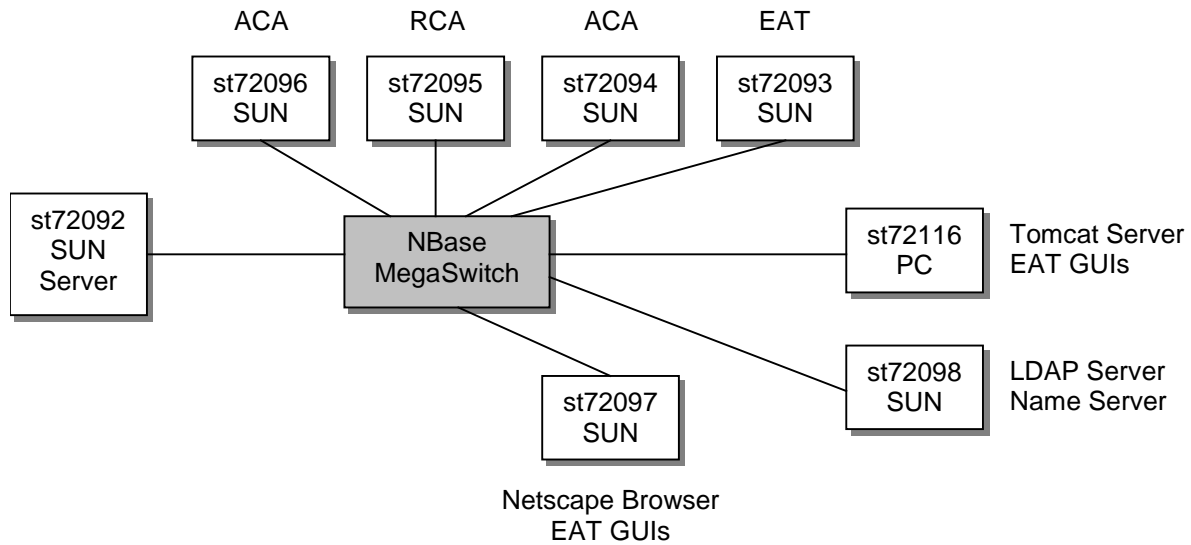


Figure 1. Topology of the Dresden integration site

3.2 Integration at Warsaw

Warsaw hosted the third phase of the integration. During this integration session, the routers were incorporated into the AQUILA topology and end-to-end tests were conducted in order to verify the AQUILA system functionality.

3.2.1 Terminals

For the first trial the following terminals with Ethernet 10/100 cards were available:

- 3 Sun Workstations with Solaris 2.7 operating system,
- 9 PC computers. The end-stations were equipped with Pentium III – 700Mhz processors. The operating systems installed on the PCs are Linux SuSe 6.4 and Windows NT 4.0. Two PCs were running Windows 2000 system instead of WinNT. Four terminals were equipped with GPS cards and were used with the Measurement Tools. Also, two PCs had cameras on and were used for the NetMeeting trial.

3.2.2 Routers

The below-described equipment was available during the integration session and will be dedicated to the AQUILA project for the whole period of its duration. CISCO routers were used with functionality appropriate for core and access networks. In the following, a detailed description of available routers is provided. In total, 10 CISCO routers were available in the TPS laboratory.

Router Cisco 7507 – basic version (2 routers), extended (one router)

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
1-Port Fast Ethernet 100BaseTx Port Adapter	1 module / router
4-Port Ethernet 10BaseT Port Adapter	1 module / router
1-Port Packet/SONET OC3c/STM1 Singlemode (IR) Port Adapter	1 module / router
1-Port ATM Enhanced OC3c/STM1 Singlemode(IR)Port Adapter	1 module / extended router

Router 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
4-Port Ethernet Network Module	1 module / router
1-Port Fast Ethernet Network Module (TX Only)	1 module / router
4-Port Serial Network Module	1 module / router

Router Cisco 1750 (2 routers)	
10/100 Modular Router w/1VIC, 2WIC/VIC slots, Cisco IOS IP SW	1 module / router
2-Port Serial WAN Interface Card	1 module / router
V.35 Cable, DTE Male to Smart Serial, 10 Feet	2 module / router

Router Cisco 1605 (2 routers)	
IOS software release	Cisco Internetwork Operating System Software IOS (tm) 1600 Software (C1600-SY-M), Version 12.1(5), RELEASE SOFTWARE (fc1) c1600-sy-mz.121-5.bin
Router processor	Cisco 1605 (68360) processor (revision C) with 12288K/4096K bytes of memory.
1-Port Serial WAN Interface Card	1 module / router
1-Port 10 Mbps Ethernet	2 ports/ router

3.2.3 Measurement equipment

The following measurement equipment was available for the first trial:

- InterWatch9500 with IP performance module and ATM interface – analyser/generator,
- HP BSTS with IP performance module and PoS interface – analyser/generator,
- GPS system.

3.2.4 Network topology

In the following figure, the integration topology used, is presented:

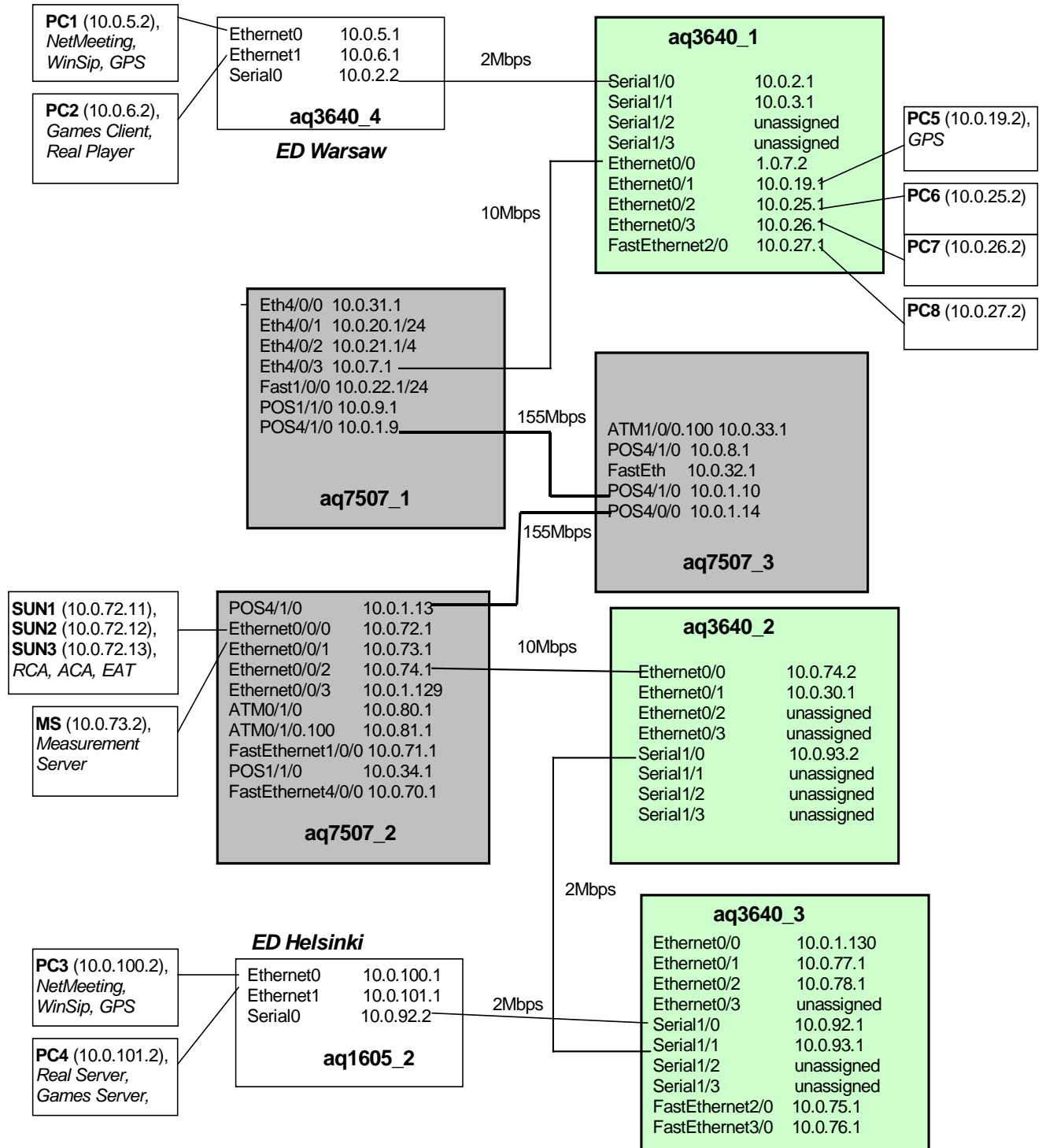


Figure 2. Topology of the Warsaw integration site

4 Phase 1: Standalone integration

In this section, the activities for standalone integration are described. A characteristic of this phase is that two of the main entities of the RCL, the RCA and the ACA, were in their entirety implemented and debugged by only one partner (SAG and NTU respectively). This fact made easier the task of standalone verification of those modules.

4.1 Common modules

This section deals with modules that are used by more than one entity. Such modules are the Name Server, Trace Server and the Persistence Layer.

Activity No.	phase1.common.1
Title	CORBA and Naming Service initialisation
Description	For the CORBA Naming Service, the transient naming server (tname-serv) contained in the JDK is used. This test verifies, that tnameserv can be started.
Preconditions	Tnameserv available
Postconditions	Tnameserv running on port 1234 (selected for AQUILA)
Notes	Note, that port numbers below 1024 are restricted to be used by the super user only on UNIX systems. So an arbitrary and unused port above 1023 has to be used.

Activity No.	phase1.common.2
Title	Persistence Layer initialisation
Description	<p>The persistence layer is realised by an LDAP server. The following topics are covered by this activity:</p> <ul style="list-style-type: none"> • Start-up of the LDAP server. • Add necessary schema entries. The following object classes have to be available: <ul style="list-style-type: none"> • javaContainer • javaSerializedObject • groupOfNames • remoteUser • ipNetwork • device • Add a root entry of “o=AQUILA,c=EU” • Enable write access to the database • Run aquila.rcl.service.GenerateServices • Run aquila.rcl.tc.GenerateTrafficClasses • Load other content (Subscriber, Pool, Subnet, Router, see [D2101], 4.8.1) to LDAP database using LDIF files • Write protect database

Preconditions	LDAP server already installed and configured
Postconditions	LDAP server with necessary schema entries running. LDAP provider URL available for RCL configuration files. LDAP content available.
Notes	At the beginning, Sun Directory Service 3.1 was used. Later, there was a shift to Netscape Directory Service. The LDAP content for Pool, Subnet and Router will have to be changed to adapt to various network topologies and scenarios.

Activity No.	phase1.common.3
Title	Trace Service initialisation
Description	<p>Start the trace server aquila.rcl.util.TraceMain. Verify, that the trace server GUI is displayed. This implicitly tests also the functions of aquila.rcl.util.Main:</p> <ul style="list-style-type: none"> • Initialises ORB • Gets reference to CORBA naming service • Gets reference to JNDI LDAP DirContext • Creates a Trace object • Reads the AQUILA property file and provides these properties for the application <p>Test these functions with the following activities:</p> <ul style="list-style-type: none"> • Try to start trace server without property file. This should not work (LDAP server and/or CORBA naming service not found). • Provide property file aquila.rc in the user.dir. • Provide property file in another location and describe this location with the property aquila.rcl.propertyFile=<path>. • Set property aquila.rcl.trace.dir in property file and check, that trace files are generated in that directory instead of user.dir.
Preconditions	phase1.common.1, phase1.common.2
Postconditions	Trace server running. Main functions available
Notes	

4.2 RCA integration

4.2.1 Available functionality for 1st trial

In the following, a list of the available functions during the integration is presented:

- Resource management, creation of the Resource Pool tree
- Resource distribution to the ACAs
- Resources re-distribution algorithm
- Keep-alive mechanism (ACA failure)

- Exception handling during failure situations

4.2.2 Testing

4.2.2.1 Start up

Activity No.	phase1.rca.startup.1
Title	RCA start-up
Description	Checks the connection with the Interaction with LDAP, CORBA. Creates the NetworkManagerImpl object and initialises the Traffic Class package.
Preconditions	Netscape directory Server should be started, the information needed should be already stored in LDAP (traffic classes, rpool info). Trace-Main is already started.
Postconditions	References to the NetworkManagerImpl object and to the traffic classes. An exception may be thrown
Notes	

Activity No.	phase1.rca.startup.2
Title	Resource pool creation & initialisation
Description	The RPool information is retrieved from the persistence layer, the RPool tree is constructed and the initial resources are assigned.
Preconditions	
Postconditions	The RPool structure is ready and the RCA can accept reservations requests. An exception may be thrown
Notes	

4.2.2.2 Main functionality

Activity No.	phase1.rca.resourceShareImpl.1
Title	Allocation of a resource
Description	Call of the ResourceShareImpl.alloc() via CORBA
Preconditions	The ACA is started and the ResourcePoolLeaf is created and instantiated
Postconditions	Reference to the allocation object. An exception may be thrown
Notes	Dummy ACA module used

Activity No.	phase1.rca.resourceShareImpl.2
Title	De-allocation of a resource
Description	Call of the ResourceShareImpl.dealloc() via CORBA
Preconditions	An appropriate reservation request should be already accepted, reference to the allocation.
Postconditions	An exception may be thrown
Notes	Dummy ACA module used

Activity No.	phase1.rca.resourceShareImpl.3
Title	De-allocation of a resource (low watermark crossed)
Description	Call of the ResourceShareImpl.dealloc() via CORBA in order to test the resource redistribution algorithm.
Preconditions	An appropriate reservation request should be already accepted, reference to the allocation.
Postconditions	An exception may be thrown
Notes	Dummy ACA module used

Activity No.	phase1.rca.resourceShareImpl.4
Title	Allocation of a resource (2)
Description	Call of the ResourceShareImpl.alloc() via CORBA Multiple allocation scenarios in order to check the functionality of the resource redistribution algorithm
Preconditions	The ACA is started and the ResourcePoolLeaf is created and instantiated
Postconditions	An exception may be thrown
Notes	Dummy ACA module used. Several reservation scenarios should cover all three cases of the algorithm

4.3 ACA integration

4.3.1 Available functionality for 1st trial

The software implementation of the ACA had been completed and the specified functions in the work package WP2.1 were completely available for the integration, with the exception of the collection of usage data.

In the following, a list of the available functions during the integration is presented:

- User log in with authentication
- User log out
- User requests
 - of network services with authorisation
 - with a service check against allowed parameter ranges
- User release of requests
- Reservations at the ingress and/or egress side of a domain (reservation style p2p, p2a)
- Configuration of the edge devices

- Resource requests or resource releases at the resource control agent (RCA)
- Exception handling during failure situations

4.3.2 Testing

4.3.2.1 Start up

Activity No.	phase1.aca.startup.1
Title	ACA initialisation
Description	Registration of the SessionManager and the LocalResourceManager in the CORBA naming service Retrieval of the resource shares from the resource control agent (RCA)
Preconditions	Running CORBA naming service
Postconditions	The SessionManager and the LocalResourceManager server object are available from the CORBA naming service.
Notes	The RCA was simulated in these stand-alone test cases.

4.3.2.2 Interaction with EAT

Activity No.	phase1.aca.eat.1
Title	Authenticated user logs in
Description	The user has to resolve the reference to the SessionManager server with help of the CORBA naming service. The SessionManager, a component of the ACA, offers the user the log in interface. This interface method needs the information of the user's account and password. With this information and correct database content the log in must be possible.
Preconditions	The user has been registered in the database
Postconditions	The user receives a valid UserAgent interface and the keep alive manager of the ACA contains the alive peer related to the user The keep alive mechanism between these two peers is active
Notes	

Activity No.	phase1.aca.eat.2
Title	Authenticated user logs in a second time
Description	A logged in user logs in a second time from the same host to the same ACA manager.
Preconditions	The user has been registered in the database.
Postconditions	The user receives another valid UserAgent interface and the keep alive manager of the ACA contains the alive peer concerning this user. The keep alive mechanism is not affected.
Notes	The keep-alive messages between the user and the ACA manager are independent of how often the user logs in from the same machine to the same ACA Manager.

Activity No.	phase1.aca.eat.3
--------------	------------------

Title	Not Authenticated user logs in
Description	Identical to phase1.aca.eat.1 LoginException is thrown
Preconditions	The user is not registered in the database
Postconditions	No reference to an ACA's UserReservation object is available for EAT
Notes	

Activity No.	phase1.aca.eat.4
Title	User logs in with a wrong password
Description	Identical to phase1.aca.eat.1 Login Exception is thrown
Preconditions	The user is registered in the database.
Postconditions	No reference to an ACA's UserReservation object is available at EAT
Notes	

Activity No.	phase1.aca.eat.5
Title	User logs out
Description	The user calls the log out method of the UserAgent interface;
Preconditions	phase1.aca.eat.1
Postconditions	The user has no valid reference to an UserAgent object The keep alive manager contains no alive peer related to this user and the keep alive mechanism between these two peers is inactive
Notes	

Activity No.	phase1.aca.eat.6
Title	Sequentially log in / log out
Description	Different users log in and log out sequentially.
Preconditions	The users are registered in the database.
Postconditions	No exception is expected
Notes	Users are logged in or logged out, it depends on the interruption of the test sequence.

Activity No.	phase1.aca.eat.7
Title	Request Network Services (authorised user)
Description	The user requests the network services registered for this user in the database
Preconditions	phase1.aca.eat.1 The user is authorised to request these services.
Postconditions	The user has a valid reference to a UserReservation object.
Notes	

Activity No.	phase1.aca.eat.8
Title	Request Network Services (not authorised user)
Description	The user requests Network Services for which he/she is not authorised

Preconditions	phase1.aca.eat.1 The user is NOT authorised to request all possible services.
Postconditions	For an unauthorised request the user receives no reference to a UserReservation object. No Communication session is started at ACA.
Notes	

Activity No.	phase1.aca.eat.9
Title	Release Network Services
Description	The user releases the requested network services.
Preconditions	phase1.aca.eat.6
Postconditions	The user has no valid reference to a UserReservation object.
Notes	

Activity No.	phase1.aca.eat.10
Title	Request Network Services at an inactive ACA
Description	The users request a network service from an Access Control Agent that is not available at this moment.
Preconditions	phase1.aca.eat.1 The user is authorised to request these services.
Postconditions	The ACA manager throws a RequestException. The user has no valid reference to a UserReservation object.
Notes	

Activity No.	phase1.aca.eat.11
Title	Invalid Network Services
Description	The user requests an network service that is not defined
Preconditions	phase1.aca.eat.6
Postconditions	A RequestException is thrown The user has no valid reference to a UserReservation object
Notes	

Activity No.	phase1.aca.eat.12
Title	Invalid Scope
Description	The user requests a Reservation Style that is not defined
Preconditions	phase1.aca.eat.6
Postconditions	A RequestException is thrown The user has no valid reference to a UserReservation object
Notes	

Activity No.	phase1.aca.eat.13
Title	Valid Flow Description
Description	The user requests an authorised network service several times, and in the different requests the destination addresses should be chosen out of different networks that are a class A/B or C. In each request the subnet

	mask changes the “subnetting” area
Preconditions	phase1.aca.eat.6
Postconditions	The user has a valid reference to a UserReservation object.
Notes	

Activity No.	phase1.aca.eat.14
Title	Invalid Flow Description
Description	The user requests an authorised network service, but the description of the flow contains a destination address or a range of addresses that can not be mapped to a network.
Preconditions	phase1.aca.eat.6
Postconditions	A RequestException is thrown. The user has valid reference to a UserReservation object.
Notes	

Activity No.	phase1.aca.eat.15
Title	User logs out with current stored reservations.
Description	The user has been logged in AND has requested reservation, which are still stored for this user. The user logs out without an explicit release of the reserved resources.
Preconditions	A user logs in and requests authorised network services phase1.aca.eat.7
Postconditions	The user is not logged in and no reservations are stored for this user
Notes	

Activity No.	phase1.aca.eat.16
Title	Request Network Services (authorised user) but no resources are available at the egress side.
Description	The user requests the network services registered for this user in the database but no resources are available at the egress side.
Preconditions	phase1.aca.eat.7 The user is authorised to request these services.
Postconditions	The user receives no valid reference to a UserReservation object. No resources are reserved for this user request and the EAT receives a RequestException.
Notes	

Activity No.	phase1.aca.eat.17
Title	Request Network Services (authorised user) but no resources are available at the ingress side.
Description	The user requests the network services registered for this user in the database but no resources are available at the ingress side.
Preconditions	phase1.aca.eat.7 The user is authorised to request these services.
Postconditions	The user has no valid reference to a UserReservation object. No re-

	sources are reserved for this user request and the EAT receives a RequestException. The resources at the egress side requested before the ingress reservation are released.
Notes	

4.3.2.3 Ingress ACA – Egress ACA Interworking

Activity No.	phase1.aca.ingress-egress.1
Title	ACA – ACA Interworking
Description	<p>Interworking between the ACA at the ingress and ACA at the egress side and the ACA manager (located on a third device->third party scenario)</p> <p>The user starts a reservation request at the ACA manager (part of network1). The source IP address specified in the request is part of the second network and the destination IP address range is part of the third network.</p>
Preconditions	<p>phase1.aca.eat.1</p> <p>Three different access IP network descriptions are stored in LDAP database and at each access network one ACA is located.</p>
Postconditions	The user receives a reference to a UserReservation object
Notes	

Activity No.	phase1.aca.ingress-egress.2
Title	Trace of the ingress ACA, egress ACA, and ACA manager information on each local station
Description	This test case checks if the logging mechanism is available. Therefore the ACA components have to be started and the user initiates a log in action. Then the expected files have to be stored at each device and the logging and debugging operations have to be stored correctly.
Preconditions	The ACA is started on different devices
Postconditions	On each device a logging file has to be created with the file name <aca-name>.log
Notes	

Activity No.	phase1.aca.ingress-egress.3
Title	Trace of the ingress ACA, egress ACA, and ACA manager information on the trace server
Description	The trace server logs the trace and the debug information of each RCL component in a central file.
Preconditions	<p>The trace server is running</p> <p>phase1.aca.ingress-egress.2</p>
Postconditions	On the device where the trace server is started, a logging file has to be created with the file name aquila.log
Notes	

Activity No.	phase1.aca.ingress-egress.4
Title	Ingress ACA identically with Egress ACA
Description	A user logs in and requests a Network Service. The source address range and the destination address range specified in the request are mapped to the same Admission Control Agent.
Preconditions	User has been logged in
Postconditions	No reservations were done at this ACA. The user is always logged in and the requested Network Service is successfully offered. But no reservations are done.
Notes	

Activity No.	phase1.aca.ingress-egress.5
Title	Request resources from an ACA after the reset of this ACA
Description	A user logs in and requests a Network Service. After the service requests the ACA breaks down and restarts. After restart of the ACA the user requests another Network Service.
Preconditions	The user logs in and requests Network Services. See phase1.aca.eat.7
Postconditions	The user has a reference to the Network Service, which was requested after the restart of the ACA.
Notes	

4.3.2.4 Keep alive mechanism tests

Activity No.	phase1.aca.keepalive.1
Title	Keep-alives between ACAs
Description	Keep alive messages are transmitted between the EAT and the ACA
Preconditions	Authenticated user logs in
Postconditions	“hello” messages are transmitted from ACA to EAT and from EAT to ACA; the time interval has to be the same as coded.
Notes	

Activity No.	phase1.aca.keepalive.2
Title	Keep-alives between ACAs (2)
Description	The number of the keep alive messages transmitted between the EAT and the ACA are independent from the number of users.
Preconditions	Many authenticated users log in, using the same EAT.
Postconditions	“hello” messages are transmitted from ACA to EAT and from EAT to ACA; the time interval has to be the same as coded and the number of the messages has to be independent from the number of users which log in with the same EAT.
Notes	

Activity No.	phase1.aca.keepalive.3
Title	ACA Manager breakdown
Description	A third party reservation is done and then the ACA manager breaks

	down. The KeepAlive mechanism initiates the release of the requested services at the ingress and egress side.
Preconditions	phase1.aca.ingress-egress.1
Postconditions	No stored reservations exist at the ingress and egress ACAs
Notes	

Activity No.	phase1.aca.keepalive.4
Title	Ingress ACA breakdown
Description	A third party reservation is initiated and the ingress ACA breaks down. The KeepAlive mechanism informs the ACA manager that the ingress peer is not available. Then the ACA manager releases the reservation at the egress side and informs the EAT with an event (ReservationDown).
Preconditions	phase1.aca.ingress-egress.1
Postconditions	The communication session is still running at the ACA manager but the requested reservations have been released.
Notes	

Activity No.	phase1.aca.keepalive.5
Title	Egress ACA breakdown
Description	A third party reservation is initiated and the egress ACA breaks down. The KeepAlive mechanism informs the ACA manager that the egress peer is not available. Then the ACA manager releases the reservation at the ingress side and informs the EAT with an event (Reservation-Down).
Preconditions	phase1.aca.ingress-egress.1
Postconditions	The communication session is still running at the ACA manager but the requested reservations have been released.
Notes	

Activity No.	phase1.aca.keepalive.6
Title	Interruption of the ACA-ACA connection
Description	A third party reservation is initiated and the ingress or egress ACA is disconnected from the network. The KeepAlive mechanism informs the ACA manager that the ingress or egress peer is not available. Then the ACA manager releases the reservation at the egress or ingress side and informs the EAT with an event (ReservationDown).
Preconditions	phase1.aca.ingress-egress.1
Postconditions	The communication session is still running at the ACA manager but the requested reservations have been released.
Notes	

4.3.2.5 Interaction with the Edge Device

Activity No.	phase1.aca.edgeDevice.1
--------------	-------------------------

Title	Connection to an edge router via telnet
Description	<p>The EDAdapter sends all the commands to the router via a telnet connection. The EDAdapter parses the prompts after each committed command. To configure the different QoS functionalities one have to change between different modes. The prompt of each mode is different. Therefore the correctness of all the prompts has to be tested. For testing the prompt, simply a test-configuration has to send to the router.</p> <p>The EDAdapter should work independent of the telnet connection. The commands are stored in the telnet class and send later if the connection breaks down and is re-established. The correct reconnection should be tested. This functionality can be simulated by disconnecting the EDAdapter from the net and by a reconnection.</p>
Preconditions	Prompts, test configuration
Postconditions	None
Notes	

Activity No.	phase1.aca.edgeDevice.2
Title	Initialise Router
Description	<p>During the start up of the ACA, the EDAdapter sends the initial commands to the router. First the telnet connection to the router has to be established and for the configuration the mode has to change. The EDAdapter can start after a successful configuration of the scheduler. By a break during the start-up, the EDAdapter should go back to the beginning and try the start-up again. This mechanism can be tested with a wrong command in the start-up.</p> <p>During the runtime the scheduler is static. The configuration of the scheduler is also part of the initialisation. After the initialisation the EDAdapter should end in the EXEC modus. For the test of the right behaviour of the Scheduler we need a generator and an analyser. The generator should send packets marked with the DSCP of the different traffic classes. With the analyser the correct scheduling of the packets should be controlled.</p> <p>The WRED mechanism also belongs to the Scheduler. Therefore the packets should be marked with a different DSCP.</p>
Preconditions	Configuration strings
Postconditions	Configuration on the ED (right prompt)
Notes	

Activity No.	phase1.aca.edgeDevice.3
Title	Establish new flow
Description	<p>For each ingress flow the EDAdapter configures the classifier, marker and policer. The classifier is handled with an access-list. The classifier should be tested with different values for the classification.</p> <p>The classifier can distinguish the packets with different fields, like the address, protocol, port, and DSCP. For testing, the generator should</p>

	<p>send packets with different entries in the fields to the router. Maybe a starting point is the source and destination address.</p> <p>For testing of the policer the router should be stimulated with IP traffic, conforming and non-conforming to the traffic spec. For the first trial we use three different traffic models. All three traffic models belonging to different traffic classes should be tested. The marking of the conforming and non-conforming packets is described in [D1301].</p>
Preconditions	ClassifierInfo, MarkerInfo, TrafficDescription
Postconditions	Reference to the new IngressFlow object
Notes	

Activity No.	phase1.aca.edgeDevice.4
Title	Get statistics
Description	The router takes statistics from every interface. From this statistics the behaviour of the policer can be checked. The interface statistic can get with the “show interface” command. The number of conforming and non-conforming packets is listed. The statistic data of the router should be compared with the statistic data from the analyser.
Preconditions	IngressFlow
Postconditions	None
Notes	

4.4 EAT integration

4.4.1 Available functionality for 1st trial

For the first trial, the following EAT components and their functionality is available:

- EAT Manager:
 - user login and logout
 - reservation request and release
 - accounting information retrieval
 - network service retrieval
- Proxy (NetMeeting Proxy):
 - detection of flows
 - registration of applications
 - notification to EAT Manager about IP addresses and ports

- GUIs:
 - login
 - reservation
 - release
- EAT Script:
 - parsing of XML scripts with login and reservation scenarios

4.4.2 Tests

4.4.2.1 EAT Manager

Activity No.	phase1.eat.eatManager.1
Title	Login to ACA
Description	Creation a new EndUser object or, if it is already existing, returning the reference of the existing one Call of the SessionManager.login method via CORBA in the first case
Preconditions	End-user's account name, password
Postconditions	Reference to a new UserAgent object
Notes	Dummy ACA

Activity No.	phase1.eat.eatManager.2
Title	Reservation request to ACA
Description	Creation of a new Reservation object Call of the UserAgent.requestReservation method via CORBA
Preconditions	Network service id, SLS, etc.
Postconditions	Reference to a new UserReservation object
Notes	Dummy ACA

Activity No.	phase1.eat.eatManager.3
Title	Accounting information retrieval from ACA
Description	Call of the UserReservation.getAccountingData method via CORBA
Preconditions	None
Postconditions	Accounting data
Notes	Dummy ACA

Activity No.	phase1.eat.eatManager.4
Title	Reservation release to ACA
Description	Call of the UserReservation.release method via CORBA Destruction of the Reservation object
Preconditions	None

Postconditions	None
Notes	Dummy ACA

Activity No.	phase1.eat.eatManager.5
Title	Logout to ACA
Description	Call of the UserAgent.logout method via CORBA Destruction of the EndUser object
Preconditions	None
Postconditions	None
Notes	Dummy ACA, reservations are released and destroyed before

Activity No.	phase1.eat.eatManager.6
Title	Network service retrieval
Description	Call of the ServiceManager.getAllNetworkServices and ServiceManager.getNetworkService methods via CORBA Update of list of MirroredNetworkServices objects
Preconditions	for the second method: service id
Postconditions	Network services as singleton or sequence
Notes	Dummy service package

Activity No.	phase1.eat.eatManager.7
Title	EAT script processing
Description	Sequential reading of an XML source containing the specification of logins, reservation request, releases, and logouts
Preconditions	XML file that is based on the EATScript.dtd
Postconditions	Login, Request, Release, and Logout scenario
Notes	Separate tool necessary

Activity No.	phase1.eat.eatManager.8
Title	EAT reservation recovery
Description	Recovery of former taken reservations after a interim broken ACA
Preconditions	ACA is ready again after a specific time
Postconditions	
Notes	Notification of the end-user if a recovery is not possible

Activity No.	phase1.eat.eatManager.9
Title	Subscriber package integration
Description	The EAT Manager integrates the subscriber packages for two purposes: 1. to verify the end-user's password within the EAT (for a further login of an already logged end-user) 2. to get some contact information (e.g. the e-mail address) in order to inform him/her about a broken reservation
Preconditions	subscriber package available
Postconditions	

Notes	
-------	--

4.4.2.2 GUIs

4.4.2.2.1 Login GUI

Activity No.	phase1.eat.gui.logingui.1
Title	Tomcat start up
Description	Tomcat is started. The aquila context is created in the web server
Preconditions	None
Postconditions	The user may load the Login GUI
Notes	

Activity No.	phase1.eat.gui.logingui.2
Title	Creation of the GUI
Description	The user loads the Login GUI from his browser. The JSP page is compiled at the web server and sent to the browser
Preconditions	phase1.eat.gui.logingui.1 EAT should be already started and tnameserv should be running
Postconditions	The user may log in the AQUILA network
Notes	

Activity No.	phase1.eat.gui.logingui.3
Title	User log in
Description	The user logs in. Request for account name and password
Preconditions	phase1.eat.gui.logingui.2
Postconditions	The user is logged in the AQUILA network Reference to a new QoSSessionRequest object for the logged end-user
Notes	Reference is needed by the Reservation GUI, Legacy App GUI

4.4.2.2.2 Reservation GUI

Activity No.	phase1.eat.gui.resvgui.1
Title	Creation of the GUI
Description	The JSP page is compiled at the web server. Afterwards, the available Network Services are retrieved
Preconditions	tnameserv running
Postconditions	The user may select appropriate parameters for reservation
Notes	

Activity No.	phase1.eat.gui.resvgui.2
Title	Initiation of a request to EAT Manager
Description	The user selects parameters for the reservation and submits it
Preconditions	JSP page has been compiled, tnameserv running
Postconditions	The user receives acknowledgement of the successful reservation or a message stating that the reservation was denied along with failure rea-

	son
Notes	

4.4.2.2.3 Release GUI

Activity No.	phase1.eat.gui.releasegui.1
Title	Release a request to EAT Manager
Description	The user should be able to see all the reservation he has made so far and choose which one(s) to cancel
Preconditions	Already successful reservations created (phase1.eat.gui.resvgui.2)
Postconditions	The user receives acknowledgement of the successful reservation release
Notes	

4.4.2.3 Proxy

Activity No.	phase1.eat.proxy.1
Title	Proxy instantiation
Description	The Proxy Manager module is initialised. It also instantiates all Application proxy modules. Packet filtering is initiated
Preconditions	Tnameserv running
Postconditions	Proxy is ready for operation. EAT Manager can only then be started (if Proxy is to be used)
Notes	

Activity No.	phase1.eat.proxy.2
Title	Application registration
Description	At the request of the EAT Manager, prepares for the launching of a new application.
Preconditions	phase1.eat.proxy.1
Postconditions	Proxy is searching for flows matching the criteria given by the EAT Manager
Notes	

Activity No.	phase1.eat.proxy.3
Title	IP flow detection
Description	The Proxy detects a matching flow. It collects the requested information (addresses, ports) and forwards it to the EAT Manager.
Preconditions	phase1.eat.proxy.2
Postconditions	Flow is moved from the registration list to the active flow list. Measurements are conducted
Notes	

4.5 Measurement Tools

4.5.1 Available functionality

The available functionality of the measurement components was specified by WP2.3 and reported in [D2301].

4.5.2 Tests

4.5.2.1 Measurement database

Activity No.	phase1.measurement.db.1
Title	Creation of the database
Description	As DBMS a MySQL database is used. The script has to be loaded into the MySQL database. It creates the necessary database tables and inserts some sample data.
Preconditions	MySQL server has to be running, root privileges necessary
Postconditions	The measurement database is accessible for the users "apache" and "aquila".
Notes	

4.5.2.2 CM Toolset

Activity No.	phase1.measurement.cm.1
Title	CMCaller and Resultserver start-up
Description	The CMCaller is started and ready for flows to be started by periodically scanning the contents of the flow table in the measurement database. The Resultserver is awaiting results from measurement flows.
Preconditions	phase1.measurement.db.1 (Measurement Database), ODBC-Drivers available
Postconditions	Continuous requests from CMCaller to the database, whether new flows are ready specified. Resultserver is ready for retrieving results.
Notes	

Activity No.	phase1.measurement.cm.2
Title	CMDaemon start-up
Description	CMDaemon announces itself at the CMCaller and retrieves its unique hopid. The hopstate in the database is updated and basic information like IP-address, netmask and GPS location data (optional) is stored in the database (if not already existing).
Preconditions	phase1.measurement.db.1 (CMCaller running)
Postconditions	CMDeamon ready for accepting new flows to be started
Notes	

4.5.2.3 T-Nova Toolset

Activity No.	phase1.measurement.tnova.1
Title	Connection of masterstation module and MySQL database
Description	The masterstation accesses the MySQL database for reading configuration data and storing measurement results. Therefore a working connection of the masterstation and the database is vital for the toolset.
Preconditions	Running MySQL server / Activity phase1.measurement.db.1 / The masterstation software module (“master”) and the configuration file (“master.rc”) is copied to the masterstation / The correct database name, username and password is inserted in the configuration file (For details see Chapter 8 “Procedures for set-up of new trial sites”)
Postconditions	If the Status screen 5 of the masterstation software module shows the message “Connected to database” the connection masterstation database is ok.
Notes	

Activity No.	phase1.measurement.tnova.2
Title	Measurement agent start-up
Description	The measurement agents send and receive the measurement packets and send the results to the masterstation. The measurement clients need a special kernel version and a correct installed GPS card. The client testmodule “ndm” (called with the client’s IP-address as parameter) shows the kernel version and the status of the GPS card.
Preconditions	Correct configured measurement client (For details see Chapter 8 “Procedures for set-up of new trial sites”), installed GPS card (check if the green LED on the Meinberg GPS card is on, it shows that the antenna is connected correct and working), running measurement client software (“nettest”) and client testmodule (“ndm”)
Postconditions	If the “ndm” results are all ok, the measurement client is working correct.
Notes	If the green LED on the GPS card isn’t on, wait a few minutes and check again.

4.5.2.4 Router QoS Monitoring Tool

Activity No.	phase1.measurement.monitor.1
Title	QoS Monitoring Tool database connectivity
Description	The Monitoring Tool can connect to the measurement database, and store router parameters to the database.
Preconditions	Correctly configured MySQL database (phase1.measurement.db.1)
Postconditions	The results are correctly written to the database.
Notes	

4.5.2.5 GUI

Activity No.	phase1.measurement.gui.1
--------------	--------------------------

Title	Creation of the GUI
Description	The measurement management station is accessible via a graphical user interface. It is web-based and therefore a web-server with the necessary php-scripts have to be installed.
Preconditions	Apache server with PHP4-support / MySQL server / Activity phase1.measurement.db.1
Postconditions	Measurement management station is accessible via a web-browser. Users can specify traffic and flows.
Notes	

4.6 Router testing

4.6.1 QoS mechanisms implemented in the routers

This group of tests aims to check the implementation of available QoS mechanisms in the routers.

The following functionality was checked:

- Classification (CAR): The router can identify the traffic with the address, port, DSCP, protocol. Packets with different entries should be sent to the router.
- Marking: Related to the policer. The packets are marked with the DSCP with the conform and exceed action of the policer configuration.
- Policing and shaping: Conforming and non-conforming traffic to the traffic spec is marked with a different DSCP. In AQUILA we use for the first trial 3 different models (single token, dual token, single rate). The exceed action differs also for the five traffic classes. The router should be configured for the five different traffic classes and should be stimulated with different traffic. The non-conform behaviour should also be tested with the generator.
- Dropping (WRED): Simply send packets with different DSCPs to the router and test the dropping of the right packets. For two traffic classes we used drop-tail. For the others we used a 2-RED or a RED.
- Scheduling (priority, WFQ): The Premium VoIP traffic with CBR is prioritised. The other classes are scheduled with WFQ. The router should be stimulated with traffic of the different classes. With the analyser the right behaviour of the router should be controlled.
- RSVP signalling.

For those tests one router was sufficient.

4.6.2 Scope of the tests

The objective of the tests described in this chapter is to evaluate the commercially available routers that will be used in the AQUILA trial. The purpose of these tests is twofold: (1) to measure the performance of packet forwarding and (2) to examine implementation of DiffServ capabilities.

The first group of tests verifies packet-forwarding performance of Router Under Test (RUT). The main focus of these tests is to identify limits (e.g. throughput, latency etc.) within underlying equipment, which could have significant impact on the performance of the AQUILA network.

The aim of the second group of tests is to examine the implementation of DiffServ capabilities in the tested routers. The rationale for these tests is to verify assumptions made in the specification of AQUILA traffic handling mechanisms (see [D1301]). Recall that these assumptions correspond to such mechanisms as: traffic classification, policing (single and dual token bucket), queue management schemes (Drop Tail, WRED) and scheduling algorithms (WFQ, PQ).

The test plan presented below was repeated on all routers (i.e. Cisco 7507, 3600, 2500, 1605) and on all types of interfaces (i.e. Fast Ethernet, Ethernet, Serial, POS) that will be used for the first trial.

4.6.3 Test configuration and required measurement equipment

The recommended configuration for testing the router mechanisms and the router performance is depicted on Figure 3.

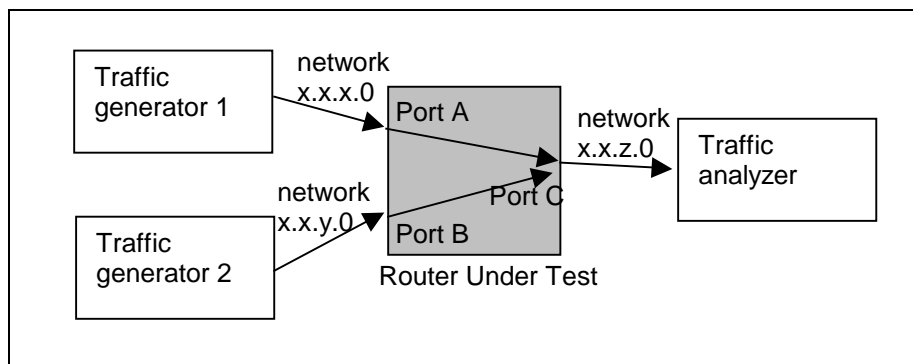


Figure 3. Recommended configuration for router testing

Two traffic generators (TG#1 and TG#2) submit test traffic to the router under test (RUT). The TG#1 (TG#2) generates foreground (background) traffic to port A (B). Traffic analyser (TA) is connected to the output port C.

The following measurement equipment were used to perform the tests:

- HP BSTS (traffic generator and analyser),
- Interwatch 95000 (traffic generator and analyser)

4.6.4 Router performance tests

The objective of the following tests is to verify the router performance used for AQUILA project. These tests were carried out according to the IP benchmarking methodology described in RFC 1944 and 1242. The tests were also carried out with the aid of the IP performance software implemented on the HP BSTS measurement equipment. The following three tests (from these defined in RFC 1944) were selected as the most important:

Throughput - the objective of the test is to determine the lossless *router throughput* as defined in RFC1944. This parameter is defined, as the maximum rate at which the number of IP packets sent to the router is equal to the number of packets received from it by the measurement instrument. Measurements should be taken for different frame sizes. The measurement procedure is based on binary search over the given range of transmission rates. First the minimum and maximum rates are tested to verify whether the throughput is within given values. Then in each step the middle of the rate range is tested. If there is no frame loss the actual throughput must be higher than the tested rate. New searching range is made up of the original higher rate and the actually tested rate. If frame losses were detected new searching range is formed by lower rate and the actually tested rate. This procedure is repeated continuously until the assumed accuracy is reached.

Test parameters	
Frame size range	Frame sizes to be used during test
Bandwidth utilization range	Minimum and maximum bandwidth utilization to be used to start the binary search procedure
Accuracy	Accuracy used to stop the binary search (defined by the utilization range that causes the searching procedure to stop)
Test duration	Maximum duration of test expressed in number of packets or real time

Latency - the objective of this test is to measure the packet (frame) delay introduced by the router. The latency is defined as the time interval starting when the last bit of the frame reaches the input port and ending when the last bit of the frame is seen on the output. Measurements should be taken for different frame sizes. The measurement procedure sends a stream of frames with given size at the lossless *throughput rate* (as measured by the throughput test). The packets should be sent to the router for at least 1 minute before the frame latency is meas-

ured. After this time one frame is selected for latency measurements. This procedure is repeated a number of times and the average value is calculated at the test end.

Latency test	
Frame size range	Frame sizes to be used during test
Bandwidth utilization	Throughput to be used in latency test. The throughput measured in the throughput test is recommended.
Test duration	Test duration (min 1 minute). The 10th last frame is used for measurement.
Number of repetitions	Number of repetition for the same frame size.

Frame Loss - the frame loss test measures the percentage of frames that are lost for a given throughput. Measurements are taken for different frame sizes and over given range of bandwidth utilization. The measurement procedure sends a specific number of frames at the given rate through the router and calculates the number of received frames. The difference between the send and received number of frames is the number of lost frames. The frame loss rate is calculated as the ratio between the number of lost frame and the number of transmitted frames.

Frame loss test	
Frame size range	Frame sizes to be used during test
Bandwidth utilization	Throughput range to be used in the test. It can be specified in steps of bandwidth utilization e.g. from 100% down to 10% in 10% steps.
Test duration	Test duration expressed in the number of packets or time units

The following test cases were executed:

Activity No.	phase1.router.performance.throughput
Title	Throughput test
Description	This test measures the maximum throughput between two selected interfaces the router under test can pass without frame loss. The measurement procedure should be repeated for different frame sizes. The frame sizes recommended for Ethernet should be used i.e. 64, 128, 256, 1024, 1280, 1518.
Preconditions	Measurement equipment and software available: HP BSTS with POS

	interface, IP performance software for HP BSTS The RUT is properly configured and operational
Postconditions	The throughput measured for each frame size should be recorded.
Notes	This test should be regarded as interface test. Testing the router performance, especially its switching fabrics, would require more traffic generators (one traffic generator per interface).

Activity No.	phase1.router.performance.latency
Title	Latency test
Description	This test measures the frame latency introduced by the router under test. The measurement procedure should be repeated for different frame sizes (see throughput test). For each frame size the maximum lossless throughput should be used.
Preconditions	See: phase1.router.performance.throughput test
Postconditions	The frame latency for each frame size should be recorded
Notes	See: phase1.router.performance.throughput test

Activity No.	phase1.router.performance.frameless
Title	Frame loss test
Description	This test measures the frame loss ratio for different router load conditions. The following interface load range should be tested i.e. 100%-10% in 10 percent steps. The test should be repeated for different frame sizes (see throughput test).
Preconditions	See: phase1.router.performance.throughput test
Postconditions	The frame loss for each frame size and throughput should be recorded
Notes	See: phase1.router.performance.throughput test

4.6.5 Tests of QoS mechanisms implemented in the routers

4.6.5.1 Packets classifier tests

The edge device should be able to classify incoming packets based on the values of different header fields: source and destination IP address, source and destination port number, protocol value and precedence bits. The aim of the group of tests presented in this section is to verify whether the classifier implemented in a tested router works properly and failures in packet classification do not occur.

In Cisco routers, the classifier function constitutes a part of the CAR (Committed Access Rate) mechanism. The matching criteria are specified in form of access-lists acting like filters for the incoming packets.

General testing procedure is as follows:

- An access-list should be created which contains a set of matching criteria for the incoming packets (see detailed test cases for the description of tested filters),

- The counters, which show statistics of matching packets should be cleared before executing the test case,
- Traffic generator sends stream of packets with a given constant rate. Submitted packets are classified on the input interface of Router Under Test.

The following test cases were executed:

Activity No.	phase1.router.mechanisms.classifier.1
Title	Classifier test 1
Description	Tests classifier based on precedence bits filter
Preconditions	The filter is configured on the router. About ½ of packets within the generated packet stream match the ‘access-list’ filter.
Postconditions	Expected result: Number of matching packets retrieved from the router statistics equals to the number of matching packets retrieved from the generator statistics.
Notes	If the result of the test case is as described in the ‘expected result’ section of the test procedure, the test is passed . Otherwise, the test is failed .

Activity No.	phase1.router.mechanisms.classifier.2
Title	Classifier test 2
Description	Tests classifier based on source address only. This filter corresponds to the ‘p2a’ reservation style, where the destination is not specified.
Preconditions	The filter is configured on the router. About ½ of packets within the generated packet stream match the ‘access-list’ filter.
Postconditions	Expected result: Number of matching packets retrieved from the router statistics equals to the number of matching packets retrieved from the generator statistics.
Notes	If the result of the test case is as described in the ‘expected result’ section of the test procedure, the test is passed . Otherwise, the test is failed .

Activity No.	phase1.router.mechanisms.classifier.3
Title	Classifier test 3
Description	Tests classifier based on: source and destination address, source and destination port number. This filter corresponds to the ‘p2p’ reservation style, where both endpoints are specified.
Preconditions	The filter is configured on the router. About ½ of packets within the generated packet stream match the ‘access-list’ filter.
Postconditions	Expected result: Number of matching packets retrieved from the router statistics equals to the number of matching packets retrieved from the generator statistics.
Notes	If the result of the test case is as described in the ‘expected result’ section of the test procedure, the test is passed . Otherwise, the test is failed .

Activity No.	phase1.router.mechanisms.classifier.4
Title	Classifier test 4
Description	Tests classifier based on: source and destination address, range of source and destination port numbers.
Preconditions	The filter is configured on the router. About ½ of packets within the generated packet stream match the ‘access-list’ filter.
Postconditions	Expected result: Number of matching packets retrieved from the router statistics equals to the number of matching packets retrieved from the generator statistics.
Notes	If the result of the test case is as described in the ‘expected result’ section of the test procedure, the test is passed . Otherwise, the test is failed .

Activity No.	phase1.router.mechanisms.classifier.5
Title	Classifier test 5
Description	Tests classifier based on: source address, source port number, and a set of destination addresses. This filter corresponds to p2m reservation style.
Preconditions	The filter is configured on the router. About ½ of packets within the generated packet stream match the ‘access-list’ filter.
Postconditions	Expected result: Number of matching packets retrieved from the router statistics equals to the number of matching packets retrieved from the generator statistics.
Notes	If the result of the test case is as described in the ‘expected result’ section of the test procedure, the test is passed . Otherwise, the test is failed .

Activity No.	phase1.router.mechanisms.classifier.6
Title	Classifier test 6
Description	Tests classifier based on: source and destination address, source and destination port number, protocol type, and precedence bits. It is a full classification filter used in ‘extended access-list’.
Preconditions	The filter is configured on the router. About ½ of packets within the generated packet stream match the ‘access-list’ filter.
Postconditions	Expected result: Number of matching packets retrieved from the router statistics equals to the number of matching packets retrieved from the generator statistics.
Notes	If the result of the test case is as described in the ‘expected result’ section of the test procedure, the test is passed . Otherwise, the test is failed .

4.6.5.2 Traffic conditioner tests

For traffic class TCL1 the single token bucket mechanism (STB) with dropping is used. In this class all conforming packets (“in profile”) are marked with DSCP=”110000xx” while non-conforming packets are simply dropped.

The aim of the STB mechanism test is to verify whether the router under test marks and drops packets correctly. The following cases should be tested:

- The traffic generated to the router follows the traffic contract. In this test all packets should be marked with correct code-point (as “in profile” packets) and no packets should be dropped.
- The traffic generated to the router does not conform to the assumed profile. In this case all “out of profile” packets should be dropped. The number of dropped packets should correspond to the level the generated traffic exceeds the traffic contract.

The following test cases should be executed:

Activity No.	phase1.router.mechanisms.conditioner.tcl1.1
Title	TCL1 traffic conditioning mechanism test 1
Description	In this test the conforming traffic is send to the router. The objective of the test is to verify that the conditioning mechanism does not drop any packet for the conforming traffic.
Preconditions	The testing equipment is available and connected to the router under test. The traffic classification and conditioning function (e.g. CAR mechanism in Cisco router) is configured according to the TCL1 class specification.
Postconditions	Expected results: the number of dropped packets should be zero
Notes	The test is passed if the number of dropped packet by the policer mechanism is equal zero otherwise it is failed

Activity No.	phase1.router.mechanisms.conditioner.tcl1.2
Title	TCL1 traffic conditioning mechanism test 2
Description	In this test the non-conforming traffic is send to the router. The objective of this test is to verify that the conditioning mechanism does drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the peak rate (PR parameter).
Preconditions	See router.conditioner.tcl1.1 test
Postconditions	Expected results: the number of dropped packets reflects the difference between the declared and actual value of peak rate descriptor
Notes	The above individual tests for non conforming traffic assume that only one parameter (traffic descriptor) is violated at a time. More complicated scenarios can also be tested with more that one parameter violated. The test is passed if the number of dropped packets corresponds to the level the traffic parameters of a test stream exceeds the traffic contract otherwise it is failed .

Activity No.	phase1.router.mechanisms.conditioner.tcl1.3
Title	TCL1 traffic conditioning mechanism test 3
Description	In this test the non-conforming traffic is send to the router. The objective of this test is to verify that the conditioning mechanism dose drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the bucket size for PR (x_1M_1)
Preconditions	See: phase1.router.conditioner.tcl1.1 test
Postconditions	Expected results: the number of dropped packets reflects the difference between the declared and actual value of bucket size for PR (x_1M_1)
Notes	See: phase1.router.conditioner.tcl1.2

Activity No.	phase1.router.mechanisms.conditioner.tcl1.4
Title	TCL1 traffic conditioning mechanism test 4
Description	The objective of this test is to verify that the conditioning mechanism does drop all packets whose length exceeds the maximum packet size (M_1 parameter). The stream of packets with x% of packets with length greater than M_1 should be generated to the RUT.
Preconditions	See: phase1.router.mechanism.conditioner.tcl1.1 test
Postconditions	Expected results: x% of packets should be dropped
Notes	See: phase1.router.mechanism.conditioner.tcl1.1 and router.mechanism.conditioner.tcl1.2 tests

The traffic conditioning for traffic class TCL2 is done by dual token bucket mechanism (DTB). The DTB distinguishes between “in” and “out of profile” packets based on the traffic descriptors. In case of TCL2 class the traffic conditioning mechanism should be configured in that way that all “in profile packets” are market with DSCP=”101000xx” while all “out of profile” packet are dropped.

The objective of the TCL2 traffic conditioning mechanism tests is to verify that the RUT correctly marks and drops packets. The following cases should be tested:

- The traffic generated to the RUT conforms to the traffic contract. In this test all sent packets should be marked with correct code point and no packets should be dropped.
- The traffic generated to the router does not conform to the assumed profile. In this case all “out of profile” packets should be dropped. The number of dropped packets should correspond to the level the generated traffic exceeds the traffic contract.

Activity No.	phase1.router.mechanisms.conditioner.tcl2.1
Title	TCL2 traffic conditioning mechanism test 1
Description	In this test the conforming traffic is send to the router. The objective of the test is to verify that the conditioning mechanism does not drop any packet for the conforming traffic.
Preconditions	The testing equipment is available and connected to the router under test. The RUT is properly configured and operational especially the traffic classification and conditioning function (e.g. CAR mechanism in Cisco

	router) is configured according to the TCL2 class specification.
Postconditions	Expected results: the number of dropped packets should be zero, all packets should be marked with code point DSCP="101000xx"
Notes	The test is passed if the expected results are met otherwise it fails

Activity No.	phase1.router.mechanisms.conditioner.tcl2.2
Title	TCL2 traffic conditioning mechanism test 2
Description	In this test the non-conforming traffic is sent to the router. The objective of this test is to verify that the conditioning mechanism drops correct number of packet from the incoming packet stream. The test packets stream does not conform to the peak rate (PR parameter).
Preconditions	See phase1.router.mechanism.conditioner.tcl2.1 test
Postconditions	Expected results: all received packets are marked with code point DSCP="101000xx", the number of dropped packets is proportional to the difference between the declared and actual value of peak rate descriptor
Notes	The above tests for non-conforming traffic assume that only one parameter (traffic descriptor) is violated at a time i.e. peak rate. More complicated scenarios can also be tested with more than one parameter violated. The test is passed if the number of dropped packets corresponds to the level the traffic parameters of a test stream exceeds the traffic contract otherwise it is fails .

Activity No.	phase1.router.mechanisms.conditioner.tcl2.3
Title	TCL2 traffic conditioning mechanism test 3
Description	In this test the non-conforming traffic is send to the router. The objective of this test is to verify that the conditioning mechanism does drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the sustainable rate (SR parameter)
Preconditions	See phase1.router.mechanism.conditioner.tcl2.1 test
Postconditions	Expected results: all received packets are marked with code point DSCP="101000xx", the number of dropped packets is proportional to the difference between the declared and actual value of sustainable rate descriptor
Notes	See phase1.router.mechanism.conditioner.tcl2.2 test

Activity No.	phase1.router.mechanisms.conditioner.tcl2.4
Title	TCL2 traffic conditioning mechanism test 4
Description	In this test the non-conforming traffic is send to the router. The objective of this test is to verify that the conditioning mechanism does drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the bucket size for PR (x_2M_2)
Preconditions	See phase1.router.mechanism.conditioner.tcl2.1 test
Postconditions	Expected results: all received packets are marked with code point DSCP="101000xx", the number of dropped packets is proportional to the difference between the declared and actual value of bucket size for PR

Notes	See phase1.router.mechanism.conditioner.tcl2.2 test
-------	---

Activity No.	phase1.router.mechanisms.conditioner.tcl2.5
Title	TCL2 traffic conditioning mechanism test 5
Description	In this test the non-conforming traffic is send to the router. The objective of this test is to verify that the conditioning mechanism does drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the bucket size for SR (BSS parameter)
Preconditions	See phase1.router.mechanism.conditioner.tcl2.1 test
Postconditions	Expected results: all received packets are marked with code point DSCP="101000xx", the number of dropped packets is proportional to the difference between the declared and actual value of BSS descriptor
Notes	See phase1.router.mechanism.conditioner.tcl2.2 test

For traffic class TCL3 the single token bucket mechanism (STB) with marking is used. In this class the “in-profile” packets are marked with DSCP code point “10000xx” and the “out of profile” are marked with code point “011000xx”, respectively.

The aim of the STB mechanism test is to verify that the router under test correctly marks packets for TCL3. The following cases should be tested:

- The generated traffic to the router conforms to the traffic contract. In this test all sent packets should be marked with correct code point (“in profile” packets) and no packets should be dropped or marked as “out of profile” packets.
- The traffic generated to the router doesn’t conform to the assumed profile. In this case all “out of profile” packets should be marked with DSCP=“”. The number of dropped (or marked) packets should correspond to the level the generated traffic exceeds the traffic contract.

The following test cases should be executed:

Activity No.	phase1.router.mechanisms.conditioner.tcl3.1
Title	TCL3 traffic conditioning mechanism test 1
Description	In this test the conforming traffic is send to the router. The objective of the test is to verify that the conditioning mechanism does not drop any packet for the conforming traffic.
Preconditions	The testing equipment is available and connected to the router under test. The traffic classification and conditioning function (e.g. CAR mechanism in Cisco router) is configured according to the TCL3 class specification.
Postconditions	Expected results: the number of dropped packets should be zero
Notes	The test is passed if the number of dropped packet by the policer mechanism is equal zero otherwise it is failed

Activity No.	phase1.router.mechanisms.conditioner.tcl3.2
Title	TCL3 traffic conditioning mechanism test 2
Description	In this test the non-conforming traffic is send to the router. The objective

	of this test is to verify that the conditioning mechanism dose drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the sustainable rate (SR parameter)
Preconditions	See phase1.router.conditioner.tcl3.1 test
Postconditions	Expected results: the number of marked packets reflects the difference between the declared and actual value of sustainable rate descriptor
Notes	The above individual tests for non conforming traffic assume that only one parameter (traffic descriptor) is violated at a time. More complicated scenarios can also be tested with more that one parameter violated. The test is passed if the number of marked packets corresponds to the level the traffic parameters of a test stream exceeds the traffic contract otherwise it is failed .

Activity No.	phase1.router.mechanisms.conditioner.tcl3.3
Title	TCL3 traffic conditioning mechanism test 3
Description	In this test the non-conforming traffic is send to the router. The objective of this test is to verify that the conditioning mechanism dose drop correct number of packet from the incoming packet stream. The test packet stream does not conform to the bucket size for SR (BSS parameter)
Preconditions	See phase1.router.conditioner.tcl3.1 test
Postconditions	Expected results: the number of dropped packets reflects the difference between the declared and actual value of BSS descriptor
Notes	See phase1.router.conditioner.tcl3.2

The traffic conditioning for traffic class TCL4 is done by dual token bucket mechanism (DTB). The DTB distinguishes between “in” and “out of profile” packets based on the traffic descriptors. In case of TCL4 class the traffic conditioning mechanism should mark all “in profile packets” with DSCP=”010000xx” while all “out of profile” packets with DSCP=”001000xx”.

The following tests verify that the router under test correctly marks packets. The following cases should be tested:

- The traffic generated to the router conforms to declared values of traffic descriptors. In this case all sent packets should be marked with code point DSCP=”010000xx”
- The traffic generated to the router exceeds one more traffic descriptor. In this case all “out of profile” packets should be marked with DSCP=”001000xx”. The number of packets with DSCP=”001000xx” should correspond to the level the generated traffic exceeds the assumed parameters.

The following test cases should be executed:

Activity No.	phase1.router.mechanisms.conditioner.tcl4.1
Title	TCL4 traffic conditioning mechanism test 1
Description	In this test the conforming traffic is send to the router. The objective of the test is to verify that the conditioning mechanism correctly marks “in

	profile” packets.
Preconditions	The testing equipment is available and connected to the router under test. The RUT is properly configured and operational especially the traffic classification and conditioning function (e.g. CAR mechanism in Cisco router) is configured according to the TCL4 class specification.
Postconditions	Expected results: no packets should be dropped, all packets should be marked with code point DSCP=”010000xx”
Notes	The test is passed if the expected results are met otherwise it fails

Activity No.	phase1.router.mechanisms.conditioner.tcl4.2
Title	TCL4 traffic conditioning mechanism test 2
Description	The objective of this test is to verify that the conditioning mechanism correctly marks “in-profile” and “out-of profile” packets. In this test the traffic sent to the router is not conforming to the peak rate parameter (PR).
Preconditions	See phase1.router.mechanism.conditioner.tcl4.1 test
Postconditions	Expected results: The number of “out of profile” packets (marked with DSCP=”001000xx”) should be proportional to the difference between actual and declared value of PR parameter
Notes	The above test assumes that only one parameter (traffic descriptor) is violated at a time. More complicated scenarios can also be tested with more that one parameter violated. The test is passed if the expected results are met otherwise it fails

Activity No.	phase1.router.mechanisms.conditioner.tcl4.3
Title	TCL4 traffic conditioning mechanism test 3
Description	The objective of this test is to verify that the conditioning mechanism correctly marks “in-profile” and “out-of profile” packets. In this test the traffic sent to the router is not conforming to the sustainable rate parameter (SR).
Preconditions	See phase1.router.mechanism.conditioner.tcl4.1 test
Postconditions	Expected results: The number of “out of profile” packets should be proportional to the difference between actual and declared value of SR parameter
Notes	See phase1.router.mechanism.conditioner.tcl4.2 test

Activity No.	phase1.router.mechanisms.conditioner.tcl4.4
Title	TCL4 traffic conditioning mechanism test 4
Description	The objective of this test is to verify that the conditioning mechanism correctly marks “in-profile” and “out-of profile” packets. In this test the traffic sent to the router is not conforming to the bucket size for peak parameter (x_4M_4).
Preconditions	See phase1.router.mechanism.conditioner.tcl4.1 test
Postconditions	Expected results: The number of “out of profile” packets should be proportional to the difference between actual and declared value of the bucket size for PR parameter (x_4M_4)

Notes	See phase1.router.mechanism.conditioner.tcl4.2 test
-------	---

Activity No.	phase1.router.mechanisms.conditioner.tcl4.5
Title	TCL4 traffic conditioning mechanism test 5
Description	The objective of this test is to verify that the conditioning mechanism correctly marks “in-profile” and “out-of profile” packets. In this test the traffic sent to the router is not conforming to the bucket size for sustainable rate parameter (BSS).
Preconditions	See phase1.router.mechanism.conditioner.tcl4.1 test
Postconditions	Expected results: The number of “out of profile” packets should be proportional to the difference between actual and declared value of BSS parameter
Notes	See phase1.router.mechanism.conditioner.tcl4.2 test

4.6.5.3 Queue management tests

Two types of queue management mechanisms are used in AQUILA: Tail Drop and WRED. WRED (Weighted Random Early Detection) is a mechanism, which randomly drops packets when queue is congested. When queue size is above the specified threshold, packet arriving in a queue can be dropped with a certain probability. Different thresholds are set for packets with different value of precedence bits, e.g. packets marked as ‘out’ may be dropped earlier than packets marked as ‘in’ (‘in’ and ‘out’ means a specified value of precedence bits).

General testing procedure is as follows:

- Traffic generator generates stream of packets with a constant rate, not greater than interface rate. 50% of packets are marked as ‘in’, and the rest is marked as ‘out’. Note, that packets are marked as ‘in’ or ‘out’ not as a result of policing on the input port, but they are marked with a proper precedence value already in the traffic generator. No packet losses should be observed at this stage.
- Background traffic is added on a tested output port. Background traffic is increased in small increments, in order to cause congestion on the output port. Packet loss rate is measured for ‘in’ and ‘out’ packets. WRED mechanism should cause dropping of packets, while packet loss rate for ‘out’ packets should be higher than for ‘in’ packets.

The following test cases should be executed:

Activity No.	phase1.router.mechanisms.queue.1
Title	Tail Drop mechanism test
Description	This test case verifies that when output port is congested, the tail drop mechanism drops packets marked as ‘out’ equally often as packets marked as ‘in’.
Preconditions	Tail drop function should be enabled on an interface of Router Under Test.
Postconditions	Tail drop mechanism should cause dropping of packets, while packet loss rate for ‘out’ packets should be the same as for ‘in’ packets.

Notes	This test checks the Tail Drop mechanism configured for an interface. This test can be also performed in a configuration with a few traffic classes and Tail Drop configured for a specific traffic class.
Activity No.	phase1.router.mechanisms.queue.2
Title	WRED mechanism test
Description	This test case verifies that when output port is congested, WRED (2-RED) mechanism drops packets marked as 'out' more often then packets marked as 'in'.
Preconditions	WRED function should be enabled on an interface of Router Under Test. Threshold values for 'in' packets should be set higher then threshold values for 'out' packets.
Postconditions	WRED mechanism should cause dropping of packets, while loss rate for 'out' packets should be higher then for 'in' packets.
Notes	This test checks the WRED mechanism configured for an interface. This test can be also performed in a configuration with a few traffic classes and WRED configured for a specific traffic class.

4.6.5.4 Scheduler tests

From the AQUILA viewpoint the most important scheduling algorithms are Priority Queuing (PQ) and Weighted Fair Queuing (WFQ). Recall that the objective of the scheduling algorithms is to control access to the link. Therefore, the performance of schedulers is a base for QoS provisioning in AQUILA network.

In general, the studies of scheduler performance concern the following properties:

- isolation of flows, which means that sources sending packets at a higher rate than negotiated should not affect the guaranteed rates of the other sources.
- fairness, which means that available bandwidth must be divided among the flows sharing the link in a fair manner. That is if flow uses less than its negotiated rate, the unused bandwidth must be fair shared among the other flows.
- bandwidth utilisation, so the algorithm must utilise the bandwidth very efficiently and should not lead to network under-utilisation.
- low delay: it means that the algorithm must provide delay guarantees for the isolated flow, which should depend only on bandwidth reservation

The above listed properties of the scheduling algorithms one can evaluate by measurements of QoS parameters like throughput, packet loss rate and packet delay in different time scales. Unfortunately, the measurement equipment available in TPS allows us to measure above parameters in large time scale only. Therefore, the tests of scheduler can be performed based on mean values.

General testing procedure is as follows:

At first, only foreground flow (with appropriate traffic descriptors) is generated, and values of throughput, packet loss and packet delay are measured. Then, the background traffic is added and its impact on QoS parameters is evaluated. The last step in proposed tests is related with evaluation of scheduler performance based on obtained results as well as theoretical ones.

Remark that more attention should be paid on specification of appropriate foreground and background traffic description (e.g. traffic patterns, packet length distribution) as well as parameters of schedulers (priority, weights, queue sizes, etc.).

Activity No.	phase1.router.mechanisms.scheduler.1
Title	PQ scheduler test
Description	<p>In this test the performance of PQ scheduler will be evaluated. The main issue concerns the influence of lower priority traffic on higher priority traffic (isolation, bandwidth utilisation and low delay). In the case of PQ the fairness is usually not taken into account.</p> <p>At first, QoS parameters should be measured for foreground traffic only, at rate about e.g. 50 % of output bandwidth. Next the background traffic should be added at rate about e.g. 70%, so that the total offered traffic should be greater than 100% of output bandwidth. Now, the QoS parameters should be measured one again.</p>
Preconditions	<p>The following configuration actions should be performed:</p> <ul style="list-style-type: none"> the proper setting of PQ parameters, it means priority and size of queues at router output port appropriate setting of traffic generators (traffic descriptor and class)
Postconditions	<p>We expect that the background flow has minor influence on foreground flow. It means that no foreground packets will be lost, and foreground packet delay will slightly increase (just about time required for transferring one packet, only). In the case of background traffic, some packets should be lost (about 20%). Also, the output link utilisation should be close to 100 %.</p>
Notes	

Activity No.	phase1.router.mechanisms.scheduler.2
Title	WFQ scheduler test #1
Description	<p>In this test the performance of WFQ scheduler will be evaluated. Especially the issues related with flow isolation, bandwidth utilisation and low delay are taken into account.</p> <p>At first, QoS parameters should be measured for foreground traffic only, generating at rate equal to the weight assigned for foreground traffic. In the next step, the background traffic should be added generating at rate equal to the weight assigned for background traffic and QoS parameters should be measured. Next, the background traffic should be increased above its weights.</p>
Preconditions	<p>The following configuration actions should be performed:</p> <ul style="list-style-type: none"> the proper setting of WFQ parameters, it means weights and size of queues at router output port

	<ul style="list-style-type: none"> appropriate setting of traffic generators (traffic descriptor and class)
Postconditions	<p>We expect that in the first step foreground packets will be served with link rate and no packets will be lost. In the second step both foreground and background packets should be served with rate not less than assigned weights, and also no packets should be lost. In the last test the foreground packets should be served with rate not less than assigned weight, and no packets should be lost, while for background traffic same packets can be lost.</p>
Notes	

Activity No.	phase1.router.mechanisms.scheduler.3
Title	WFQ scheduler test #2
Description	<p>In this test the performance of WFQ scheduler will be evaluated. Especially the issues related with fairness are taken into account. This test requires two identical flows. (Two foreground flows).</p> <p>At first, QoS parameters should be measured for both identical foreground flows, generated at rate equal to assigned weight (much less than 0.5, e.g 0.3). In the next step, the both foreground flows should be equally increased, in order to obtain overload conditions, and QoS parameters should be measured once again.</p>
Preconditions	<p>The following configuration actions should be performed:</p> <ul style="list-style-type: none"> the proper setting of WFQ parameters, it means weights and size of queues at router output port appropriate setting of traffic generators (traffic descriptor and class)
Postconditions	<p>We expect that the QoS parameters for both foreground flows will be the same. Moreover one can expected that in low load conditions no packets would be lost in contrast to overload conditions.</p>
Notes	

4.6.5.5 Other tests

4.6.5.5.1 Maximum number of flows in RUT

Establishing new flow reservation in an edge router requires configuring classifier and policer for that flow. Classifying and policing functions are processor-intensive operations. When the number of flows is large, classifying each incoming packet requires performing look-up operation in large tables of matching criteria. Establishing too many flows on a router port may result in performance degradation or even incorrect functionality of classifier and policer. This test should determine the maximum number of flows that can be established on a tested router port without negative impact on performance of these operations.

General testing procedure is as follows:

- At the beginning a single flow is established ($n=1$). Classifier and policer should be configured for this flow on the input port of RUT,

- The traffic generator generates n streams of packets that match the classifying criteria. Traffic analyser measures packet delay and packet loss ratio,
- The number of flows configured in a router and number of streams generated in a traffic generator is increased ($n+1$) and the previous step is repeated until increased packet delay or packet loss ratio is observed.

Activity No.	phase1.router.others.1
Title	Test for maximum number of flows
Description	This test determines the maximum allowed number of flows that guarantees well behaviour of classifier and policer.
Preconditions	The RUT is properly configured and operational
Postconditions	
Notes	The test measures the maximum number of flows that can be set in the RUT.

4.6.6 Test results

4.6.6.1 CISCO 7507 router tests

4.6.6.1.1 Router configuration

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
Interfaces	4 Ethernet/IEEE 802.3 interface(s) 2 Fast Ethernet/IEEE 802.3 interface(s) 2 ATM network interface(s) 3 Packet over SONET network interface(s)

4.6.6.1.2 Test configurations

The configuration used for testing the router mechanisms and the router performance is depicted on Figure 4.

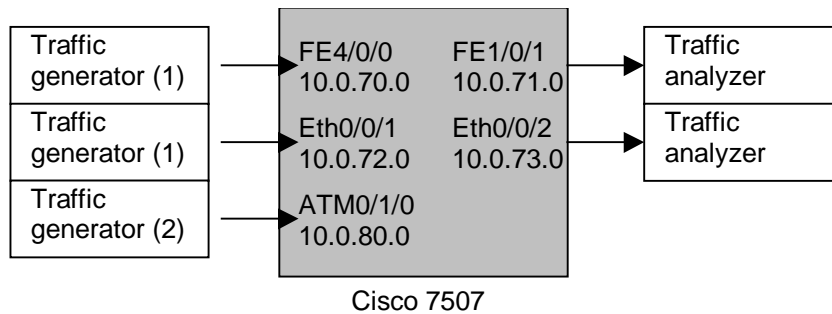


Figure 4. Configuration for 7507 router testing

4.6.6.1.3 Router performance testing

The router performance tests were done with the IP performance software available on the HP BSTS measurement equipment. The test configuration is depicted on Figure 4. The generating and receiving ports of the HP BSTS were connected to the router under test (RUT) via Fast Ethernet interfaces FE4/0/0 and FE1/0/1.

4.6.6.1.3.1 Router performance throughput test

In the throughput test the maximum lossless transmission rate between input and output port of the Cisco 7507 router as a function of frame size was measured. One can observe that for small frames e.g. 64 bytes the router performance, depending on router switching mode, is limited to 25-45 Mbps (50000-80000 packets per second). These results show the packet switching performance of the Cisco 7507 series routers. The interpretation of these results depends on the switching mode used in the tests. In CEF mode this result presents the performance of central routing processor (RSP) and thus the total number of packets the router can forward. In the distributed CEF mode the packet switching is distributed between VIP processors thus the obtained results show the packet forwarding rate of single VIP processor. The router performance is thus the sum of the performance of all VIP processors. In this mode all ports served by the same VIP processor share its packets forwarding rate. The power of the VIP processor of the 7507 router used in AQUILA TPS testbed routers is slightly less the power of the RSP processor. The VIP can forward roughly about 80000 packets per second. For longer frames the router throughput, in the tested configuration, is limited by the interface speeds.

In the current CISCO router architecture the QoS mechanisms are implemented in software so they use up some of the processors power which results in decreasing packet switching performance. For example the WFQ scheduling mechanism decreases the packets forwarding rate by 30%, to about 50000 packets per second.

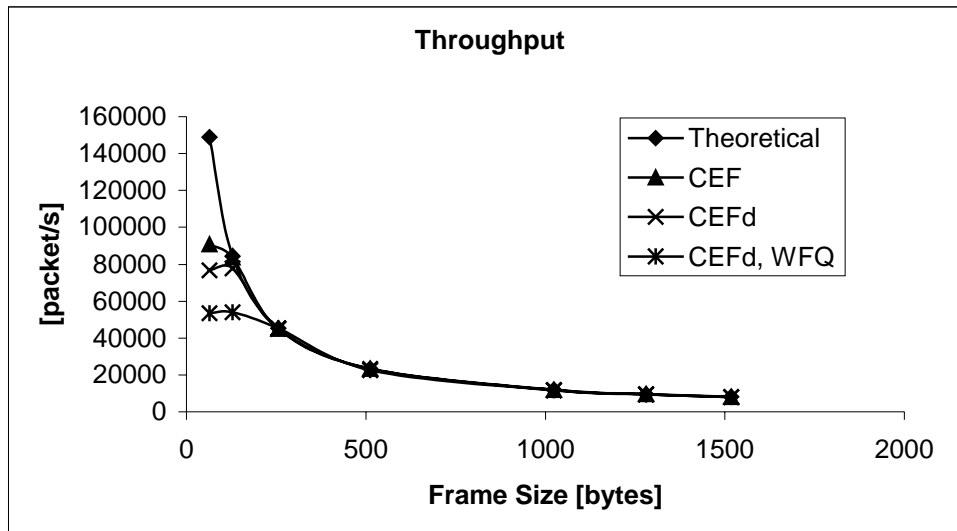


Figure 5. Throughput of the Cisco 7507 router

4.6.6.1.3.2 Router.performance.frame loss

In the frame loss test the frame loss ratio of the Cisco 7507 router as a function of router load was measured. The obtained results explain the router performance degradation observed in the throughput test. When the router load reaches the number of packets the RSP or VIP processor can forward frame loss occurs.

As can be seen from Figure 8 the router performance is severely limited by the activation of CBWFQ scheduling mechanism. In the case of small packets, frame losses occur when the Fast Ethernet interface load increases above 30 percent. For longer frames the losses occur for load above 90 percent. However, one has to remember that this performance can be reached if one VIP processor serves only one Fast Ethernet port.

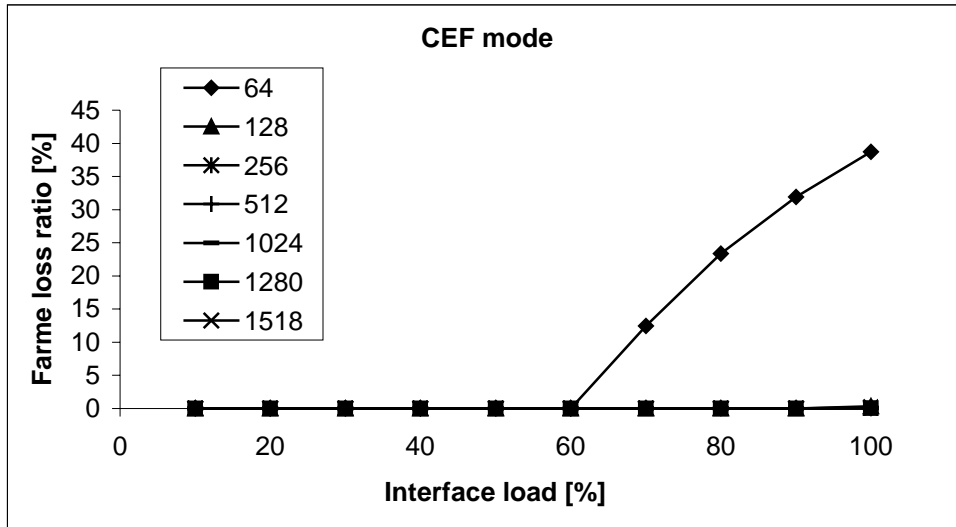


Figure 6. Frame loss of Cisco 7507 router in the CEF mode

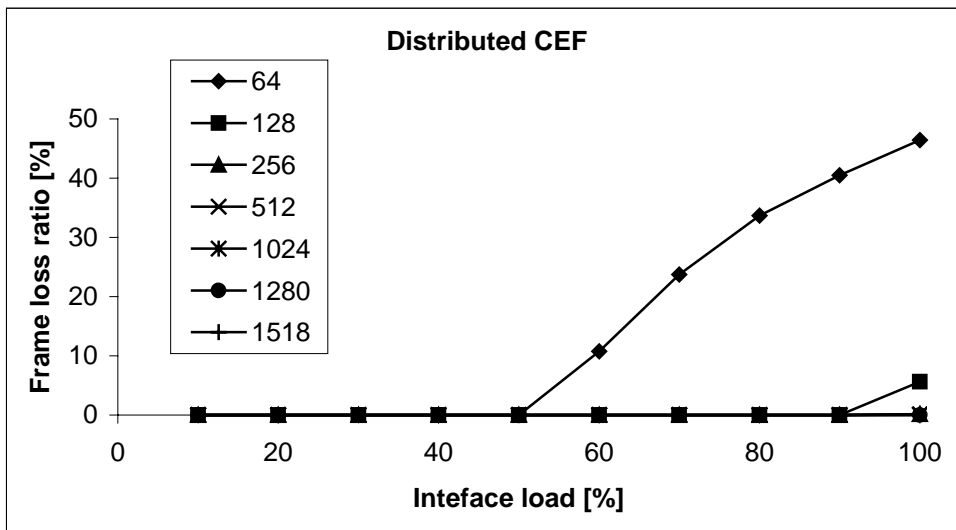


Figure 7. Frame loss of Cisco 7507 router in the distributed CEF mode

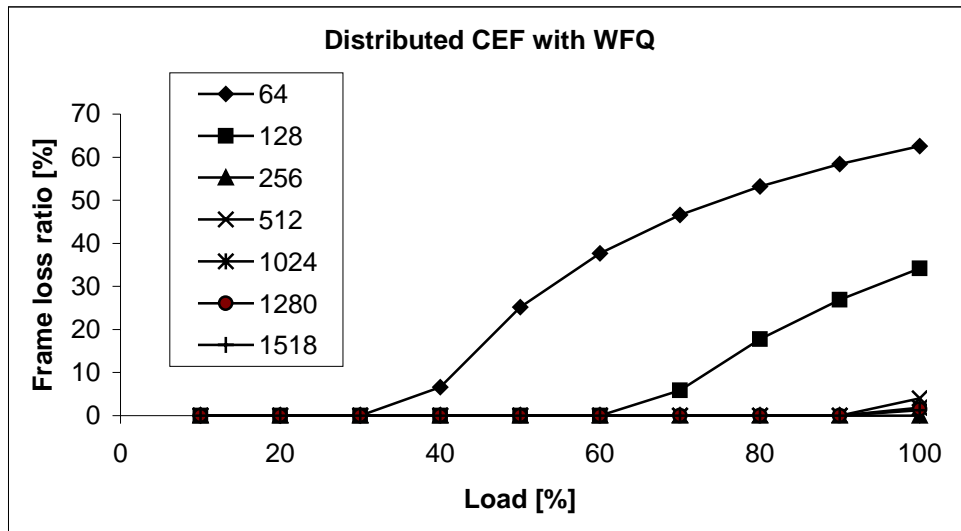


Figure 8. Frame loss of Cisco 7507 router in the distributed CEF mode with WFQ

4.6.6.1.3.3 Router.performance.latency

In the latency test the average frame latency was measured in different router switching modes. The measurements were done for different frame lengths on lightly loaded router. The results show the minimum latency introduced by Cisco 7507 series router. The CBWFQ mechanism increases the frame processing delay and thus the observed frame latency. Nevertheless the observed latency is below 0.5 ms.

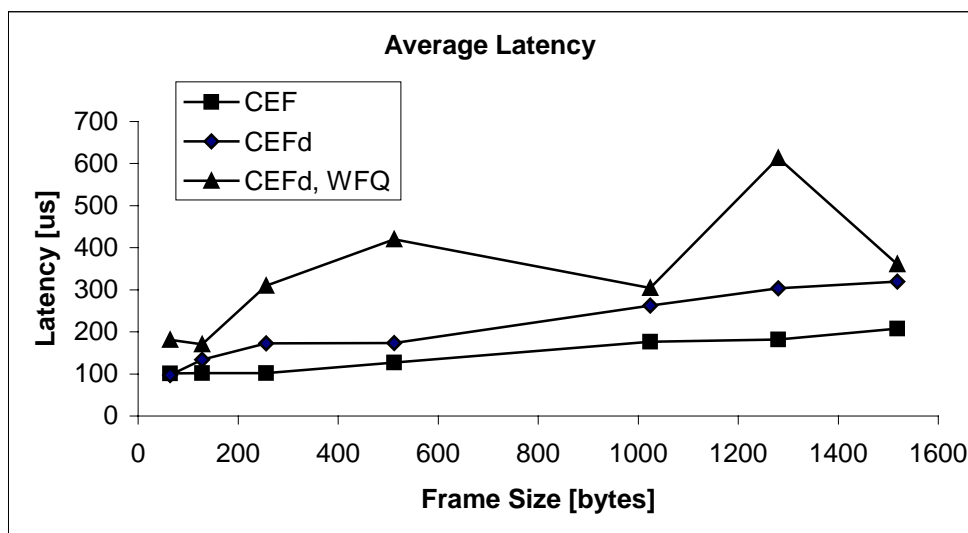


Figure 9. Frame latency of Cisco 7507 router

4.6.6.1.4 Traffic Classifier testing

Packet classifier tests were performed in configuration depicted on Figure 4. Port FE4/0/0 of the RUT, where traffic was submitted from the generator, had IP address 10.0.70.1. Port FE1/0/0 had IP address 10.0.71.1. Both ports were Fast Ethernet. Two traffic streams were sent to the router. During the test each stream generated a total of 200000 packets.

- Stream1: 30000kb/s, packets belonging to this stream did not match the criteria specified in the access list.
- Stream2: 30000kb/s, packets belonging to this stream matched the criteria specified in the access list.

Test router.mechanisms.classifier.1

```
show access-list counters:  
  Extended IP access list 105  
    permit ip any any precedence critical (200000 matches)
```

The result of 'show access-lists' command executed after the test shows that number of packets matching the classifier criteria was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier.2

```
matches: access-group 50  
  params: 40000000 bps, 20000 limit, 20000 extended limit  
  conformed 199921 packets, 209917050 bytes; action: continue  
  exceeded 79 packets, 82950 bytes; action: continue  
  last packet: 19272ms ago, current burst: 10500 bytes  
  last cleared 00:01:33 ago, conformed 17895000 bps, exceeded 7000 bps
```

The result of 'show int rate' command executed after the test shows that number of packets matching the classifier criteria was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier.3

```
Extended IP access list 105  
  permit tcp host 10.0.70.3 eq 100 host 10.0.71.3 eq 100 (200000 matches)
```

The result of 'show access-lists' command executed after the test shows that number of packets matching the classifier criteria was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier.4

```
Extended IP access list 105  
  permit tcp host 10.0.70.3 range 100 150 host 10.0.71.3 range 100 150  
(200000 matches)
```

The result of 'show access-lists' command executed after the test shows that number of packets matching the classifier criteria was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier.5

```
Extended IP access list 105  
  permit ip host 10.0.70.2 10.0.71.0 0.0.0.255 (200000 matches)
```

Packets generated in stream 2 had destination IP addresses in a range of 10.0.72.20-10.0.72.220. The result of 'show access-lists' command executed after the test shows that number of packets matching the classifier criteria was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier.6

```
Extended IP access list 105
 permit ip host 10.0.70.2 eq 100 10.0.71.0 eq 100 precedence critical
(200000 matches)
```

4.6.6.1.5 Traffic conditioning mechanism testing

The Cisco 7507 router offers CAR mechanism for traffic policing and marking that can be used to implement AQUILA traffic conditioning functions. With CAR mechanism the user traffic can be characterised by three parameters: traffic rate, burst size and excess burst size. The third parameter implements additional features in comparison to the standard token bucket algorithm. In the following tests this feature was disabled by setting the excess burst size equal to burst size. The Cisco routers do not offer double token bucket algorithm as separate mechanism. The dual token bucket can be obtained by appropriate configuration of two CAR mechanisms (two single token buckets).

In the following test the CAR was configured as single token bucket with dropping. Notice that these settings correspond to TCL1 conditioning function in AQUILA network architecture. The following parameters were set:

- Traffic rate=96 kbps
- Burst size=2000 bytes (no smaller value of bucket size than 2000 bytes is provided by CAR)

The following configuration was set on the test router:

```
aq7507_2#sh interfaces rate
FastEthernet4/0/0
  Input
    matches: access-group 111
      params: 96000 bps, 2000 limit, 2000 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 4
      exceeded 0 packets, 0 bytes; action: set-prec-transmit 3
      last packet: 811396ms ago, current burst: 0 bytes
      last cleared 00:13:11 ago, conformed 0 bps, exceeded 0 bps
```

4.6.6.1.5.1 Router.mechanisms.conditioning.tcl1.1 test

In this test the traffic conforming to the token bucket parameters was generated. The number of dropped packets, in one-minute observation interval, was recorded. The measurement procedure was repeated for different IP packet lengths i.e. 64, 256, 512, 128 and 1500 bytes.

The test results are included in table 1 (transmission rate 96 kbps). The CAR mechanism monitors the transmission rate on the frame (MAC) level. In case of conforming traffic the CAR mechanism works correctly (no packet drop was observed).

4.6.6.1.5.2 Router.mechanism.conditioning.tcl1.2 test

In this test non-conforming traffic was generated. The test traffic exceeds the CAR transmission rate of 96 kbps. The measurement procedure was repeated for different traffic rates and IP packet lengths. The number of dropped packets, in one-minute observation interval, was recorded.

The test results are presented in Table 1 and Table 2 in the form of the number of dropped packet in one minute measurement interval and packet dropping rate. The CAR mechanism works correctly. As can be seen from Figure 10 the number of dropped packet is proportional to the difference between the declared and actual transmission rate. Moreover, the dropping rate does not depend on the packet length (only on the transmission rate).

Table 1. Number of dropped packets in one-minute measurement interval

MAC rate	IP rate	64	256	512	1024	1500
96	78.8	0	0	0	0	0
96.1	78.9	10	3	1	1	1
96.5	79.2	48	14	7	4	3
97	79.6	96	28	14	7	5
98	80.4	192	55	29	14	9
99	81.2	280	79	43	21	15
100	82.1	385	111	66	29	20
105	86.2	866	250	128	65	45
110	90.3	1346	389	183	101	70

Table 2. Packet dropping rate

MAC rate	IP rate	64	256	512	1024	1500
96	78.8	0	0	0	0	0
96.1	78.9	0.00108	0.00112	0.00073	0.00144	0.00210
96.5	79.2	0.00517	0.00522	0.00509	0.00574	0.00628
97	79.6	0.01029	0.01039	0.01012	0.00999	0.01041
98	80.4	0.02038	0.02020	0.02075	0.01977	0.01854
99	81.2	0.02941	0.02873	0.03046	0.02936	0.03059
100	82.1	0.04004	0.03996	0.04629	0.04014	0.04037
105	86.2	0.08578	0.08571	0.08550	0.08568	0.08651
110	90.3	0.12726	0.12731	0.11668	0.12708	0.12846

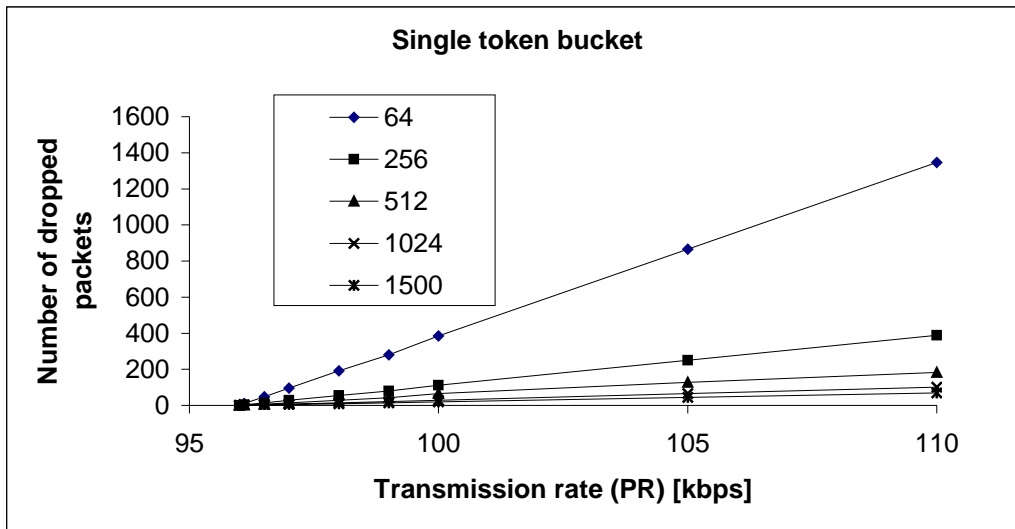


Figure 10. Number of dropped packets

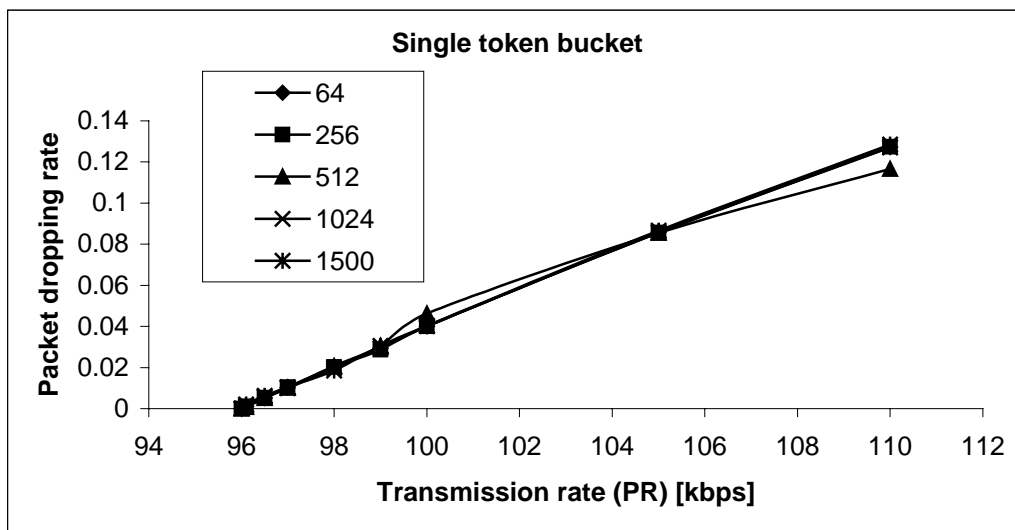


Figure 11. Packet dropping rate

4.6.6.1.5.3 Router.mechanisms.conditioning.tcl1.3 test

In this test the performance of the packet burst size policing mechanism (CAR) of the Cisco 7507 router was tested. The on-off test traffic with burst length (50 packets) exceeding the token bucket size was submitted to the router. During On period packets were transmitted with the full interface speed. The off period was long enough to allow for full bucket “regeneration”. The committed burst size was calculated from the measured number of transmitted and dropped packets. The measurements were repeated for different frame sizes.

The measured results were compared with the theoretical conforming burst sizes of the single token bucket mechanism with the given parameters. As can be seen from Figure 12 the CAR mechanism allows for longer bursts than standard token bucket algorithm (the curve Token bucket (quantization)). The conforming burst sizes of CAR mechanism are one packet longer for all packet lengths.

Table 3. Conforming burst sizes

Packet length [Bytes]		2000-1001	1000-667	666-501	500-401	400-334	333-286	285-251	250-223	222-201
Conforming burst size [Packets]		2	3	4	5	6	7	8	9	10
Packet length [Bytes]	200-182	181-167	166-154	153-143	142-134	133-126	125-118	117-112	111-106	105-101
Conforming burst size [Packets]	11	12	13	14	15	16	17	18	19	20

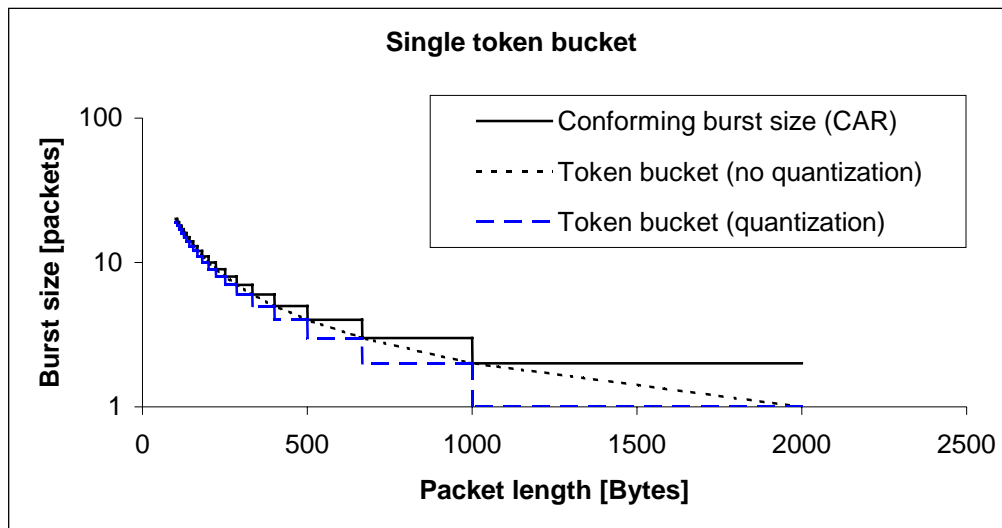


Figure 12. Conforming burst sizes

Remark: Exactly the same results were obtained for CAR mechanism configured as single token bucket with marking (conditioning mechanism for class TCL3). The dropping rate represents the packets marking rate for class TCL3 conditioning mechanism. The conditioning mechanisms for classes TCL2 and TCL4 were not tested separately as they are implemented by two CAR mechanisms.

4.6.6.1.6 Queue management tests

4.6.6.1.6.1 Router.mechanism.queue.2

Weighted Random Early Detection mechanism was tested on the Fast Ethernet port of Cisco 7507 router. The test configuration was as depicted on Figure 4. The WRED mechanism was configured as follows:

Current random-detect configuration:

```
FastEthernet1/0/0
  Queueing strategy: fifo
  Packet drop strategy: VIP-based random early detection (DWRED)
  Exp-weight-constant: 9 (1/512)
  Mean queue depth: 0
  Queue size: 0          Maximum available buffers: 25600
  Output packets: 30947 WRED drops: 0 No buffer: 0
```

Class	Random drop	Tail drop	Minimum threshold	Maximum threshold	Mark probability	Output Packets
0	0	0	6400	12800	1/10	30946
1	0	0	7200	12800	1/10	0
2	0	0	2000	6800	1/5	0
3	0	0	8000	12800	1/5	0
4	0	0	9600	12800	1/10	0
5	0	0	10400	12800	1/10	0
6	0	0	11200	12800	1/10	0
7	0	0	12000	12800	1/10	0

Two traffic streams were submitted to the router,

Stream1: constant bit rate, 38616kb/s, packet length 512B, precedence 2 (WRED: min threshold 2000, max threshold 6800)

Stream2: constant bit rate, 38616kb/s, packet length 512B, precedence 3 (WRED: min threshold 8000, max threshold 12800)

Additionally, background traffic was generated in order to overload the output port. When the mean queue size of the output port increased above the min threshold value, the WRED mechanism began to drop packets. As different thresholds were set for packets with precedence 2 and 3, the drop behaviour was different for two submitted traffic streams. Figure 13 shows packet loss ratio as a function of mean queue size for packets belonging to Stream1 and Stream2.

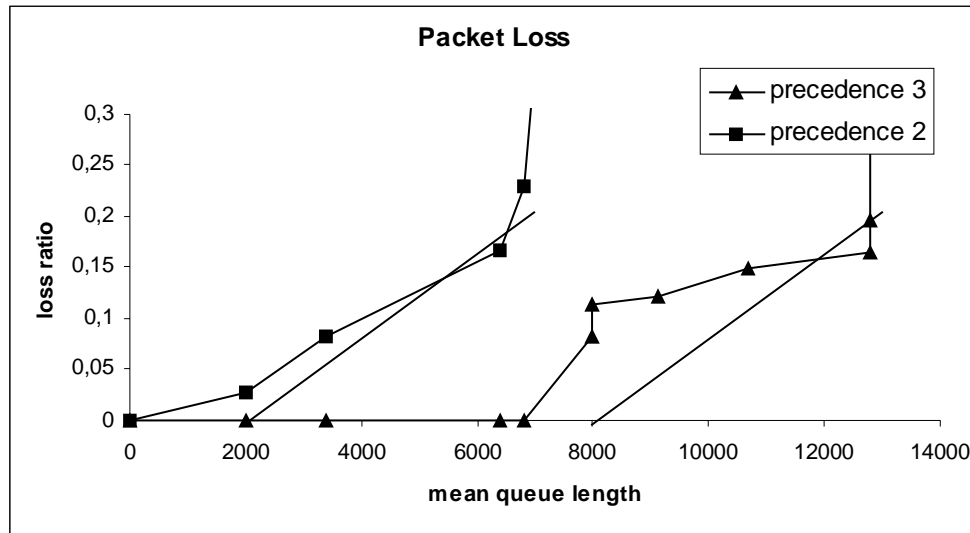


Figure 13. Packet loss ratio with WRED mechanism used

Straight lines on the figure show the theoretical value of packet loss ratio, resulting from the configured thresholds. One can observe, that the measured values are consistent with the theoretical ones, which means that the WRED mechanism works as expected.

4.6.6.1.7 Scheduling mechanisms testing

4.6.6.1.7.1 CBWFQ mechanism without LLQ (priority queue)

The CBWFQ mechanism was configured with two classes (queues) distinguished by precedence bits.

```
aq7507_2#sh policy-map
Policy Map policy4
  class prec2
    bandwidth percent 49
  class prec3
    bandwidth percent 49

aq7507_2#sh queueing int fast 1/0/0
Interface FastEthernet1/0/0 queueing strategy: VIP-based fair queueing
FastEthernet1/0/0 queue size 0
  pkts output 824017, wfq drops 0, nobuffer drops 0
WFQ: aggregate queue limit 25000 max available buffers 25000

Class 0: weight 2 limit 500 qsize 0 pkts output 2038 drops 0
Class 2: weight 49 limit 12250 qsize 0 pkts output 408164 drops 0
Class 3: weight 49 limit 12250 qsize 0 pkts output 413811 drops 0
```

Two traffic stream A and B with different precedence bits were submitted to the 7507 router via Fast Ethernet or Ethernet interfaces. The traffic load of stream A and B was equal and slightly higher than the output port capacity to simulate overload conditions.

Table 4. WFQ weights

Stream A	49	40	30	20	10	5
Stream B	49	59	69	79	89	94

The bandwidth shares of stream A and B on Fast Ethernet interfaces for different WFQ weights were measured and compared with theoretical values. The tested WFQ weights are shown in Table 4. The measurements were done for packet length of 256, 512 and 1024 bytes. Tests smaller values were not possible due to the router performance limitation. The bandwidth shares of test traffic streams (in Mbps) are presented in the Table 5.

Table 5. Transmission rates in WFQ scheduler on Fast Ethernet

Packet length	Flow	WFQ weights (streams A;B)					
		49;49	40;59	30;69	20;79	10;89	5;94
256	Stream A	43.61	29.07	21.81	14.53	14.23	14.21
	Stream B	43.61	57.96	65.42	72.68	72.91	72.99
512	Stream A	44.78	29.85	22.39	14.91	8.14	4.26
	Stream B	44.78	59.70	67.17	74.58	81.42	85.30
1024	Stream A	47.27	31.51	23.63	15.75	8.59	4.50
	Stream B	47.27	63.01	70.89	78.77	85.93	90.03

The Figure 14 presents the bandwidth share of stream A in the output port capacity for different setting of WFQ weights and compares it with theoretical values. In case of Fast Ethernet interface, one may observe the differences between the performance of ideal WFQ scheduler and the implementation available in Cisco router. The differences in rate allocation is represented by the fairness index calculated as follow (see Figure 15):

$$FI = \frac{\left(\sum_i \frac{x_i}{y_i} \right)^2}{n \sum_i \left(\frac{x_i}{y_i} \right)^2}$$

where x_i represent the measured and y_i the theoretical values.

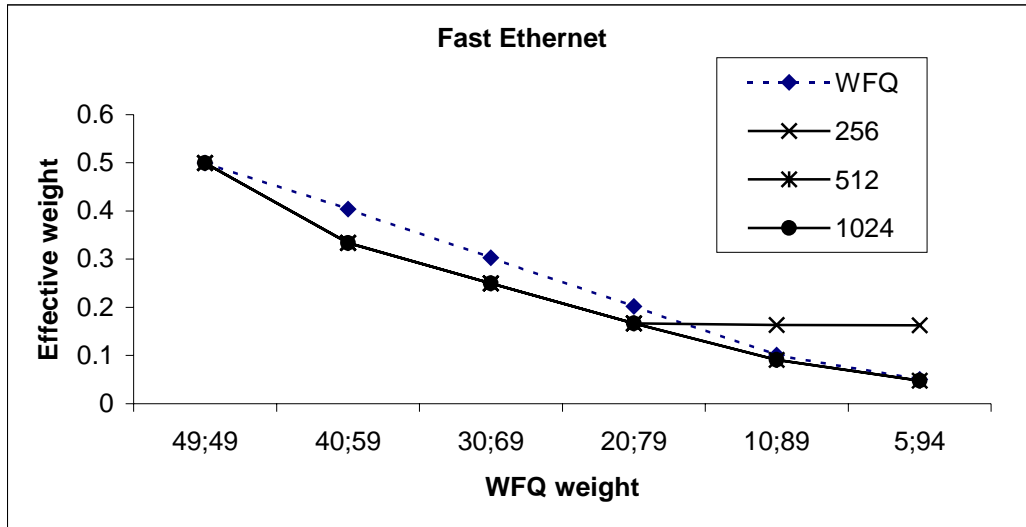


Figure 14. CBWFQ scheduler performance on Fast Ethernet

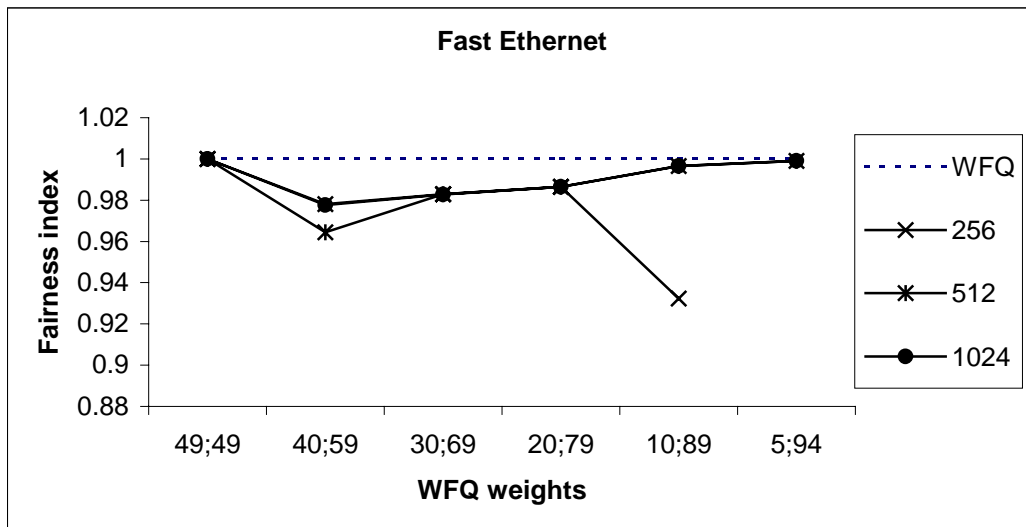


Figure 15. Fairness index of CBWFQ scheduler on Fast Ethernet

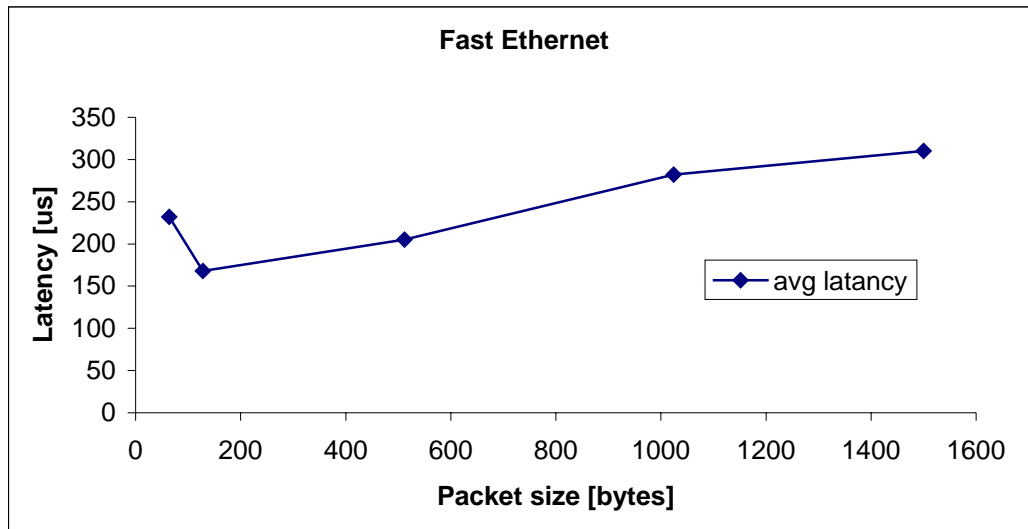


Figure 16. Packet latency in CBWFQ scheduler on Fast Ethernet

The results for Ethernet are shown in Table 6. The tested WFQ weights are shown in Table 4. The measurements were done for packet length from 64 to 1024 bytes.

The Figure 17 presents the bandwidth share of stream A in the output port capacity for different setting of WFQ weights and compares it with theoretical values. The obtained results are similar to those of Fast Ethernet interface. The CBWFQ rates differ from the ideal rates of WFQ scheduler. The differences are shown in the form of fairness index on Figure 18.

Table 6. Transmission rate in WFQ scheduler on Ethernet

Packet length	Flow	WFQ weights (streams A;B)					
		49;49	40;59	30;69	20;79	10;89	5;94
64	Stream A	3.137	2.097	1.568	1.046	0.571	0.298
	Stream B	3.137	4.183	4.706	5.229	5.703	5.975
128	Stream A	3.855	2.569	1.927	1.284	0.700	0.367
	Stream B	3.855	5.140	5.783	6.426	7.009	7.343
256	Stream A	4.354	2.902	2.175	1.450	0.791	0.414
	Stream B	4.354	5.804	6.529	7.254	7.916	8.292
512	Stream A	4.653	3.101	2.327	1.548	0.844	0.442
	Stream B	4.653	6.205	6.980	7.754	8.462	8.864
1024	Stream A	4.817	3.211	2.408	1.606	0.877	0.459
	Stream B	4.817	6.423	7.225	8.028	8.765	9.183

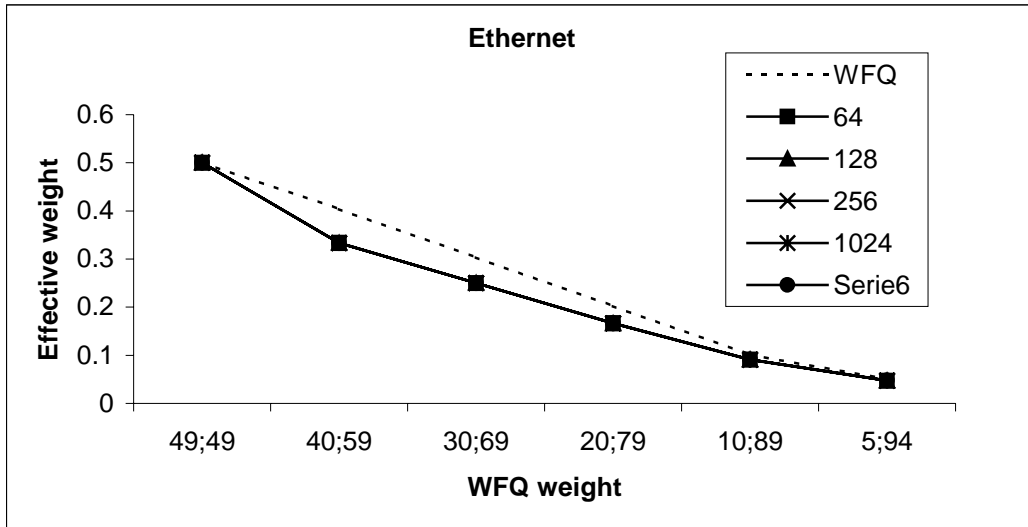


Figure 17. CBWFQ scheduler performance on Ethernet

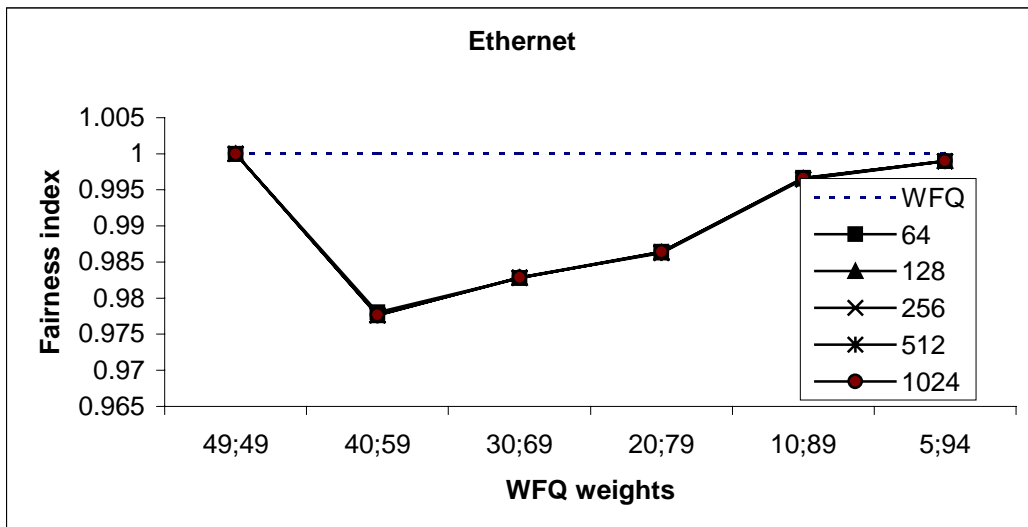


Figure 18. Fairness index of CBWFQ scheduler on Ethernet

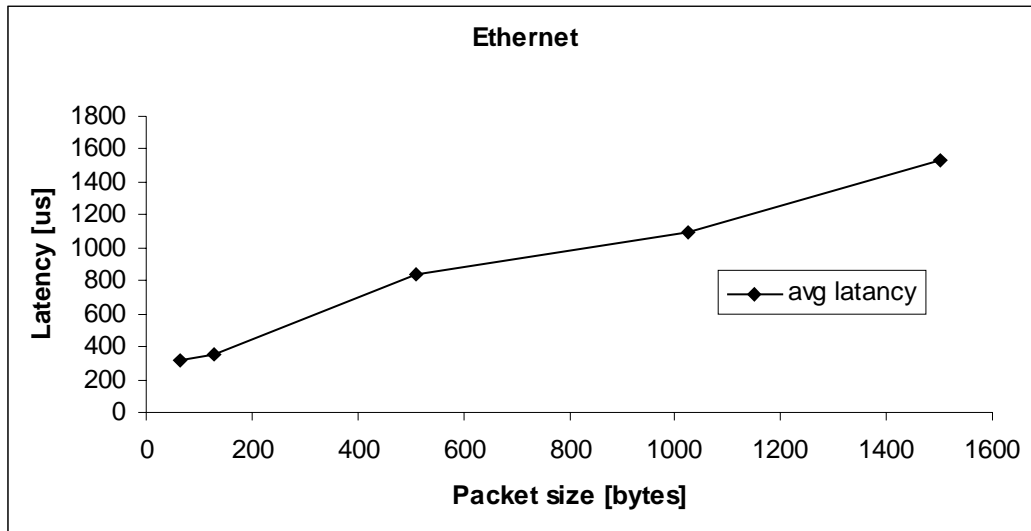


Figure 19. Packet latency in CBWFQ scheduler on Ethernet

4.6.6.1.7.2 CBWFQ with LLQ (priority queue)

The CBWFQ mechanism was configured with two classes, distinguished by precedence bits value. One of the classes was served by strict priority queue. The configuration of the CBWFQ was as follows:

```
aq7507_2#sh policy-map
  Policy Map policy5
    class prec2
      priority 30000

aq7507_2#sh queueing int fast 1/0/0
Interface FastEthernet1/0/0 queueing strategy: VIP-based fair queueing
  FastEthernet1/0/0 queue size 17501
  pkts output 1227072, wfq drops 1487634, nobuffer drops 0
  WFQ: aggregate queue limit 25000 max available buffers 25000

Priority Class: limit 7500 qsize 0 pkts output 342012 drops 370154
  Class 2: bandwidth 30000 exceed drops 370154

Non-Priority Class: limit 17500 qsize 17501 pkts output 884942 drops 373663
  available bandwidth 70000
  Class 0: weight 100 limit 17500 qsize 17499 pkts output 883940 drops 373013
```

Two traffic streams were submitted: Foreground Stream, which was served by the priority queue and Background Stream, which was served as best effort. The rate of foreground stream was 30Mbit/s. Table 7 shows the number of foreground packets dropped during the test and latency introduced by the scheduler to foreground packets as a function of increasing background load.

Table 7. Performance of LLQ scheduler

BG bit rate (Mbit/s)	FG avg latency (us)	FG max latency (us)	FG min latency (us)	FG lost packets
9.80	242	519	174	0

18.70	249	529	172	0
27.61	256	552	174	0
36.55	263	600	175	0
45.38	276	504	177	0
54.30	287	541	177	0
63.19	310	570	196	0
67.88	318	596	211	0
69.48	332	601	235	0
70.48	335	628	237	0
71.52	3284	3452	2919	0
72.22	3283	3445	2935	0
81.00	3281	3465	2916	0
89.94	3280	3446	2865	0
95.85	3280	5878	1071	0

In the measurements shown in the table below the solid line, the best effort queue was congested (FG bit rate + BG bit rate > link capacity). No packet losses were observed in the foreground stream, which means that the priority mechanism works properly.

Latency introduced by the LLQ scheduler to the packets served with priority is depicted on Figure 20. One can observe significant increase of latency (from about 200us to about 3ms) when the sum of foreground and background load exceeds the link capacity.

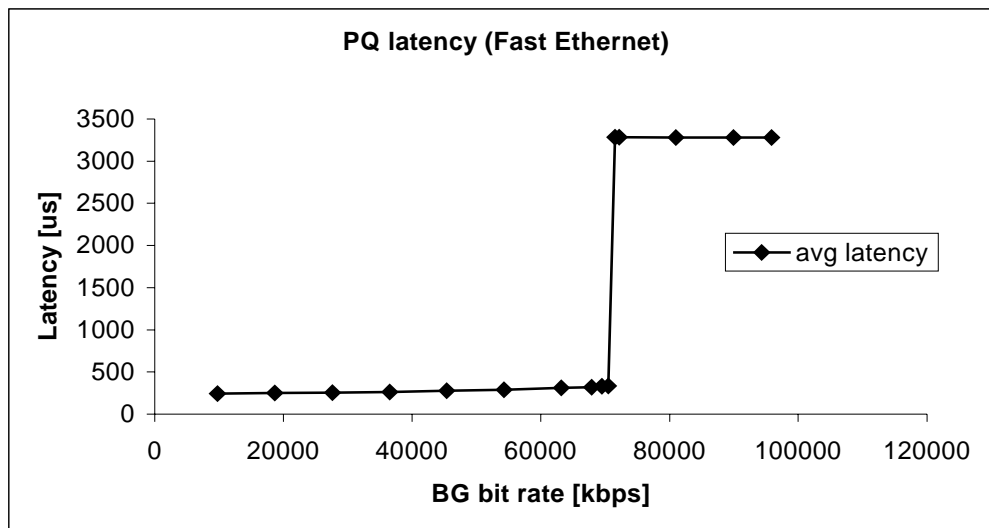


Figure 20. Latency introduced by the LLQ scheduler as a function of background traffic load

Latency introduced to the priority packets as a function of foreground stream rate is depicted on Figure 21 (background stream rate was 95.85 Mb/s). One can observe, that latency is not dependent on the amount of traffic submitted to the priority queue.

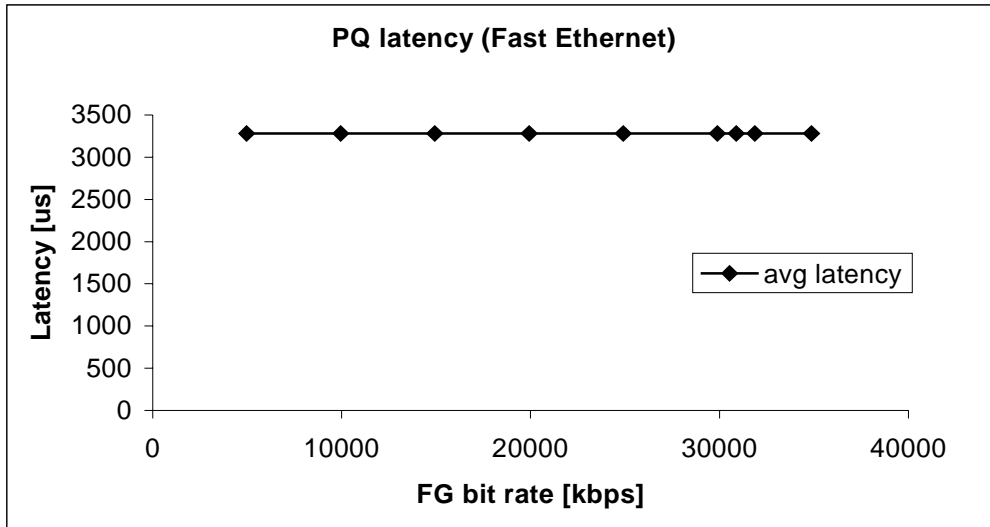


Figure 21. Latency introduced by the LLQ scheduler as a function of foreground traffic load

Figure 22 shows the average latency as a function of length of the packets sent in the background stream. Foreground packet length was constant (1024B). Figure 23 shows the average latency as a function of length of the packets sent in the foreground stream, while the background packet length was constant and equal to 1024B. One can observe, that LLQ latency increases with the size of packets in the foreground and background stream.

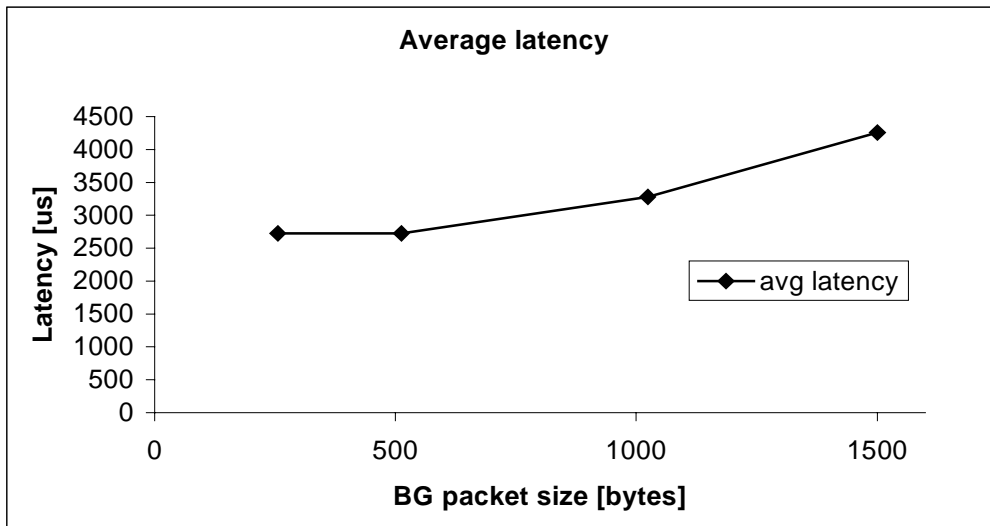


Figure 22. LLQ latency as a function of BG packet length

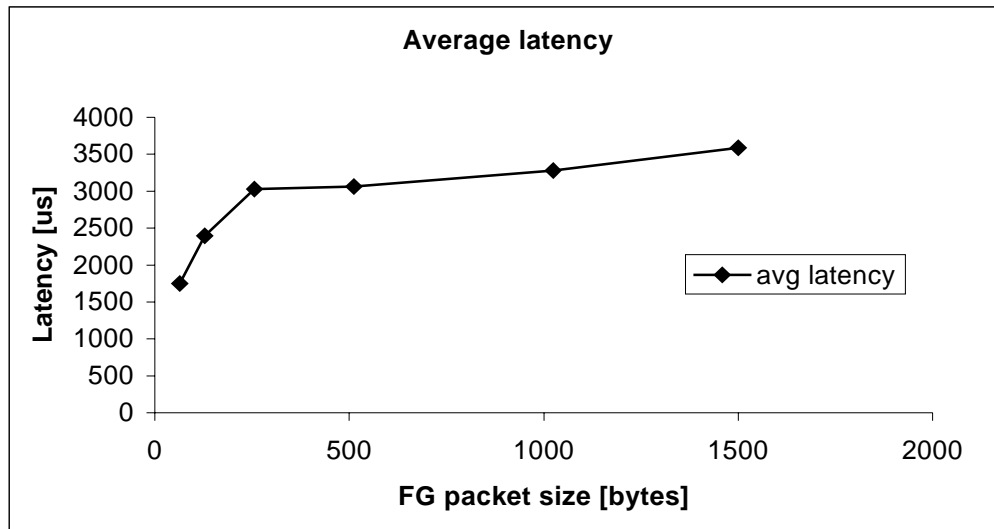


Figure 23. LLQ latency as a function of FG packet length

4.6.6.1.8 Number of flows test

The performance of the router with configured classifiers and policers was tested on Fast Ethernet interfaces of 7507 router. Different switching modes were tested (see section 4.6.6.1.3).

Setting a flow reservation on the router consists in configuring packet classifier and traffic policer for that flow. Policing the flow with a dual token bucket algorithm requires configuring two CAR mechanisms. These mechanisms use up some amount of processor power, so one can expect degradation of packet forwarding performance when flow reservations are configured on the router.

The testing procedure was as follows. Packet classifier (access-list) and a dual token bucket policer (2 CARs) were configured on the router for each flow reservation. For each flow, traffic generator submitted matching packet stream with a rate smaller than configured policed rate. Total rate of traffic submitted to the router was smaller than the rate of the output interface. Then the interface throughput was measured (for packet size 64B). The results of the test should be interpreted as follows: with x flow reservations configured; the router is not able to forward more than y packets/s. Router throughput as a function of number of configured flow reservations is depicted on Figure 24.

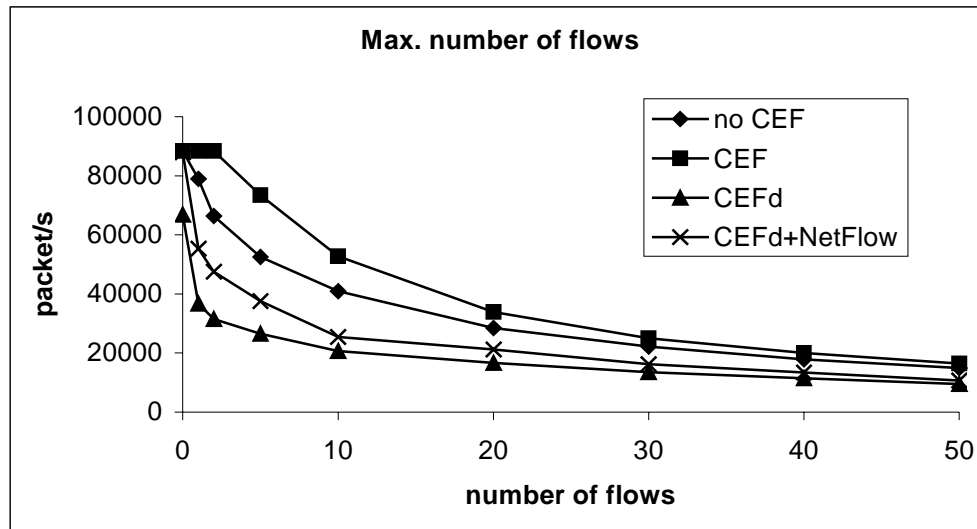


Figure 24. Throughput as a function of number of flow reservations

With 50 flow reservations configured, the router can forward up to 9500 packets/s and with 10 flow reservations, the router can forward up to 25500 packets/s (in CEF distributed mode). Table 8 shows the resulting maximum bit rate on a 100Mbits Fast Ethernet interface, calculated for different packet lengths.

Table 8. Maximum bit rate with 10 and 50 flow reservations

10 flow reservations (25500 packets/s)	
Packet size	Max. bit rate
64 B	13 Mbit/s
128 B	26,1 Mbit/s
512 B	100 Mbit/s (interface rate)
1024 B	100 Mbit/s (interface rate)
1500 B	100 Mbit/s (interface rate)
50 flow reservations (9500 packets/s)	
Packet size	Max. bit rate
64 B	4,8 Mbit/s
128 B	9,7 Mbit/s
512 B	38,9 Mbit/s
1024 B	77,8 Mbit/s
1500 B	100 Mbit/s (interface rate)

4.6.6.2 CISCO 3640 router tests

4.6.6.2.1 Router configuration

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. Processor board ID 21989213 R4700 CPU at 100Mhz Implementation 33, Rev 1.0
Interfaces	4 Ethernet/IEEE 802.3 interface(s) 2 Fast Ethernet/IEEE 802.3 interface(s) 4 Serial network interface(s)

4.6.6.2.2 Test configurations

The configuration used for testing the 3640 router mechanisms and the router performance is depicted on Figure 25.

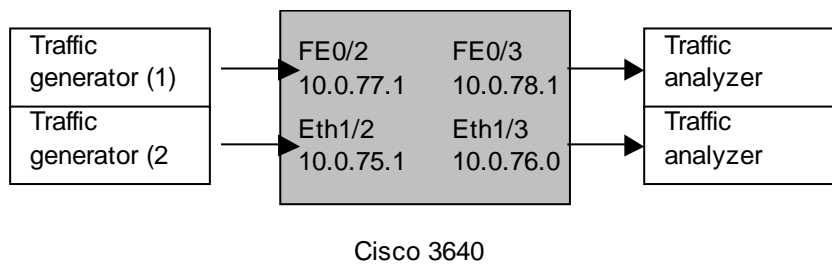


Figure 25. Configuration for 3640 router testing

4.6.6.2.3 Router performance

The router performance tests were done with the IP performance software available on the HP BSTS measurement equipment. The test configuration is depicted on the Figure 25. The generating and receiving ports of the HP BSTS were connected to the router under test (RUT) via FE0/2 and FE0/3 Fast Ethernet interfaces.

4.6.6.2.3.1 Router.performance.throughput test

In the throughput test the maximum lossless transmission rate between input and output port of the Cisco 3640 router as a function of frame size was measured. Two switching mode of the 3640 series router were tested namely the CEF and RSP (no CEF) routing modes. In both modes the 3640 router is able to switch about 50000 packets per second. In case of smaller

packets the router throughput on the Fast Ethernet interfaces is limited by the power of router processor.

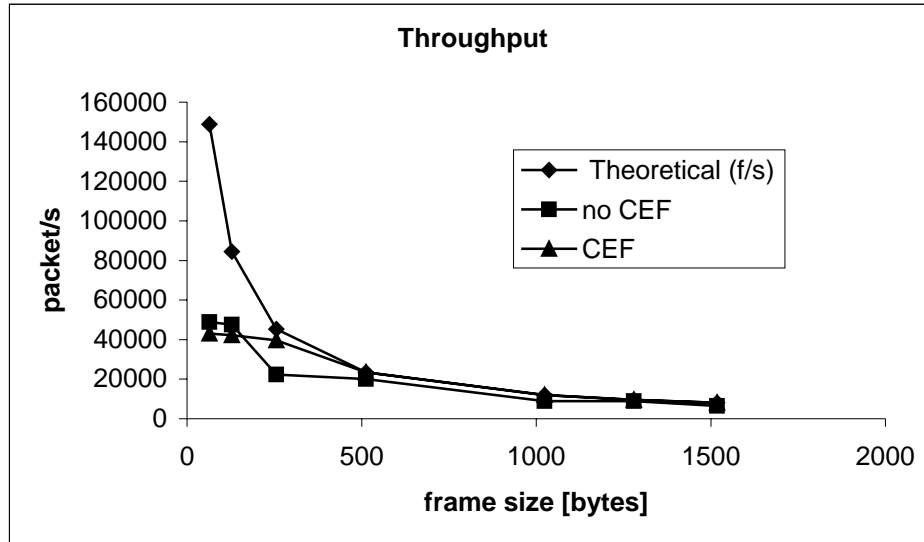


Figure 26. Throughput of the Cisco 3640 router

4.6.6.2.3.2 Router.performance.frameless test

In the frame loss test the frame loss ratio of the Cisco 3640 router as a function of router load was measured. The obtained results explain the router performance degradation observed in the throughput test. The frame loss ratio depends on the packet size. The load threshold above which the packet losses occur, depend on packet size as the router is capable of processing constant number of packets. Similar results were obtained for RSP and CEF switching modes.

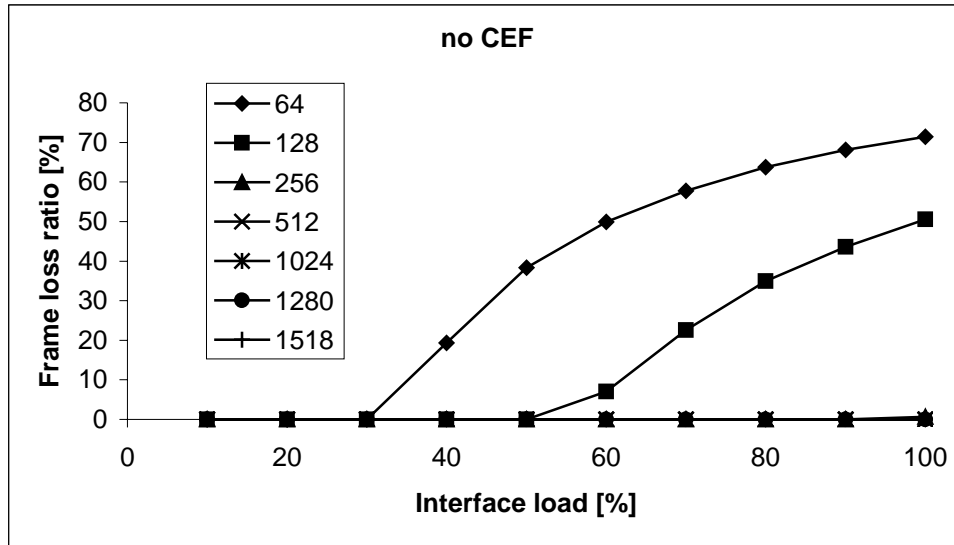


Figure 27. Frame loss of Cisco 3640 router in the RSP (no CEF) mode

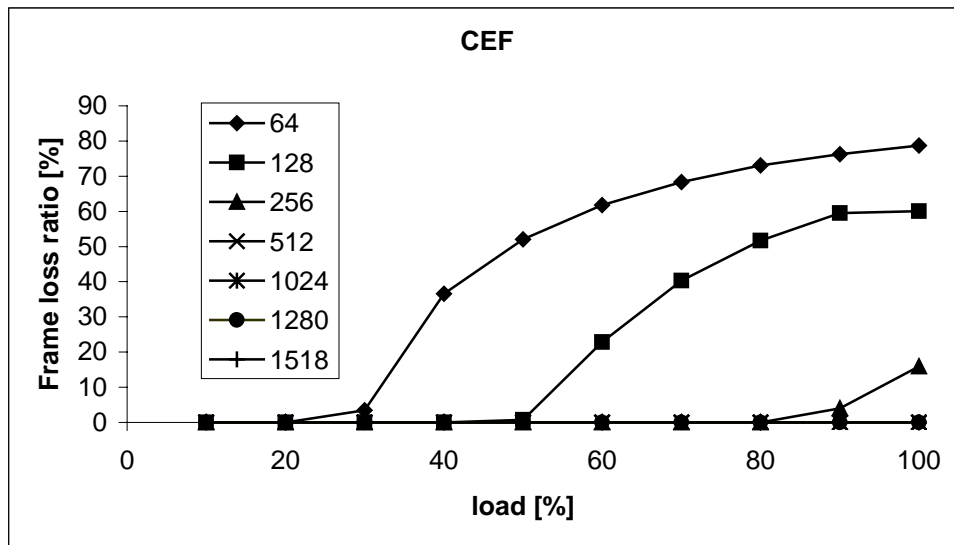


Figure 28. Frame loss of Cisco 3640 router in the CEF mode

4.6.6.2.3.3 Router.performance.latency test

In this test the average frame latency was measured for different router switching modes (RSP and CEF). The measurements were done for different frame lengths on lightly loaded router. The results show the minimum latency introduced by Cisco 3640 series router. The average router 3640 latency is between 100 and 300 microseconds depending on Frame size.

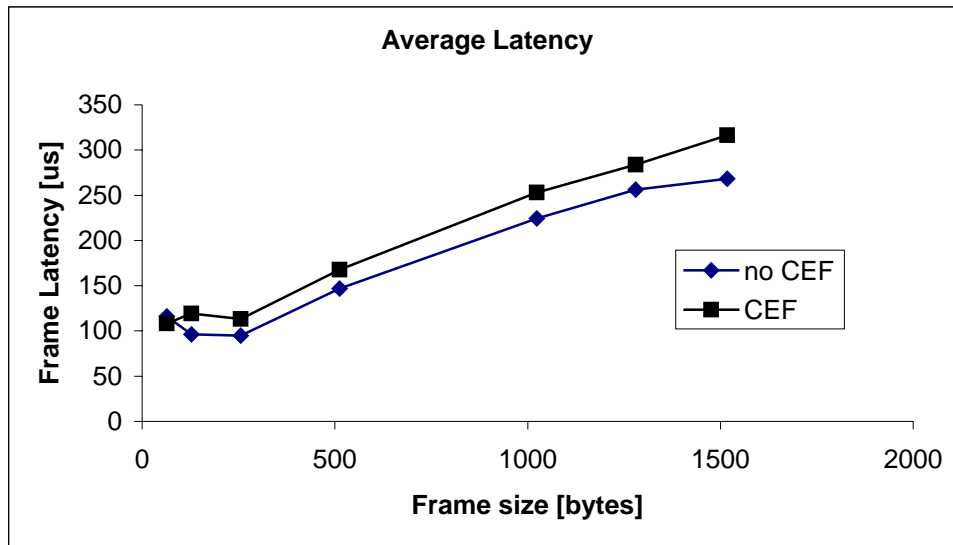


Figure 29. Frame latency of Cisco 3640 router

4.6.6.2.4 Traffic conditioning mechanism testing

In the following test the CAR was configured as single token bucket with dropping. Notice that these settings correspond to TCL1 conditioning function in AQUILA network architecture. The following parameters were set:

- Traffic rate=96 kbps
- Burst size=2000 bytes (no smaller value of bucket size then 2000 bytes is provided by CAR)

4.6.6.2.4.1 Router.mechanisms.conditioning.tcl1.1 test

In this test the traffic conforming to the token bucket parameters was generated. The number of dropped packets, in one-minute observation interval, was recorded. The measurement procedure was repeated for different IP packet lengths i.e. 64, 128, 256, 512, 1024 and 1500 bytes.

The test results are included in Table 9. In case of conforming traffic the CAR mechanism works correctly (no packet drop was observed).

4.6.6.2.4.2 Router.mechanism.conditioning.tcl1.2 test

In this test non-conforming traffic was generated. The test traffic exceeds the CAR transmission rate of 96 kbps. The measurement procedure was repeated for different traffic rates and IP packet lengths. The number of dropped packets, in one-minute observation interval, was recorded.

The test results are presented in Table 1 and Table 2 in the form of the number of dropped packet in one minute measurement interval and packet dropping rate. The CAR mechanism works correctly. As can be seen from Figure 10 the number of dropped packet is proportional to the difference between the declared and actual transmission rate. Moreover, the dropping rate doesn't depend on the packet length (only on the transmission rate).

Table 9. Number of dropped packets in one-minute measurement interval

MAC rate	IP rate	64	256	512	1024	1500
96	78.8	0	0	0	0	0
96.1	78.9	10	3	1	1	1
96.5	79.2	48	14	7	4	3
97	79.6	96	28	14	7	5
98	80.4	192	55	29	14	9
99	81.2	280	79	43	21	15
100	82.1	385	111	66	29	20
105	86.2	866	250	128	65	45
110	90.3	1346	389	183	101	70

Table 10. Packet dropping rate

MAC rate	IP rate	64	256	512	1024	1500
96	78.8	0	0	0	0	0
96.1	78.9	0.00108	0.00112	0.00073	0.00144	0.00210
96.5	79.2	0.00517	0.00522	0.00509	0.00574	0.00628
97	79.6	0.01029	0.01039	0.01012	0.00999	0.01041
98	80.4	0.02038	0.02020	0.02075	0.01977	0.01854
99	81.2	0.02941	0.02873	0.03046	0.02936	0.03059
100	82.1	0.04004	0.03996	0.04629	0.04014	0.04037
105	86.2	0.08578	0.08571	0.08550	0.08568	0.08651
110	90.3	0.12726	0.12731	0.11668	0.12708	0.12846

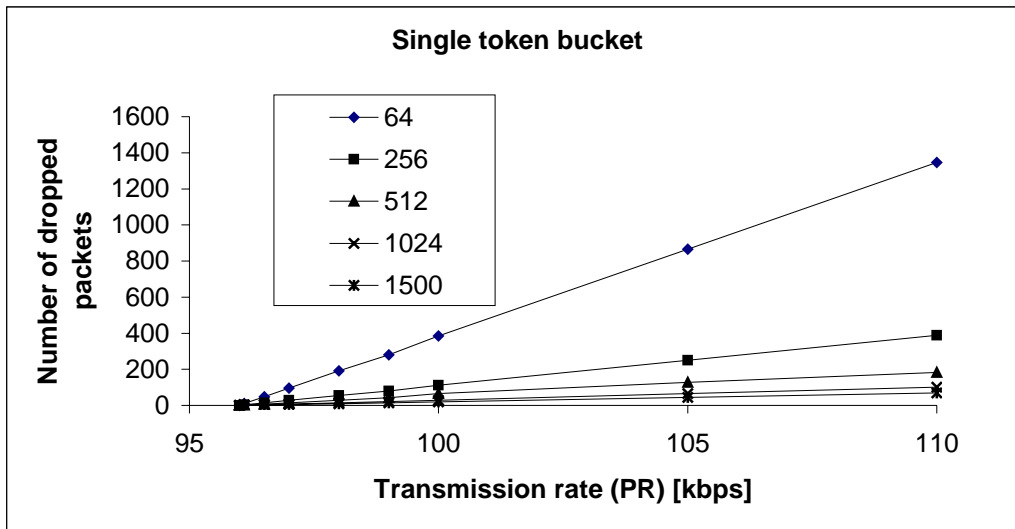


Figure 30. Number of dropped packets

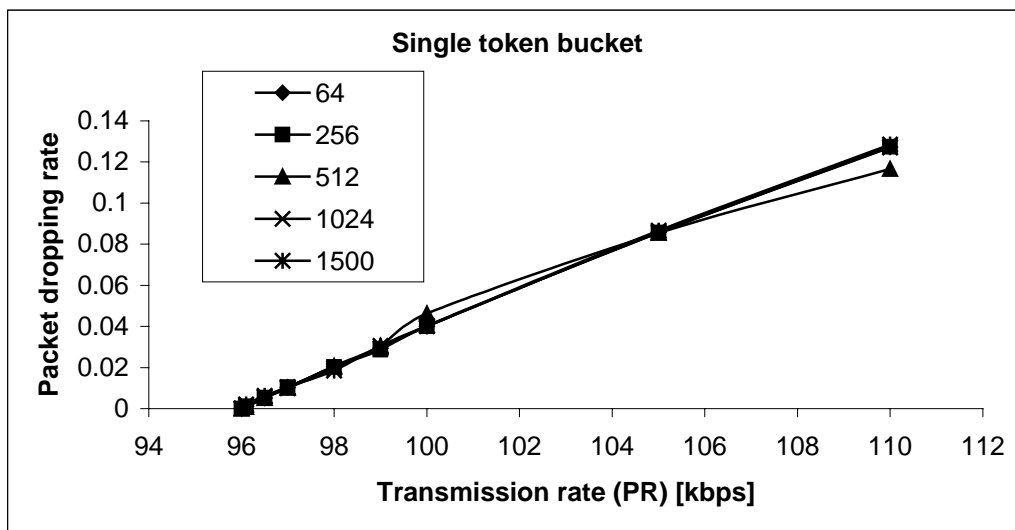


Figure 31. Packet dropping rate

4.6.6.2.4.3 Router.mechanisms.conditioning.tcl1.3 test

In this test the performance of the packet burst size policing mechanism (CAR) of the Cisco 3640 router was tested. The on-off test traffic with burst length (50 packets) exceeding the token bucket size was submitted to the router. During On period packets were transmitted with the full interface speed. The off period was long enough to allow for full bucket “regeneration”. The committed burst size was calculated from the measured number of transmitted and dropped packets. The measurements were repeated for different frame sizes.

The measured results were compared with the theoretical conforming burst sizes of the single token bucket mechanism with the given parameters. As can be seen from Figure 32 the CAR mechanism allows for longer bursts than standard token bucket algorithm (similarly as for 7507).

Table 11. Conforming burst sizes

Packet length (Bytes)		2000-1001	1000-667	666-501	500-401	400-334	333-286	285-251	250-223	222-201
Conforming burst size (Packets)		2	3	4	5	6	7	8	9	10
Packet length (Bytes)	200-182	181-167	166-154	153-143	142-134	133-126	125-118	117-112	111-106	105-101
Conforming burst size (Packets)	11	12	13	14	15	16	17	18	19	20

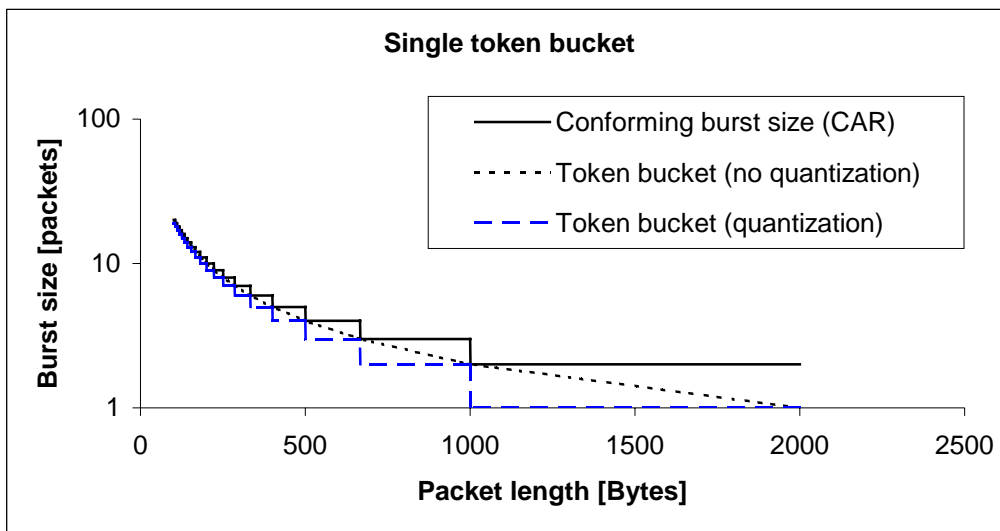


Figure 32. Conforming burst sizes

Remark: Exactly the same results were obtained for CAR mechanism configured as single token bucket with marking (conditioning mechanism for class TCL3). The dropping rate represents the packets marking rate for class TCL3 conditioning mechanism. The conditioning mechanisms for classes TCL2 and TCL4 were not tested separately as they are implemented by two CAR mechanisms.

4.6.6.2.5 Scheduling mechanisms testing

4.6.6.2.5.1 CBWFQ mechanism without LLQ (priority queue)

The CBWFQ mechanism was configured with two classes (queues) distinguished by precedence bits.

Two traffic stream A and B with different precedence bits were submitted to the 3640 router via Ethernet interfaces. The traffic load of stream A and B was equal and slightly higher than the output port capacity to simulate overload conditions.

Table 12. WFQ weights

Stream A	50	40	30	20	10	5
Stream B	50	60	70	80	90	95

The bandwidth shares of stream A and B on Ethernet interfaces for different WFQ weights were measured and compared with theoretical values. The tested WFQ weights are shown in Table 12. The measurements were done for packet length of 256, 512 and 1024 bytes. Tests smaller values were not possible due to the router performance limitation. The bandwidth shares of test traffic streams (in Mbps) are presented in the Table 13.

Table 13. Transmission rates in WFQ scheduler on Ethernet

Packet length	Flow	WFQ weights (streams A;B)					
		50;50	40;60	30;70	20;80	10;90	5;95
256	Stream A	4,352	3,480	2,587	1,776	0,905	0,451
	Stream B	4,352	5,226	6,119	6,930	7,801	8,255
512	Stream A	4,653	3,719	2,765	1,896	0,967	0,479
	Stream B	4,653	5,583	6,541	7,815	8,339	8,827
1024	Stream A	4,817	3,850	2,859	1,966	0,500	0,250
	Stream B	4,817	5,784	6,775	7,668	8,634	4,571

The Figure 33 presents the bandwidth share of stream A in the output port capacity for different setting of WFQ weights and compares it with theoretical values. In case of Fast Ethernet interface, one may observe the differences between the performance of ideal WFQ scheduler and the implementation available in Cisco router. The differences in rate allocation is represented by the fairness index calculated as follow (see Figure 34):

$$FI = \frac{\left(\sum_i \frac{x_i}{y_i} \right)^2}{n \sum_i \left(\frac{x_i}{y_i} \right)^2}$$

where x_i represent the measured and y_i the theoretical values.

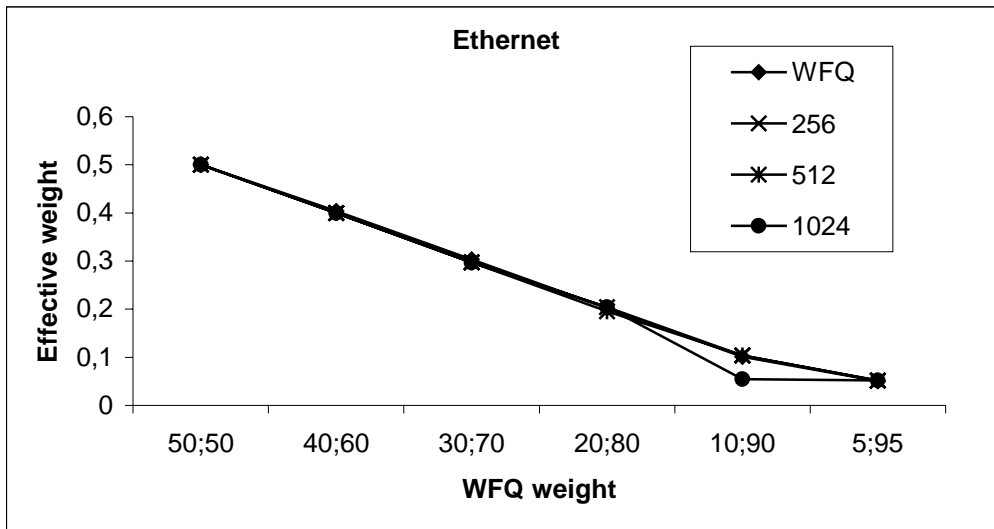


Figure 33. CBWFQ scheduler performance on Ethernet

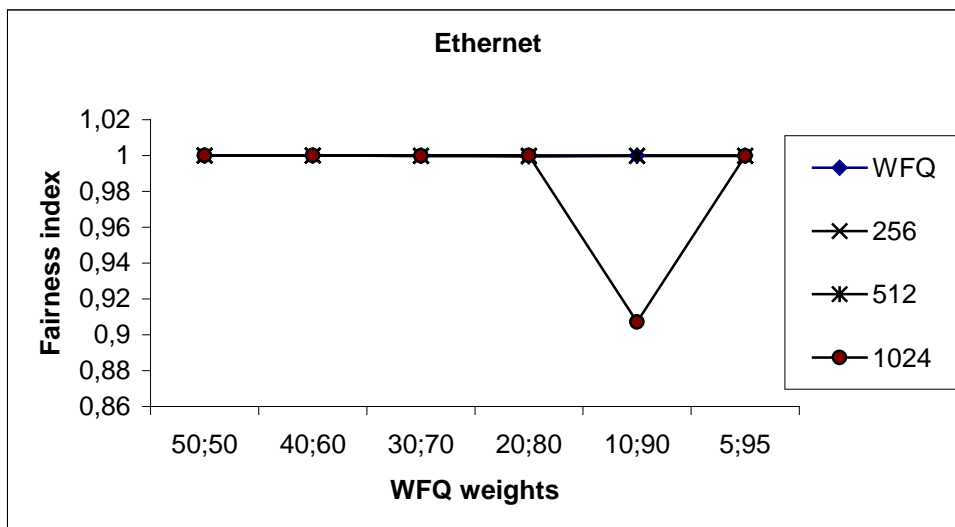


Figure 34. Fairness index of CBWFQ scheduler on Fast Ethernet

Due to the router performance limitations, WFQ scheduler was not tested on the Fast Ethernet interface.

4.6.6.2.6 Number of flows test

The performance of the router with configured classifiers and policers was tested on Fast Ethernet interfaces of 3640 router. Different switching modes were tested (RSP and CEF).

The testing procedure was as follows. Packet classifier (access-list) and a dual token bucket policer (2 CARs) were configured on the router for each flow reservation. For each flow, traffic generator submitted matching packet stream with a rate smaller than configured policed rate. Total rate of traffic submitted to the router was smaller than the rate of the output interface. Then the interface throughput was measured (for packet size 64B). The results of the test should be interpreted as follows: with x flow reservations configured; the router is not able to forward more than y packets/s. Router throughput as a function of number of configured flow reservations is depicted on Figure 35.

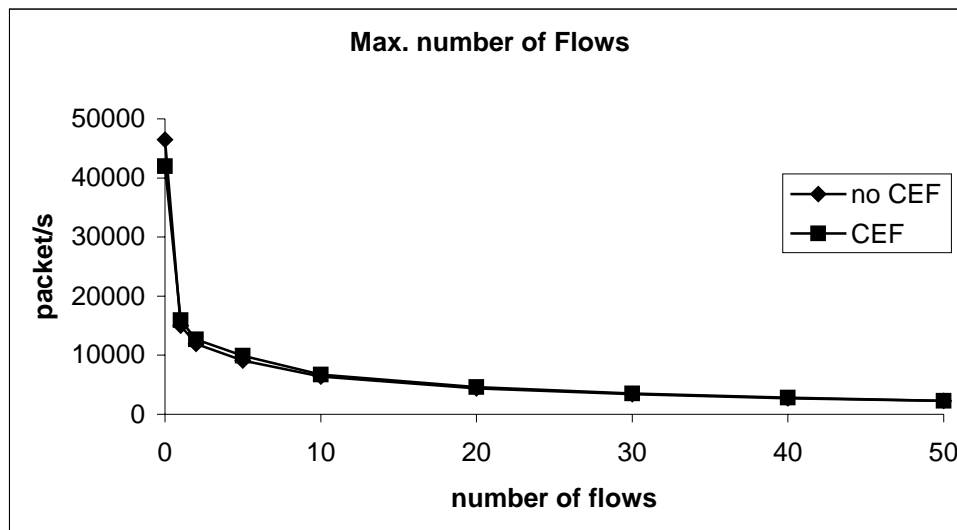


Figure 35. Throughput as a function of number of flow reservations

With 50 flow reservations configured, the router can forward up to 2300 packets/s and with 10 flow reservations, the router can forward up to 6700 packets/s (in CEF mode). Table 14 shows the resulting maximum bit rate on a 100Mbps Fast Ethernet interface, calculated for different packet lengths.

Table 14. Maximum bit rate with 10 and 50 flow reservations

10 flow reservations (6700 packets/s)	
Packet size	Max. Bit rate
64 B	3,4 Mbit/s
128 B	6,8 Mbit/s
512 B	27,4 Mbit/s
1024 B	54,8 Mbit/s
1500 B	80,4 Mbit/s
50 flow reservations (2300 packets/s)	
Packet size	Max. Bit rate
64 B	1,2 Mbit/s
128 B	2,3 Mbit/s
512 B	9,4 Mbit/s

1024 B	18,8 Mbit/s
1500 B	27,6 Mbit/s

4.6.6.3 CISCO 1605 router tests

4.6.6.3.1 Router configuration

IOS software release	Cisco Internetwork Operating System Software IOS (tm) 1600 Software (C1600-SY-M) Version 12.1(3)T, RELEASE SOFTWARE (fc1) c1600-sy-mz.121-3.T.bin
Router processor	Cisco 1605 (68360) processor (revision C) with 12288K/4096K bytes of memory. Processor board ID 17053319, with hardware revision 00000002
Interfaces	2 Ethernet/IEEE 802.3 interface(s) 1 Serial (sync/async) network interface(s)

4.6.6.3.2 Test configurations

The configuration used for testing the 1605 router mechanisms and the router performance is depicted on Figure 36.

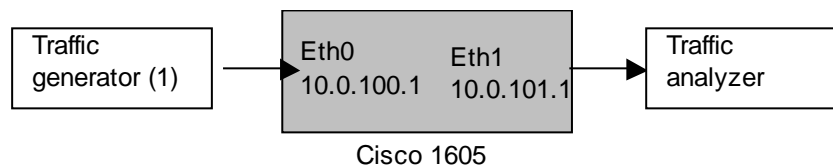


Figure 36. Configuration for router testing

4.6.6.3.3 Router performance

The router performance tests were done with the IP performance software available on the HP BSTS measurement equipment. The test configuration is depicted on the Figure 36. The generating and receiving ports of the HP BSTS were connected to the router under test (RUT) via Eth0 and Eth1 Ethernet interfaces.

4.6.6.3.3.1 Router performance throughput test

In the throughput test the maximum lossless transmission rate between input and output port of the Cisco 1605 router as a function of frame size was measured. Two switching mode of the 1605 series router were tested namely the CEF and RSP (no CEF) routing modes. In both modes the 1605 router is able to switch about 4000 packets per second. This value limits the performance of the router (which can be especially seen for small packets).

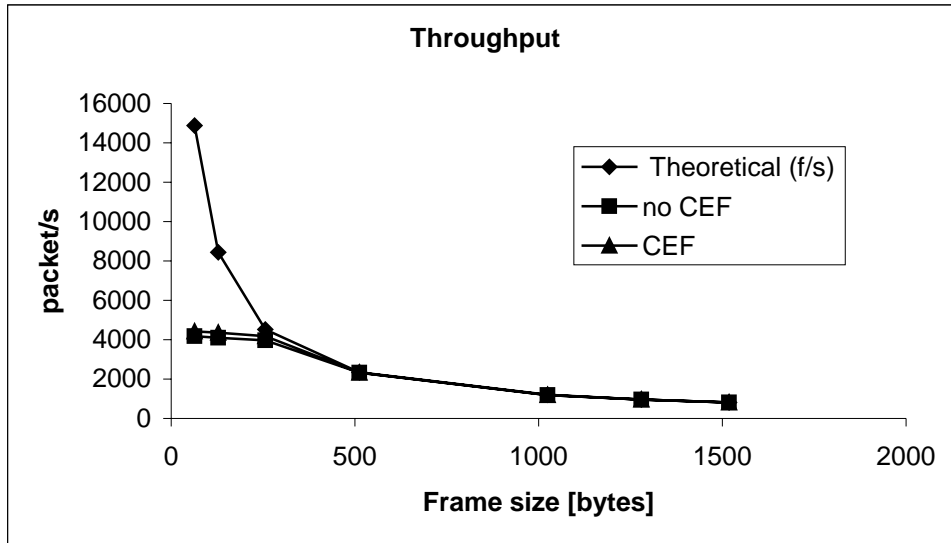


Figure 37. Throughput as a function of number of flow reservations

4.6.6.3.3.2 router.performance.frame loss test

In the frame loss test the frame loss ratio of the Cisco 1605 router as a function of router load was measured. The test results are generally similar to those obtained for 7507 and 3640 routers. When the offered load excess the number of packets the router can process the packet loss occurs. Similar results were obtained for RSP and CEF switching modes.

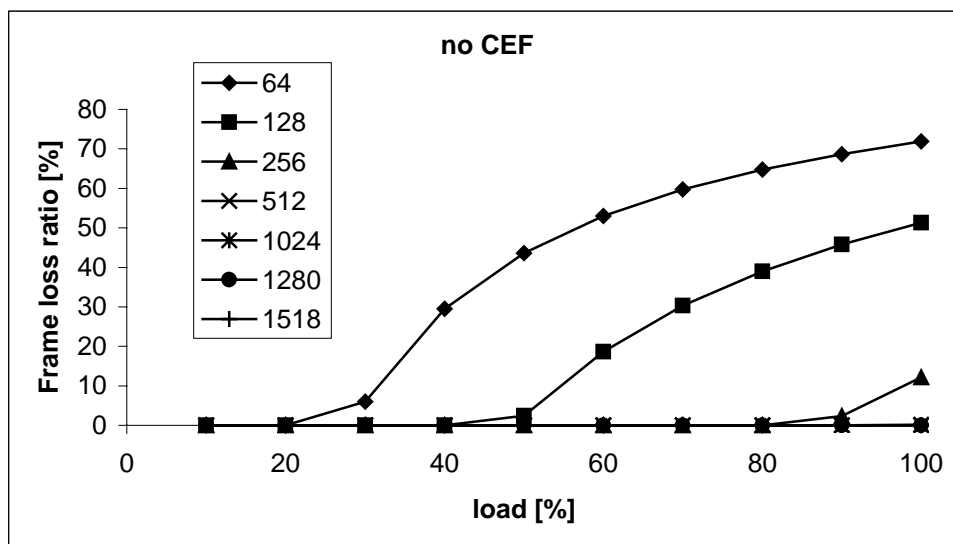


Figure 38. Frame loss of Cisco 1605 router in the RSP (no CEF) mode

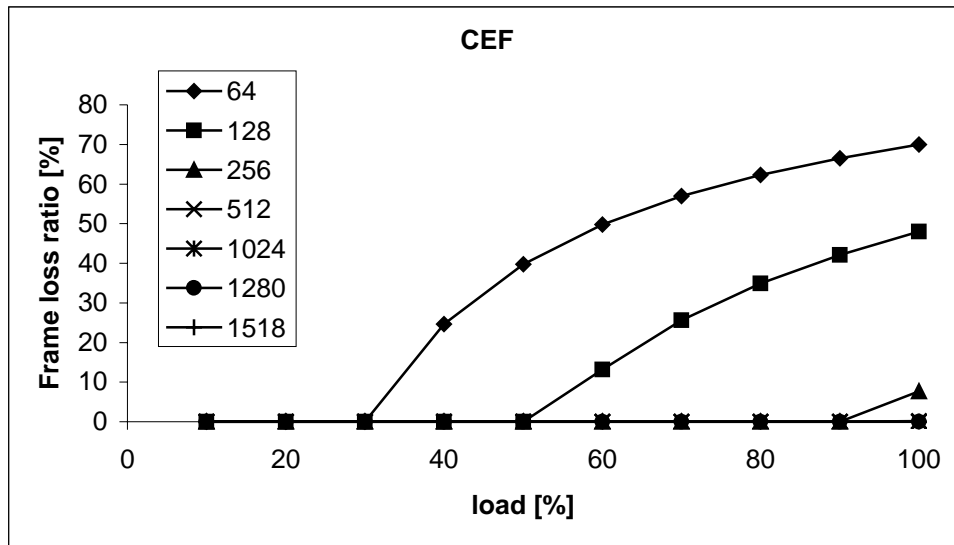


Figure 39. Frame loss of Cisco 1605 router in the CEF mode

4.6.6.3.3 Router.performance.latency test

In this test the average frame latency was measured for different router switching modes (RSP and CEF). The measurements were done for different frame lengths on lightly loaded router. The results show the minimum latency introduced by Cisco 1605 series router. The average 1605 router latency is between 04 and 1.2 milliseconds depending on Frame size.

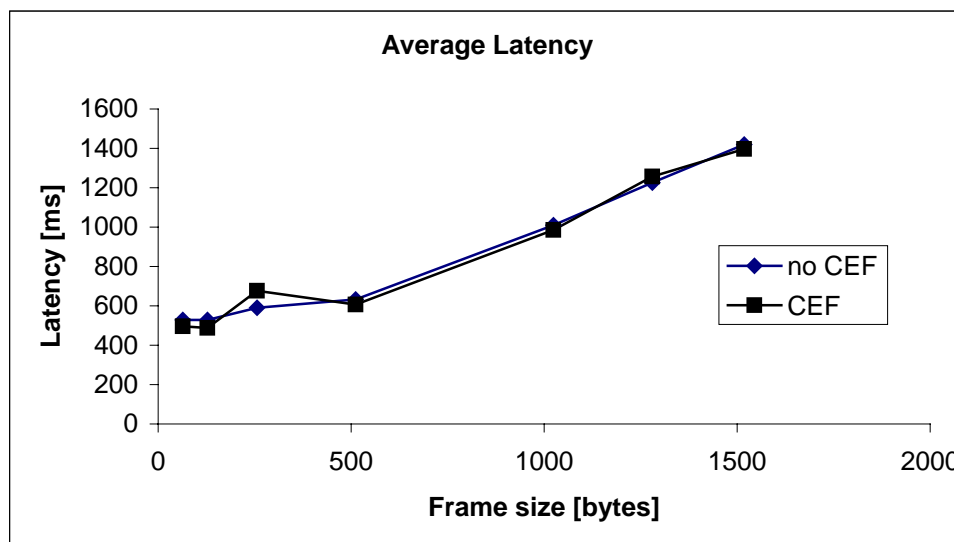


Figure 40. Frame latency of Cisco 1605 router

4.6.6.3.4 Number of flows test

The performance of the router with configured classifiers and policers was tested on Ethernet interfaces of 1605 router. Different switching modes were tested (RSP and CEF).

The testing procedure was as follows. Packet classifier (access-list) and a dual token bucket policer (2 CARs) were configured on the router for each flow reservation. For each flow, traffic generator submitted matching packet stream with a rate smaller than configured policed rate. Total rate of traffic submitted to the router was smaller than the rate of the output interface. Then the interface throughput was measured (for packet size 64B). The results of the test should be interpreted as follows: with x flow reservations configured; the router is not able to forward more than y packets/s. Router throughput as a function of number of configured flow reservations is depicted on Figure 41.

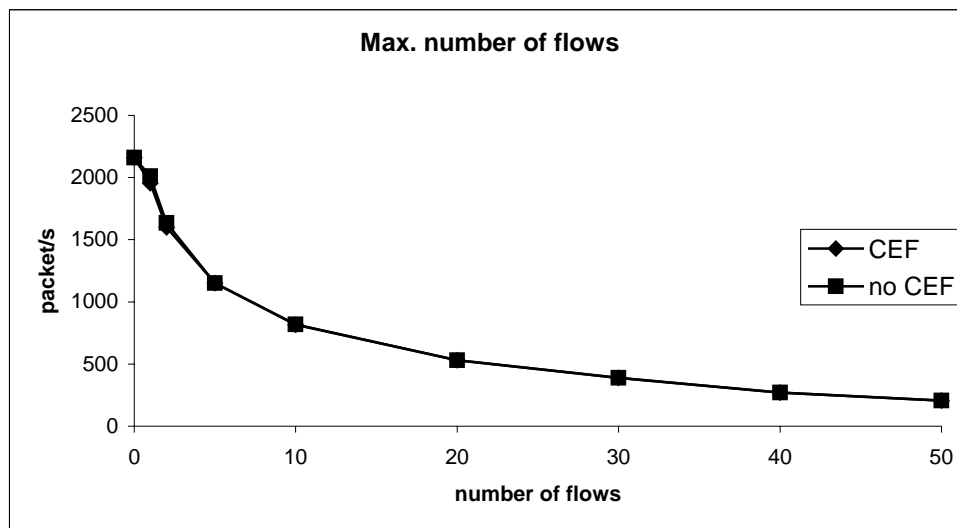


Figure 41. Throughput as a function of number of flow reservations

With 50 flow reservations configured, the router can forward up to 815 packets/s and with 10 flow reservations, the router can forward up to 207 packets/s (in CEF mode). Table 15 shows the resulting maximum bit rate on a 10Mbits Ethernet interface, calculated for different packet lengths.

Table 15. Maximum bit rate with 10 and 50 flow reservations

10 flow reservations (815 packets/s)	
Packet size	Max. Bit rate
64 B	417 kbit/s
128 B	834 kbit/s
512 B	3,3 Mbit/s
1024 B	6,6 Mbit/s
1500 B	9,7 Mbit/s
50 flow reservations (207 packets/s)	

Packet size	Max. Bit rate
64 B	105 kbit/s
128 B	211 kbit/s
512 B	842 kbit/s
1024 B	1,7 Mbit/s
1500 B	2,5 Mbit/s

4.6.6.4 CISCO 2600 Router Tests

4.6.6.4.1 Router configuration

IOS software release	Cisco Internetwork Operating System Software IOS (tm) 1600 Software (C1600-SY-M) Version 12.1(5)T
Router processor	MPC860 (revision 0x101) with 45056k/4096k bytes of memory Processor
Interfaces	2 Fast Ethernet/IEEE 802.3 interfaces 2 Serial (sync/async) network interfaces 2 Voice FXS interfaces

4.6.6.4.2 Test configurations

The configuration used for testing the router mechanisms and the router performance is shown in figure 2-1.

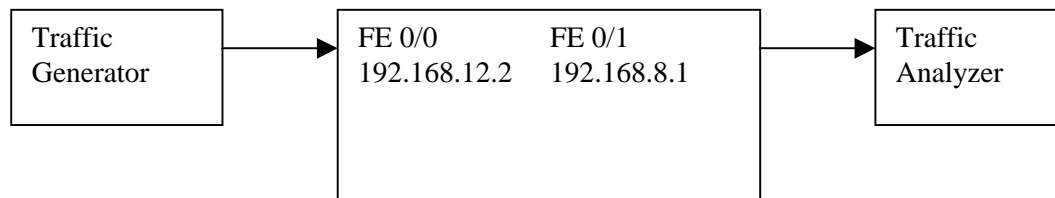


Figure 42. Configuration for 2620 router tests

4.6.6.4.3 Router performance testing

Router performance tests were done with the SmartApplications version 2.32 (NetCom Systems). The test configuration is shown in Figure 42.

4.6.6.4.3.1 Router.performance.throughput test

In the throughput test the maximum lossless rate between input and output port of the Cisco 2620 router was measured as a function of frame size.

The results are illustrated in the following two figures. Figure 43 shows the throughput in packets per second as a function of frame size.

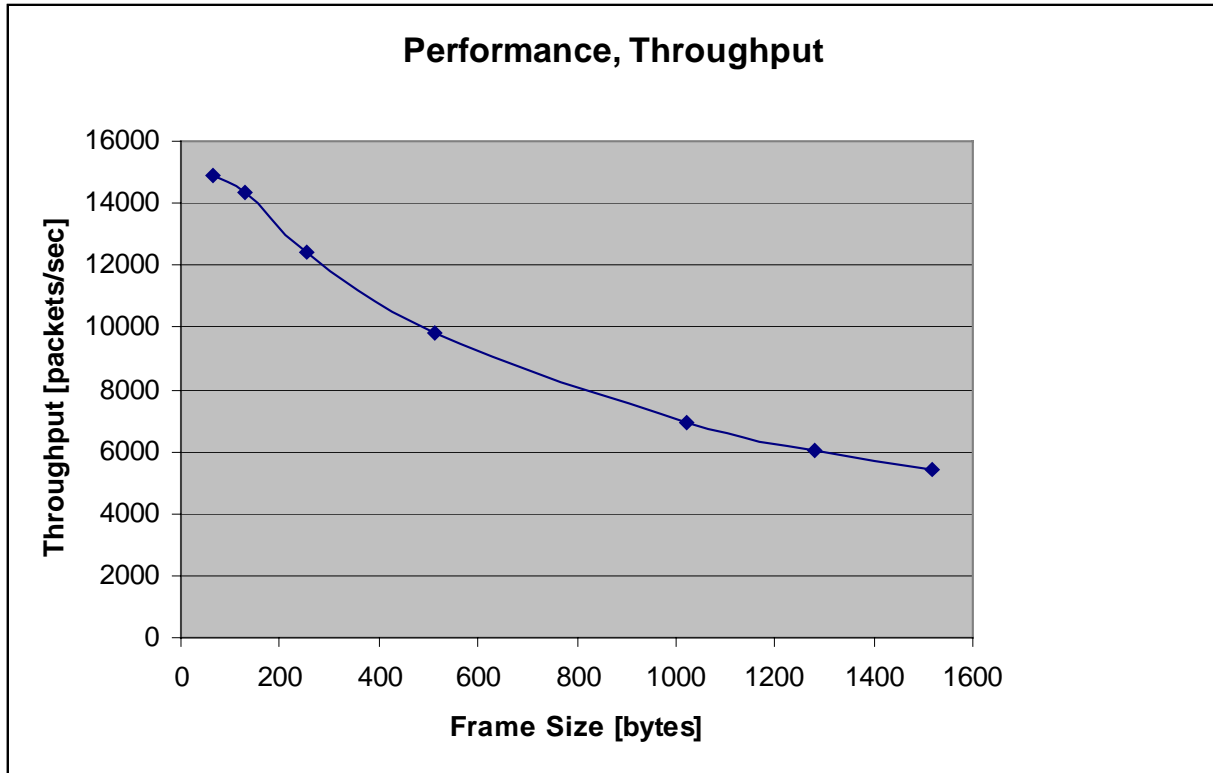


Figure 43. Throughput [packets] as a function of frame size

Figure 44 shows the throughput in bytes per second as a function of frame size.

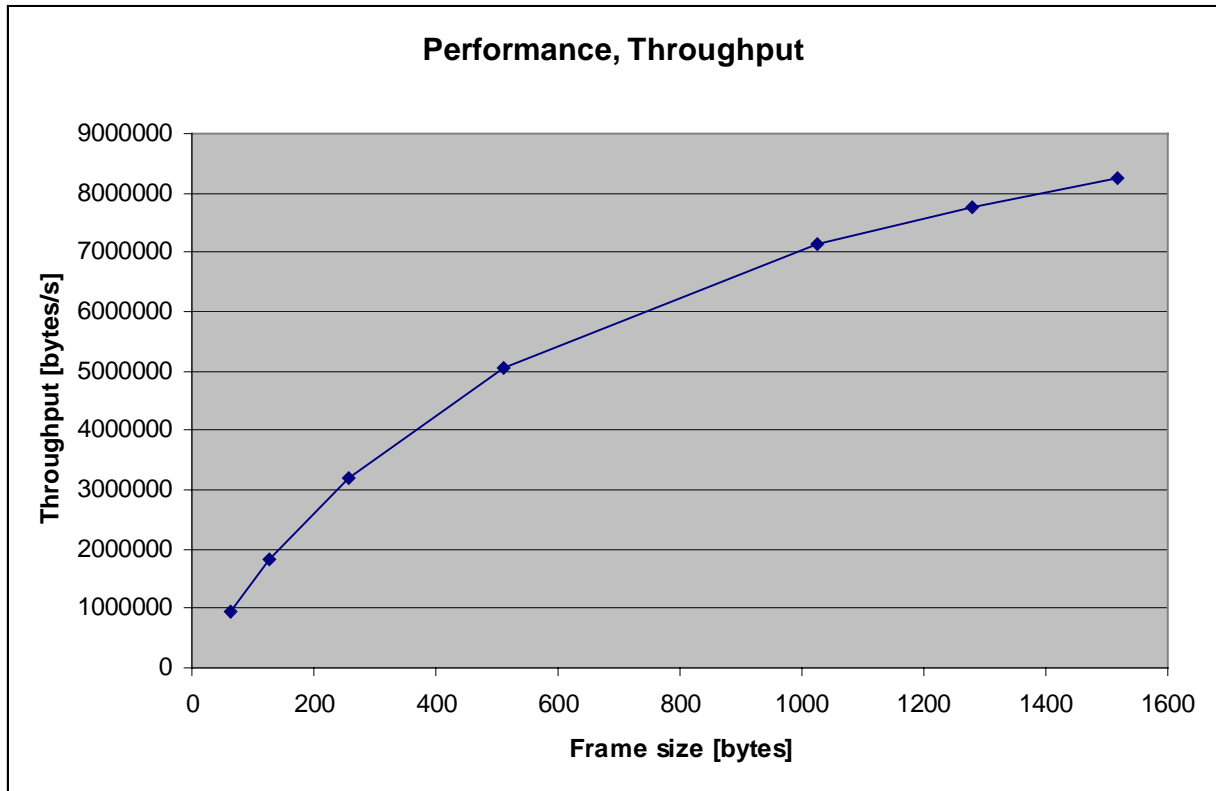


Figure 44. Throughput [bytes] as a function of frame size

4.6.6.4.3.2 Router.performance.frameless test

In the frame loss test the frame loss ratio of the Cisco 2620 router was measured as a function of router load. The results of frame loss tests can be seen in Figure 45 as a function of interface load.

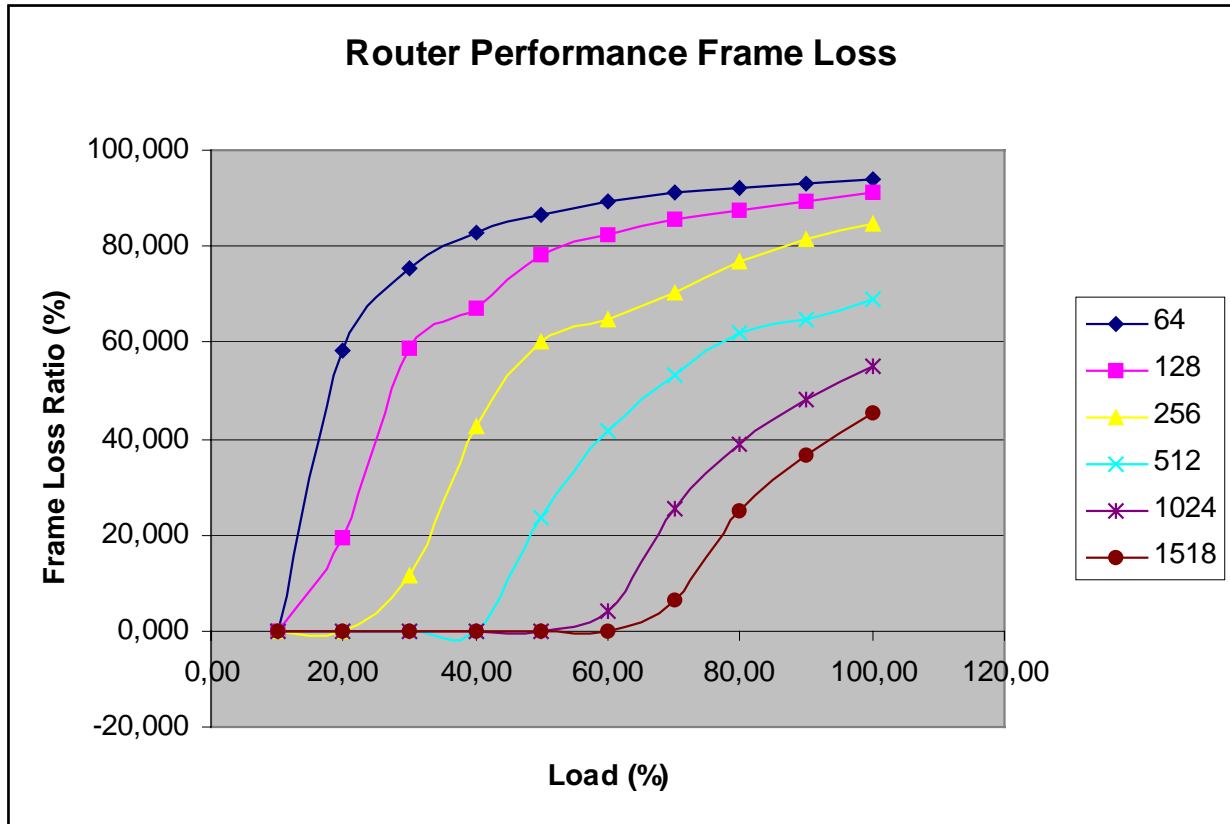


Figure 45. Router performance frame loss

4.6.6.4.3.3 Router.performance.latency test

Latency tests were done for different frame lengths on lightly loaded router. The results are presented in Figure 46.

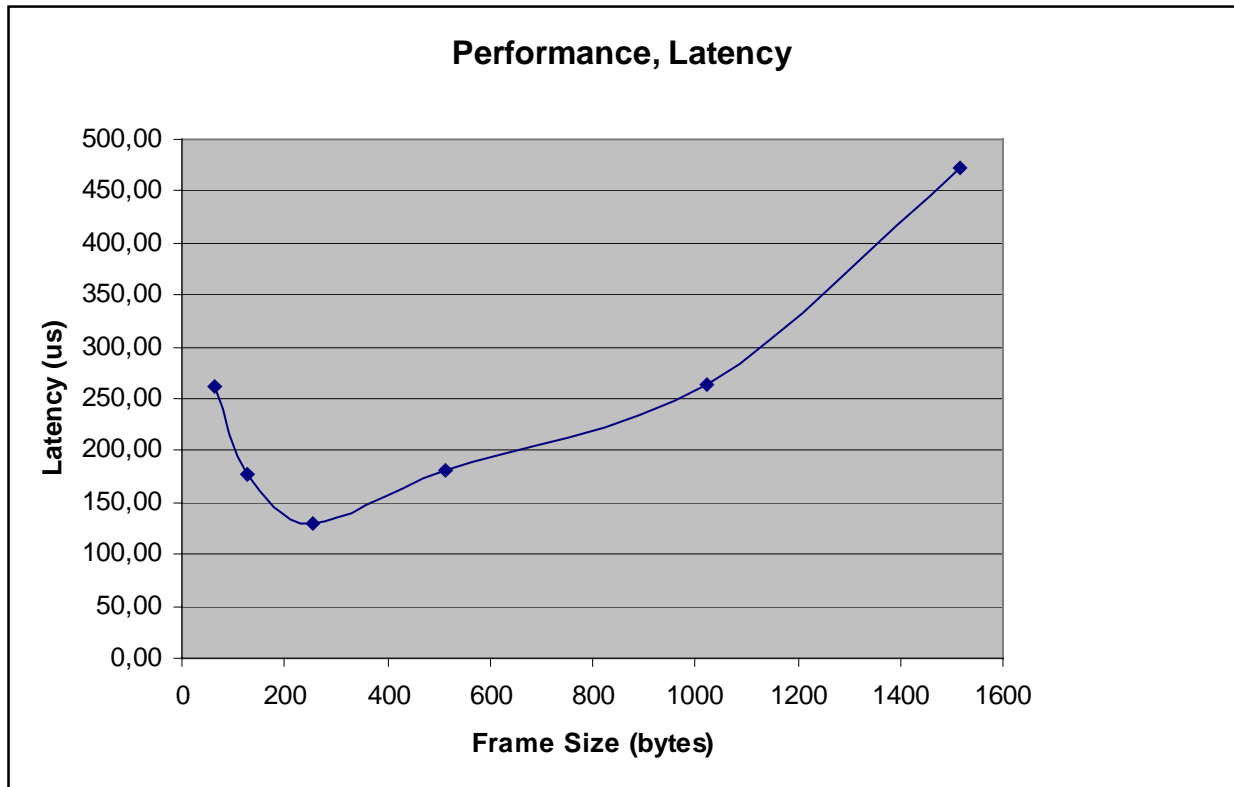


Figure 46. Router performance latency

4.6.6.4.4 Traffic classifier testing

The traffic classifier tests were done with the SmartFlow version 1.20.018 (Spirent Communications).

The traffic classifier tests were performed in configurations outlined in Figure 42 Both the input and the output ports were Fast Ethernet and their IP addresses were 192.168.12.2 and 192.168.8.1 respectively. Two traffic streams were sent to the router. The test duration was 10 seconds and each stream generated 4222 packets of size 128B.

- Stream1: 430kb/s, packets did not match the criteria specified in the access list.
- Stream2: 430kb/s, packets matched the criteria specified in the access list.

Test router.mechanisms.classifier 1

```
Show access-list counters:
Extended IP access list 110
Permit ip any any precedence flash (4222 matches)
```

The result of “show access-list”-command, executed after the test, shows that the number of packets matching to the precedence bits was equal to the number of packets generated in

stream 2.

Test router.mechanisms.classifier 2

```
Show access-list counters:
  Extended IP access list 110
    Permit ip host 192.168.12.17 any (4222 matches)
```

The result of “show access-list”-command, executed after the test, shows that the number of packets matching to the classifier based on source address was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier 3

```
Show access-list counters:
  Extended IP access list 110
    permit tcp host 192.168.12.17 eq talk host 192.168.8.18 eq talk (4222
    matches)
```

The result of “show access-list”-command, executed after the test, shows that the number of packets matching to the classifier based on source and destination addresses was equal to the number of packets generated in steam 2.

Test router.mechanisms.classifier 4

```
Show access-list counters:
  Extended IP access list 110
    permit tcp host 192.168.12.17 range talk 518 host 192.168.8.18 range
    talk 518 (4222 matches)
```

The result of “show access-list”-command, executed after the test, shows that the number of packets matching to the classifier based on source and destination address range of source and destination port numbers was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier 5

```
Show access-list counters:
  Extended IP access list 110
    permit tcp host 192.168.12.17 eq talk 192.168.8.0 0.0.0.255 eq talk
    (4222 matches)
```

The result of “show access-list”-command, executed after the test, shows that the number of packets matching to the classifier based on source address, and a set of destination addresses was equal to the number of packets generated in stream 2.

Test router.mechanisms.classifier 6

```
Show access-list counters:
  Extended IP access list 110
    permit tcp host 192.168.12.17 eq talk 192.168.8.0 0.0.0.255 eq talk
    precedence flash (4222 matches)
```

The result of “show access-list”-command, executed after the test, shows that the number of packets matching to the classifier based on source and destination address, protocol type and precedence bits was equal to the number of packets generated in stream 3.

4.6.6.4.5 Traffic conditioning mechanism testing

The Cisco 2600 routers offer CAR mechanism for traffic policing and marking. With CAR mechanism the user traffic can be characterised by three parameters: traffic rate, burst size and exceed burst size. The third parameter implements an additional feature in comparison to the standard token bucket algorithm. In the following tests this feature was disabled by setting the exceed burst size equal to burst size.

In the following test the CAR was configuring as a single token bucket with dropping. These settings correspond to TCL1 conditioning function in AQUILA network architecture. The following parameters were set:

- Traffic rate = 1M
- Burst size = 2000B and Exceed burst size = 2000B

The following configuration was set on the test router:

```
interface FastEthernet0/0
 ip address 192.168.12.2 255.255.255.0
 rate-limit input dscp 48 1000000 2000 2000 conform-action
   transmit exceed-action drop
 no ip mroute-cache
 speed 100
 full-duplex
 no cdp enable
```

4.6.6.4.5.1 Router.mechanisms.conditioning.tcl1.1 test

Traffic conforming to the token bucket parameters was generated. The test duration was one minute and the number of dropped packets was recorded. The measurement was repeated for different frame sizes.

The test results are included in Table 16 (transmission rate 970-990 kbps). The CAR mechanism monitors the transmission rate at the frame level. In case of conforming traffic the CAR mechanism works correctly meaning that no packet drops were observed.

4.6.6.4.5.2 Router.mechanisms.conditioning.tcl1.2 test

Traffic not conforming to the token bucket parameters was generated. The test traffic exceeds the rate limit of 1Mbps. The measurement was repeated for different traffic rates and frame sizes. The number of dropped packets, in one-minute observation interval, was recorded.

The test results are presented in the following two tables. Table 16 presents the number of dropped packets in one-minute measurement interval and Table 17 presents the packet dropping rate. The same information is also presented in Figure 47 and Figure 48 respectively.

Table 16. Number of dropped packets in one-minute measurement interval

Trans. rate	64	128	256	512	1024	1500
-------------	-----------	------------	------------	------------	-------------	-------------

970-990	0	0	0	0	0	0
990-1010	0	0	0	0	0	19
1010-1030	0	0	124	177	117	117
1030-1050	0	316	667	348	261	216
1050-1070	0	1266	1211	741	404	315
1070-1090	1762	3288	1754	1023	548	413
1090-1110	3553	4308	2298	1305	692	512
1110-1130	5341	5383	2841	1586	835	611

Table 17. Packet dropping rate

Trans. rate	64	128	256	512	1024	1500
970-990	0	0	0	0	0	0
990-1010	0	0	0	0	0	0.37758
1010-1030	0	0	0.41484	1.18450	1.56606	2.28026
1030-1050	0	0.51965	2.19163	3.00821	3.42790	4.13002
1050-1070	0	2.04775	3.90923	4.77849	5.20753	5.91216
1070-1090	1.38975	5.14989	5.56454	6.47919	6.93495	7.61010
1090-1110	2.76345	6.64200	7.16669	8.12022	8.60162	9.26529
1110-1130	4.09721	8.17116	8.71259	9.69853	10.19661	10.86222

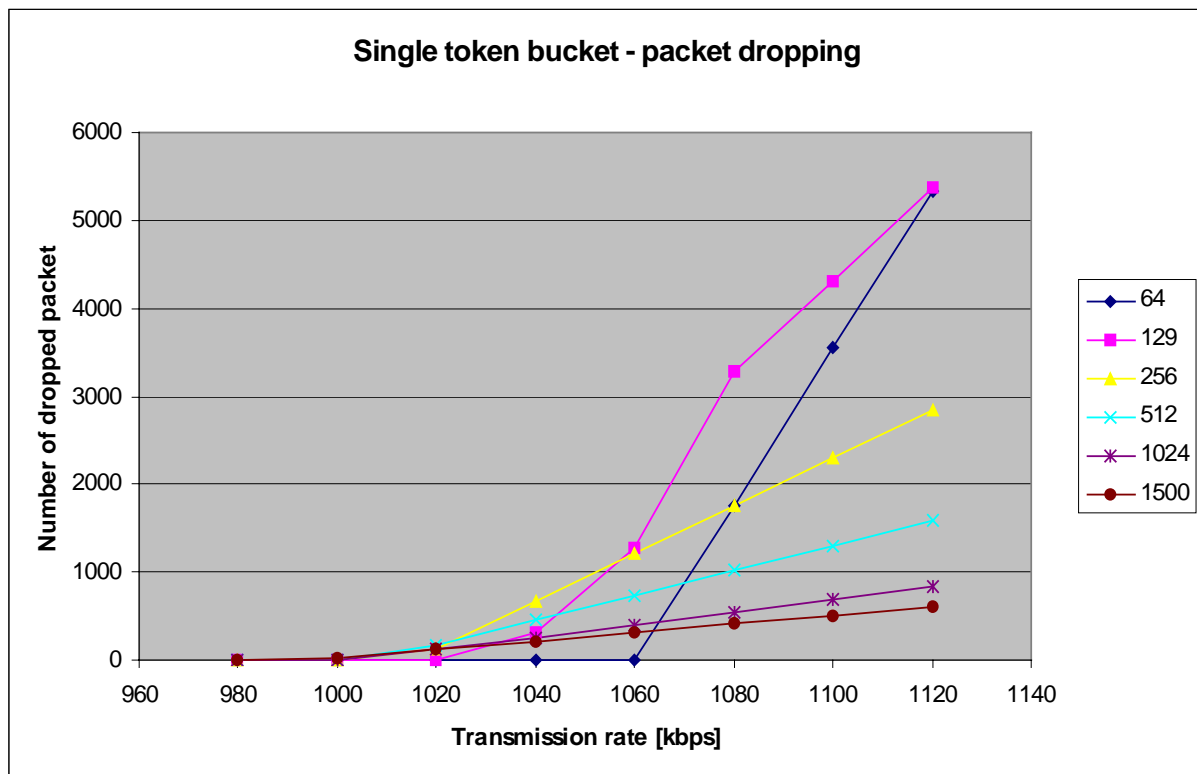


Figure 47. Number of dropped packets in one-minute measurement interval

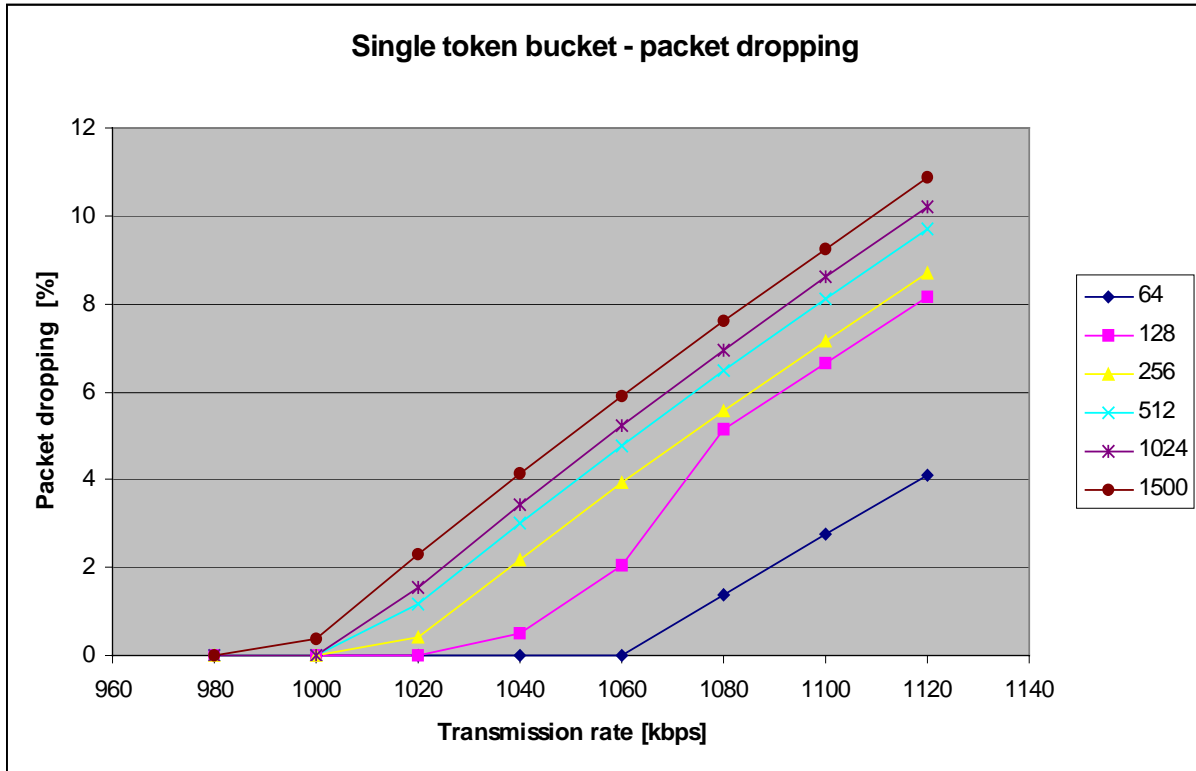


Figure 48. Packet dropping rate

4.6.6.4.6 Queue management testing

4.6.6.4.6.1 Router.mechanims.queue2 test

Weighted Random Early Detection mechanism was tested on the Fast Ethernet port of Cisco 2600 router. The WRED mechanism was configured as follows:

```
Service-policy output: testi (1295)

Class-map: testi (match-any) (1297/4)
 25490 packets, 38133040 bytes
 5 minute offered rate 717000 bps, drop rate 342000 bps
Match: ip precedence 1 (1301)
 12745 packets, 19066520 bytes
 5 minute rate 347000 bps
Match: ip precedence 2 (1305)
 12744 packets, 19066520 bytes
 5 minute rate 347000 bps
Weighted Fair Queueing
  Output Queue: Conversation 265
  Bandwidth 7425 (kbps)
```



```
(pkts matched/bytes matched) 25488/38130048
(depth/total drops/no-buffer drops)
    0/12470/0
exponential weight: 9
mean queue depth: 0
```

Class (Prec)	Random drop pkts/bytes	Tail drop pkts/bytes	Minimum threshold	Maximum threshold	Mark probability
0	0/0	0/0	20	40	1/10
1	21/31416	12449/18623704	50	100	1/10
2	0/0	0/0	4095	4096	1/10
3	0/0	0/0	26	40	1/10
4	0/0	0/0	28	40	1/10
5	0/0	0/0	30	40	1/10
6	0/0	0/0	32	40	1/10
7	0/0	0/0	34	40	1/10
rsvp	0/0	0/0	36	40	1/10

```
policy-map testi
class testi
  bandwidth 7425
  random-detect
  random-detect precedence 1 50 100
  random-detect precedence 2 4095 4096
```

```
interface FastEthernet0/1
 ip address 192.168.8.1 255.255.255.0
 no ip mroute-cache
 speed 10
 full-duplex
 service-policy output testi
 no cdp enable
```

Two traffic streams submitted to the router are described next.

- Stream 1: constant bit rate of 7 893 600 bits/s, packet length 1500B, precedence 1 (WRED: min threshold 50, max threshold 100)
- Stream 2: constant bit rate of 7 893 600 bits/s, packet length 1500B, precedence 2 (WRED: min threshold 4095, max threshold 4096)

It was discovered that with this configuration majority of the packets in stream 1 were lost but all the packets of the stream 2 were able to go through. Changing the minimum and maximum threshold values changed also the throughput as expected.

4.6.6.4.7 Scheduling mechanism testing

The WFQ mechanism was configured with three classes distinguished by precedence bits. The following configuration was used:

```
Service-policy output: PQ (1477)

Class-map: PQ (match-all) (1479/5)
 4385 packets, 6559960 bytes
 5 minute offered rate 122000 bps, drop rate 0 bps
 Match: ip precedence 5 (1483)
```

```
Weighted Fair Queueing
  Strict Priority
  Output Queue: Conversation 264
  Bandwidth 5000 (kbps) Burst 125000 (Bytes)
  (pkts matched/bytes matched) 4385/6559960
  (total drops/bytes drops) 127/189992
Class-map: testi (match-any) (1487/4)
  4385 packets, 6559960 bytes
  5 minute offered rate 122000 bps, drop rate 50000 bps
Match: ip precedence 1 (1491)
  0 packets, 0 bytes
  5 minute rate 0 bps
Match: ip precedence 2 (1495)
  4385 packets, 6559960 bytes
  5 minute rate 122000 bps
Weighted Fair Queueing
  Output Queue: Conversation 266
  Bandwidth 2492 (kbps) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 4384/6558464
  (depth/total drops/no-buffer drops) 0/2009/0
Class-map: class-default (match-any) (1499/0)
  4401 packets, 6560937 bytes
  5 minute offered rate 122000 bps, drop rate 120000 bps
Match: any (1503)
Weighted Fair Queueing
  Output Queue: Conversation 265
  Bandwidth 8 (kbps) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 4385/6559960
  (depth/total drops/no-buffer drops) 0/4314/0
Class-map match-all PQ
  match ip precedence 5
class-map match-any testi
  match ip precedence 1
  match ip precedence 2

policy-map PQ
  class PQ
    priority 5000
  class testi
    bandwidth 2492
  class class-default
    bandwidth 8

interface FastEthernet0/1
  ip address 192.168.8.1 255.255.255.0
  no ip mroute-cache
  speed 10
  full-duplex
  service-policy output PQ
  no cdp enable
```

The link bandwidth was limited to 10M to make it easier to overload the link. All classes together can use at maximum 75% of the whole link bandwidth.

- Stream 1: constant bit rate 5260kbit/sec, preference 2
- Stream 2: constant bit rate 5260kbit/sec, preference 5
- Stream 3: constant bit rate 5260kbit/sec, preference 0

The results for scheduling test are presented in the following two figures. Figure 49 presents the bandwidth share between these three flows and in Figure 50, the latency for the same three flows can be seen.

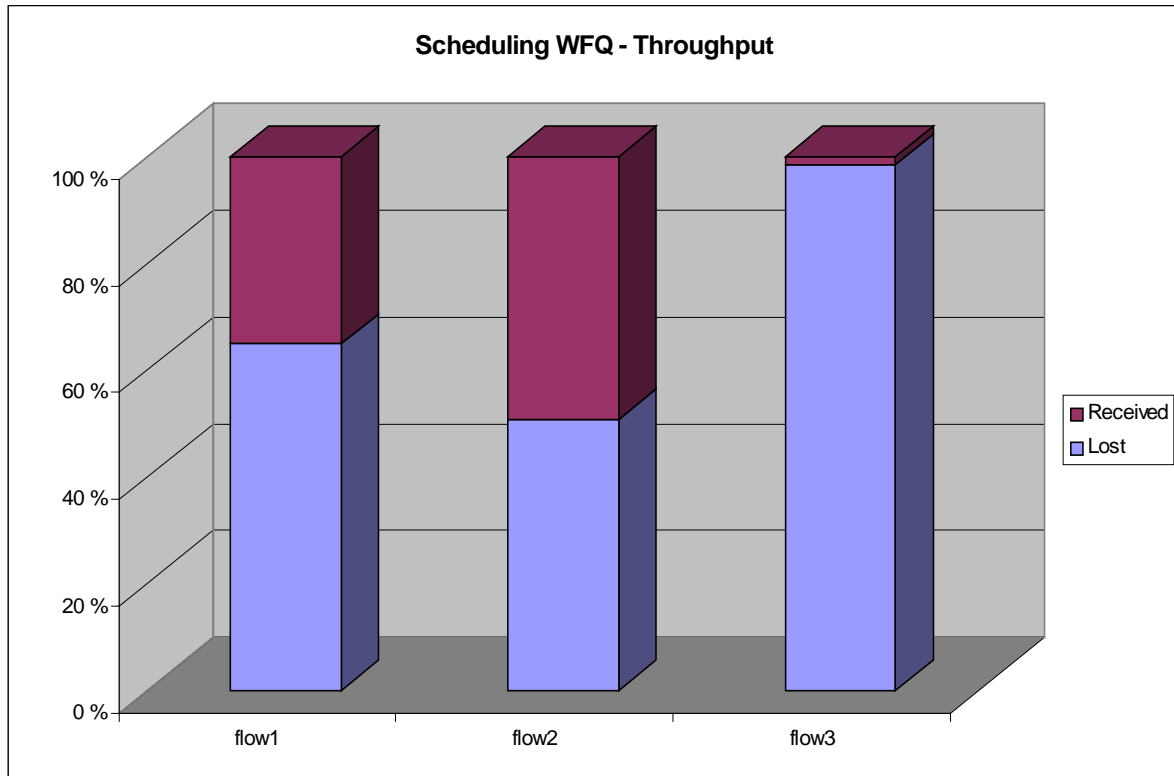


Figure 49. Rate of received and lost packets in three flows

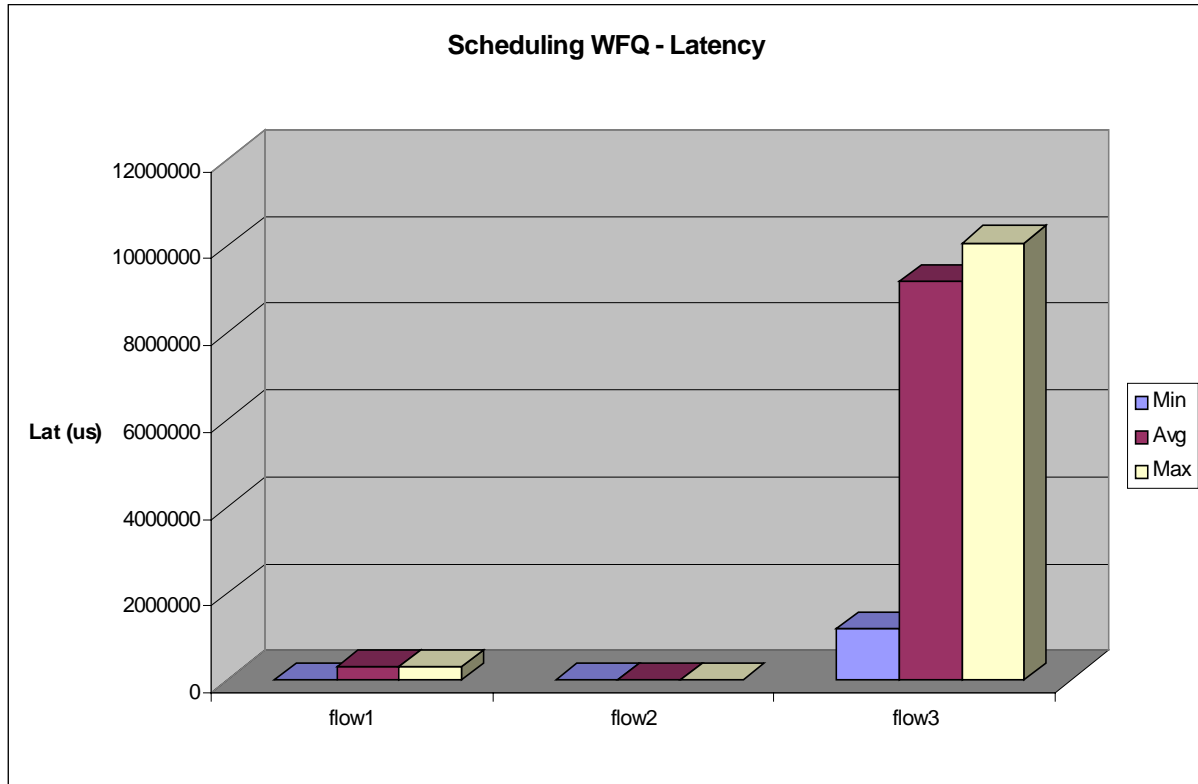


Figure 50. Latency for the three flows

4.6.6.4.8 Number of flows tests

Setting a flow reservation on the router consists of configuring packet classifier and traffic policer for that flow. Policing the flow with a token bucket algorithm requires configuring CAR mechanism. Classifying and policing are processor-intensive operations. Establishing too many flows on a router port may cause performance degradation.

For these tests access-lists and CARs were configured on the routers. For each flow, traffic generator submitted matching packet stream and the throughput is searched as described in router performance tests. Test was repeated with different frame sizes and different numbers of the flows. The results can be seen in Table 18 and Figure 51.

Table 18. Maximum bit rate with different number of flows

Num of flows	64	128	256	512	1024	1500
1	3547597	6810726	12683878	22330573	36049715	44407200
5	2880768	4297216	10673152	19171328	27889664	40302000
10	2428416	4242432	8724480	16359424	26435584	36012000
25	1561600	3015680	5237760	11397120	20520960	26640000

50 1021440 1996800 3962880 7577600 13926400 19440000

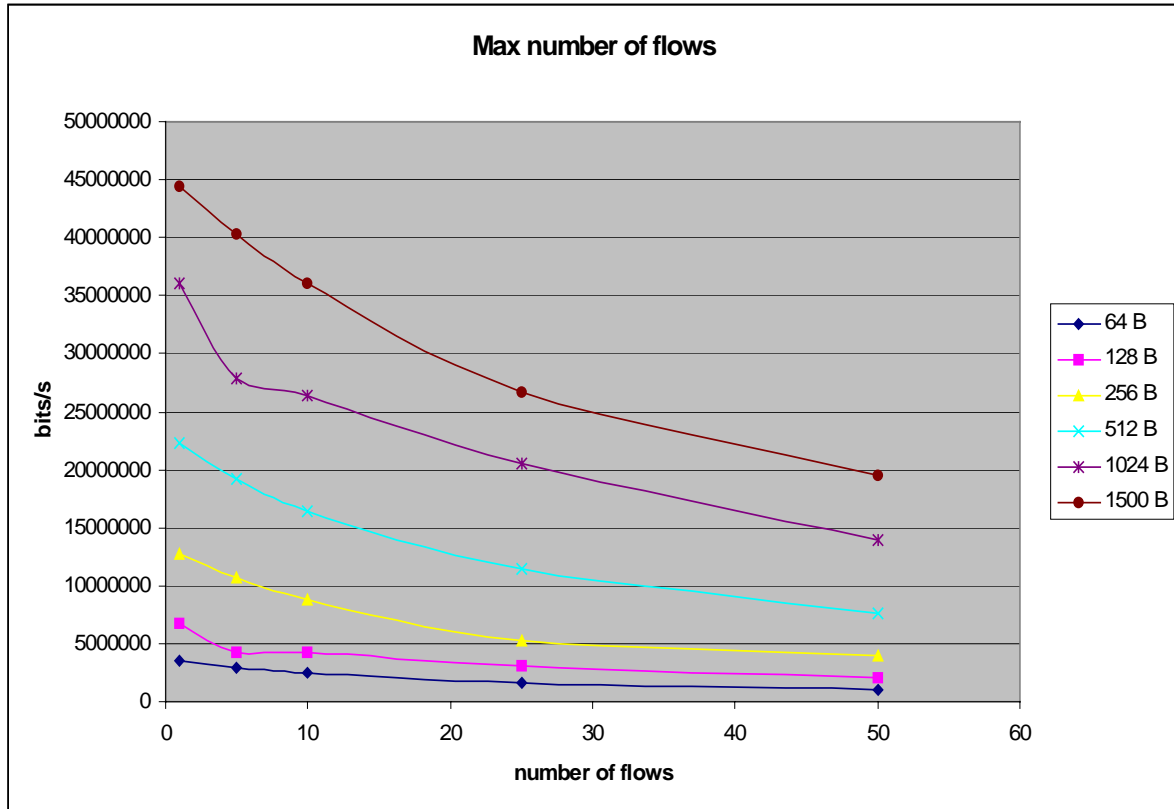


Figure 51. Maximal bit rate for different number of flows

The similar tests were repeated for five flows using DSCP bits instead of access lists. The test results varied so that the throughput was slightly better for some frame sizes when DSCP bits were used but for some frame sizes the results were slightly better by access list. The conclusion was that the impact to the throughput is not notable whether DSCP bits or access lists are used for the policing.

4.6.6.5 CISCO 827 and 1750 Router Tests

4.6.6.5.1 Cisco 827 Router and Test Configurations

For those tests, IOS Software release 12.1.1XB was used.

4.6.6.5.1.1 Router.performance.throughput test

In the throughput test the maximum lossless rate between input and output port of the Cisco 827 router was measured as a function of frame size.

The input and output rates were limited to 10M bits. The results are illustrated in Figure 52 as a function of frame size. It can be seen that the throughput increased as the frame size increased. The throughput reached 100% when the size of the frames was 512 bytes.

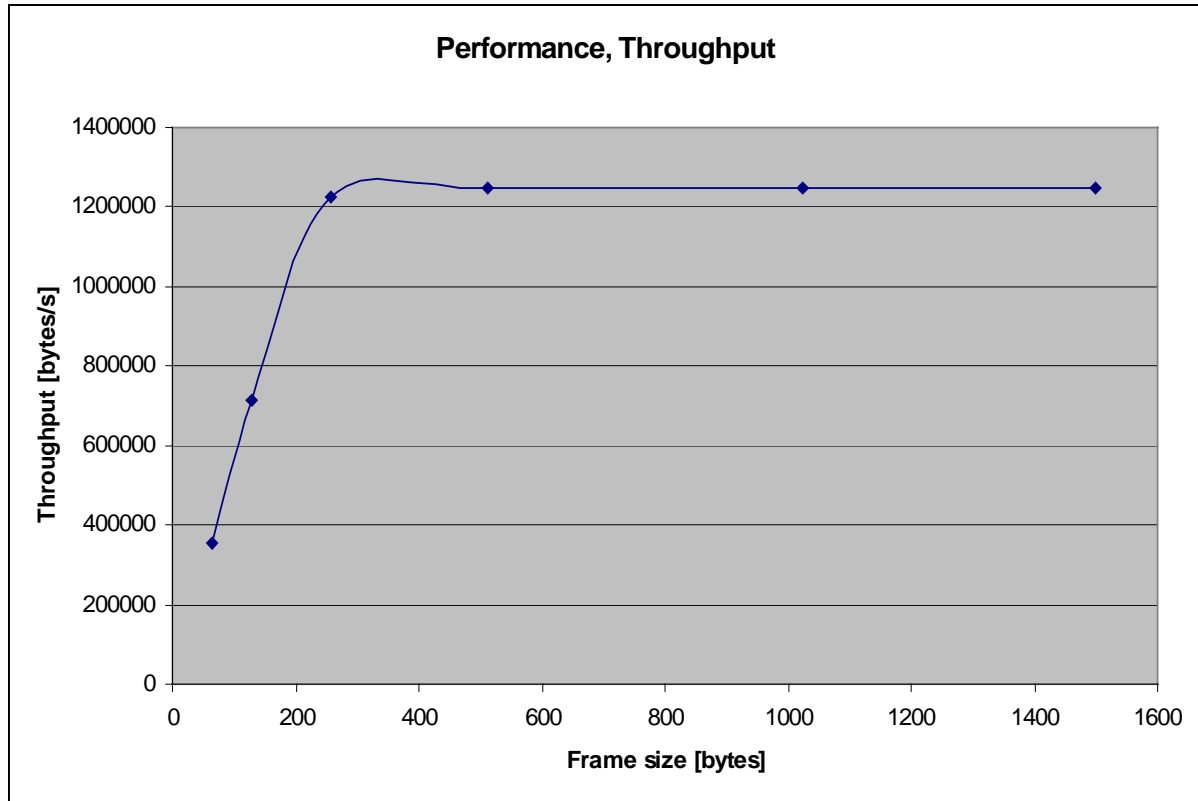


Figure 52. Throughput [bytes] as a function of frame size

4.6.6.5.1.2 Other tests for the Cisco 827

Other tests were also done to the Cisco 827 but unfortunately it was soon found out that it did not support CAR or any other policing mechanism.

Other drawback was that the delay using PQ (CBWFQ + Priority) was unacceptable.

4.6.6.5.2 Cisco 1750 Router and Test Configurations

For those tests, IOS Software release 12.1.3XJ was used.

4.6.6.5.2.1 Tests for the Cisco 1750

Cisco 1750 was tested with the ADSL interface card but we were not able to get the CBWFQ work at all. Every flow was treated equally regardless of the traffic class. Cisco Finland is now investigating it.

4.6.6.6 Summary

The tests of AQUILA routers show that their packets forwarding performance is limited, especially in case of small packets sizes. This effect is easy to observe on the high-speed interfaces (e.g. Fast Ethernet. POS). Depending on the network topology the router performance, not the link capacities, can be a limiting factor that determines the network resources. Moreover, the activation of QoS mechanisms like CBWFQ, CAR etc. have significant influence on the router performance. This is caused by the fact that they are software implemented.

The test of QoS mechanisms shows that generally they are working correctly. Some problems were detected with CBWFQ and CAR mechanisms on the 7507 router. The CBWFQ behaviour deviates from the theoretical characteristics of WFQ algorithm. The CAR mechanism allows for larger burst sizes then equivalent theoretical token bucket algorithm. The similar performance of CAR mechanism can be observed on router 3640. Moreover current version of IOS software for router 3640 doesn't support CBWFQ functionality on Fast Ethernet interfaces. In case of router 3640 the CBWFQ works correctly on standard Ethernet interface.

The router and QoS mechanism performance results should be taken into account in construction and evaluation of AQUILA trial scenarios.

5 Phase 2: RCL-only integration

This section describes the second phase of the integration. It includes the test scenarios for the Resource Control Layer entities. Interaction with the routers was left for the third phase.

5.1 RCA – ACA Interworking

Activity No.	phase2.rca-aca.1
Title	Leaf creation by ACA
Description	The ACA creates the ResourcePoolLeaf. The new object is instantiated
Preconditions	
Postconditions	Reference to the object. An exception may be thrown
Notes	

Activity No.	phase2.rca-aca.2
Title	Consumable Resource Shares retrieval by ACA
Description	The ACA calls the ResourcePoolLeaf.getResourceShares() method
Preconditions	The RCA should be ready to receive requests
Postconditions	Reference to the ResourceShareImpl objects. An exception may be thrown
Notes	Make also a reservation that cannot be accepted (insufficient resources)

Activity No.	phase2.rca-aca.3
Title	Allocation of a resource
Description	Request of resources by calling the ResourceShareImpl.alloc() method
Preconditions	RCA ready to receive requests
Postconditions	Reference to the allocation object. An exception may be thrown
Notes	Performed without the need for a request from the EAT

Activity No.	phase2.rca-aca.4
Title	De-allocation of a resource
Description	Release of resources by calling the ResourceShareImpl.alloc() method
Preconditions	An appropriate request is already accepted
Postconditions	Resource is de-allocated. An exception may be thrown
Notes	Without the need for a request from the EAT

Activity No.	phase2.rca-aca.5
Title	Request / release resources at each Traffic Class specific share
Description	The ACA requests / releases resources at different shares many times in an alternating sequence. Each request of resource should be released
Preconditions	RCA ready to receive requests. ACA ready to send requests
Postconditions	No resources should be requested at this ACA
Notes	

Activity No.	phase2.rca-aca.6
Title	Test the thresholds of each resource share
Description	The ACA requests resources from each traffic class specific share and the consumed resources are up to the limit. So each request of new resources to the different shares initiates a RCA request to the higher RCA level.
Preconditions	Requests are done so that all resources at this ACA are consumed. Resources are available at the higher RCA level.
Postconditions	The user receives a reference to the UserReservation object.
Notes	

Activity No.	phase2.rca-aca.7
Title	All resources are consumed up to the highest level
Description	All resources are consumed at this ACA up to the highest level of the RCA. Each new request has to be rejected
Preconditions	No consumable resources available.
Postconditions	Each request is rejected and a RequestException is thrown to the user
Notes	

5.2 EAT – ACA Interworking

Activity No.	phase2.eat-aca.1
Title	ACA discovery by EAT
Description	Getting the references to the ACA's SessionManager
Preconditions	IP address of ACA and object name known
Postconditions	Reference to the singleton SessionManager received
Notes	

Activity No.	phase2.eat-aca.2
Title	New login session
Description	Login of a new end-user at ACA, see also phase1.eat.eatManager.1
Preconditions	Reference to the ACA's singleton SessionManager, end-user's account name and password, EndUser event observer object, EAT's alive peer object
Postconditions	Reference to a new UserAgent object, ACA's alive peer object
Notes	EAT is multi-user capable All reservations are lost in the case that no alive-peer answers come from the ACA

Activity No.	phase2.eat-aca.3
Title	Login session cancelled by ACA
Description	EndUser.update called by ACA
Preconditions	EndUser is registered as observer at ACA UserLogout event fired

Postconditions	
Notes	Destruction of EndUser object due a timeout, for example

Activity No.	phase2.eat-aca.4
Title	New reservation session
Description	Request for a new reservation at ACA, see phase1.eat.eatmanager.2
Preconditions	Reference to a UserAgent object, network service id, SLS, etc. Reservation event observer object received
Postconditions	Reference to a new UserReservation object
Notes	

Activity No.	phase2.eat-aca.5
Title	Reservation down by ACA
Description	Reservation.update called by ACA
Preconditions	End user is registered as observer at ACA ReservationDown event fired
Postconditions	
Notes	Destruction of Reservation object due an RCL failure

Activity No.	phase2.eat-aca.6
Title	Accounting information retrieval from ACA
Description	see phase1.eat.eatmanager.4
Preconditions	Reference to an UserReservation object
Postconditions	Accounting data
Notes	

Activity No.	phase2.eat-aca.7
Title	Reservation release
Description	see phase1.eat.eatmanager.4
Preconditions	Reference to an UserReservation object
Postconditions	An exception may be thrown
Notes	

Activity No.	phase2.eat-aca.8
Title	Logout
Description	see phase1.eat.eatmanager.5
Preconditions	Reference to an UserAgent object
Postconditions	An exception may be thrown
Notes	

Activity No.	phase2.eat-aca.9
Title	Notification about broken reservation
Description	By using the event observer mechanism, the EAT Manager will be informed about a broken reservation, e.g. when <i>another</i> on the reserva-

	tion participated ACA is broken
Preconditions	Reservation object registered as event observer
Postconditions	
Notes	Reaction: Try to release the reservation at the ACA

Activity No.	phase2.eat-aca.10
Title	Time limited resources reservation request
Description	With respect to phase1.eat-aca.4 the schedule of the reservation has to be defined.
Preconditions	see phase1.eat-aca.4 but with specified schedule
Postconditions	see phase1.eat-aca.4
Notes	This feature is not yet implemented. The event observer mechanism will be used to inform the EAT about an expired reservation.

Activity No.	phase2.eat-aca.11
Title	EAT KeepAlive mechanism to ACA
Description	A user logs in (phase1.aca.eat.1)
Preconditions	The EAT and the manager ACA are placed on different devices
Postconditions	The keep alive mechanism between the EAT and the Host is active “hello” messages are transmitted between the two RCL components in the coded time frame
Notes	

Activity No.	phase2.eat-aca.12
Title	EAT KeepAlive mechanism to ACA
Description	Many users log in and use the same EAT (phase1.aca.eat.1)
Preconditions	The EAT and the manager ACA are placed on different devices
Postconditions	The keep alive mechanism between the EAT and the Host is active. “hello” messages are transmitted between the two RCL components in the coded time frame
Notes	

Activity No.	phase2.eat-aca.13
Title	Interruption of the KeepAlive mechanism between EAT and ACA
Description	A user logs in and the communication between the EAT and ACA breaks down.
Preconditions	The EAT and the manager ACA are placed on different devices
Postcon	The keep alive mechanism between the EAT and the Host is active; “hello” messages are no longer transmitted between the two RCL components; the ACA logs the user out and releases all requested resources of this user.
Notes	

Activity No.	phase2.eat-aca.14
Title	Interruption of the KeepAlive mechanism between EAT and ACA, the EAT and ACA are on the same host
Description	A user logs in and the Keep Alive messages are transmitted between the EAT and the ACA, and then the EAT breaks down.
Preconditions	The EAT and the manager ACA are placed on the same devices
Postconditions	No messages are transmitted from ACA to EAT, the ACA logs out the user and releases the requested services
Notes	

5.3 EAT - Service interworking

Activity No.	phase2.eat-service.1
Title	Service package discovery by EAT
Description	Getting the reference to the service's ServiceManager
Preconditions	IP address of service component and object name known
Postconditions	Reference to the singleton ServiceManager
Notes	

Activity No.	phase2.eat-service.2
Title	Registration as observer
Description	EAT registers itself as EventObserver at the ServiceManager
Preconditions	Reference to the singleton ServiceManager received
Postconditions	
Notes	

Activity No.	phase2.eat-service.3
Title	Network service retrieval
Description	see phase1.eat.eatManager.7
Preconditions	
Postconditions	Network services as singleton or sequence are received
Notes	Network services are mirrored within the EAT

Activity No.	phase2.eat-service.4
Title	Notification for network services change
Description	The EAT as EventObserver will be updated/notified when a new network service occurs or a relative event is fired
Preconditions	EAT is registered as observer at service Event NewNetworkService is fired
Postconditions	
Notes	

5.4 End - to - end tests

Activity No.	phase2.end-to-end.1
--------------	---------------------

Title	Reservation requests with p2p style to an egress where no egress resources are available
Description	The user requests a Network Service, which needs the p2p reservation style. The egress side is not available
Preconditions	The user is successfully logged in
Postconditions	A RequestException is thrown to the user.
Notes	

Activity No.	phase2.end-to-end.2
Title	Reservation request with the third party model
Description	The EAT, the ACA manager, the ingress ACA and the Egress ACA are on different hosts. The user requests an authorised network service
Preconditions	The user is logged in
Postconditions	The user receives a reference to the UserRequest object
Notes	

Activity No.	phase2.end-to-end.3
Title	Reservation request with the second party model
Description	The EAT, the ACA manager, the ingress ACA or the Egress ACA are on different hosts. The user requests an authorised network service.
Preconditions	The user is logged in.
Postconditions	The user receives a reference to the UserRequest object.
Notes	

Activity No.	phase2.end-to-end.4
Title	Reservation request with the one party model
Description	The ACA manager, the ingress ACA and the Egress ACA are on the same host. The user requests an authorised network service.
Preconditions	The user is logged in
Postconditions	The user receives a reference to the UserRequest object but no resources are consumed by this request.
Notes	

Activity No.	phase2.end-to-end.5
Title	Small bandwidth link reservation
Description	The user requests a network service on a small bandwidth link
Preconditions	The user is logged in
Postconditions	The user receives a reference to the UserRequest object
Notes	

Activity No.	phase2.end-to-end.6
Title	Loaded Network
Description	Request reservations with the condition that the network load in the core network is very high

Preconditions	The user is logged in. Network is heavily loaded
Postconditions	
Notes	

5.5 Failure scenarios

The following scenarios cover the cases where an RCL entity breaks down.

5.5.1 Recovery Scenarios

5.5.1.1 RCA Breakdown

Activity No.	phase2.rca.breakdown.1
Title	Enough resources at the ACA
Description	The RCA breaks down while enough resources are stored at the ACA
Preconditions	The RCA is not available
Postconditions	A new resource request to all the different shares is possible. The user receives a reference to the UserReservation object.
Notes	

Activity No.	phase2.rca.breakdown.2
Title	NOT enough resources at the ACA
Description	The RCA breaks down, and not enough resources are stored at the ACA
Preconditions	The RCA is not available.
Postconditions	A new resource request to any share is not possible. The user receives NO reference to the UserReservation object.
Notes	

Activity No.	phase2.rca.breakdown.3
Title	Resource pool distribution
Description	The RCA breaks down and restarts again.
Preconditions	The resources shares have been consumed at the ACA side and the resource shares have a specific value.
Postconditions	After the restart of the RCA the specific level of the resources shares is not modified.
Notes	

5.5.1.2 ACA Breakdown

Activity No.	phase2.aca.breakdown.1
Title	ACA breakdown
Description	Repeat the failure scenarios described in test cases phase2.rca.breakdown.1 to 3
Preconditions	

Postconditions	
Notes	

5.5.1.3 EAT Breakdown

Activity No.	phase2.eat.breakdown.1
Title	EAT recovery
Description	The EAT continually saves the current state (users, reservations) in the persistence layer. After a breakdown and a restart, the EAT tries to re-establish all users and reservations.
Preconditions	Actual state in the persistence layer stored
Postconditions	The EAT has the same status as before the breakdown
Notes	Reservation requests might be rejected. If so, the state before the EAT breakdown cannot be recovered

6 Phase 3: Overall integration

During the third phase of system integration routers are used. The main purpose of this phase is to verify the correct operation of the RCL and of its interaction with the DiffServ layer.

6.1 RCL – DiffServ Layer Interworking

Activity No.	phase3.rcl-diffserv.1
Title	Number of reservation requests satisfied
Description	<p>For each flow the EDAdapter uses one access list. The number of access lists is limited. For each flow one policer is configured. The number of policer is not limited, but maybe the processor of the router is just able to handle a limited number of policer.</p> <p>In this test the number of end-to-end reservations should be increased. The reservation can be always the same and may use a small amount of bandwidth.</p>
Preconditions	
Postconditions	
Notes	No applications involved, just the Reservation GUI

Activity No.	phase3.rcl-diffserv.2
Title	Address not in the routing table
Description	<p>The EDAdapter uses the routing table of the ED to configure the router. When a request arrives from a source address out of the router, the ACA denies the request.</p> <p>The request should come from a source out of the routing table (just adding a new host to the network). After a while, the address should be in the table and a second request should be accepted.</p>
Preconditions	No entry in the address table
Postconditions	
Notes	No applications involved, just the Reservation GUI

Activity No.	phase3.rcl-diffserv.3
Title	Telnet connection
Description	Each EDAdapter is connected to the ED with a telnet connection. After a break down of the connection the EDAdapter reconnects to the router.
Preconditions	
Postconditions	
Notes	No applications involved, just the Reservation GUI

6.2 End – to – end tests

In general, the test scenarios listed in chapter 5: RCL-only integration should also be used for the end-to-end tests in the overall test case. In the present chapter, additional scenarios are presented, that have to do with the incorporation of routers in the trial.

6.2.1 RCA

No further tests are foreseen in the case of the use of routers in the network topology. The test cases listed in chapter 5, may be used.

6.2.2 ACA

Activity No.	phase3.end-to-end.aca.1
Title	Telnet connection
Description	After the ACA has lost the connection to the ED, the end user should be able to finish the session. The ACA removes the reservation after the reconnection.
Preconditions	
Postconditions	
Notes	Any application may be used

Activity No.	phase3.end-to-end.aca.2
Title	Short sessions
Description	It is possible that the configuration of the ED could be a bottleneck of the system. The number of request for short flows should be increased. The flow should be released after a short period of time. To see the influence of the network the test should repeated for different end to end connections.
Preconditions	
Postconditions	
Notes	Reservation GUI may be used

6.2.3 EAT

6.2.3.1 EAT Manager

Activity No.	phase3.end-to-end.eat.eatManager.1
Title	Reservation request for NetMeeting
Description	A reservation for NetMeeting is requested via the GUI. NetMeeting is registered at the Proxy.
Preconditions	Reservation request via the GUI
Postconditions	Provisional reservation for NetMeeting, NetMeeting registered at the Proxy. EAT Manager waits for response
Notes	

Activity No.	phase3.end-to-end.eat.eatManager.2
Title	Response received by the Proxy for NetMeeting reservation
Description	The Proxy informs the EAT Manager about the relevant data.
Preconditions	Provisional reservation for NetMeeting created, NetMeeting registered at the Proxy
Postconditions	Complete reservation request
Notes	

Activity No.	phase3.end-to-end.eat.eatManager.3
Title	Proxy failure
Description	The Proxy crashes. The EAT Manager should not wait for a response by the proxy
Preconditions	Proxy crashes
Postconditions	The EAT Manager should operate without problem
Notes	

6.2.3.2 Proxy

Activity No.	phase3.end-to-end.eat.proxy.1
Title	IP flow detection
Description	The Proxy detects a matching flow. It collects the requested information (addresses, ports) and forwards it to the EAT Manager.
Preconditions	Proxy has start up. Application registration created beforehand (phase3.end-to-end.eat.eatManager.1)
Postconditions	Flow is moved from the registration list to the active flow list.
Notes	

6.2.3.3 GUIs

Nothing changes in the case of the GUIs (Login, Reservation, Release) in the case of the overall testing. The test scenarios are the same as those in paragraph 4.4.2.2.

6.2.4 Measurement tools

6.2.4.1 CM Toolset

Activity No.	phase3.end-to-end.measurement.cm.1
Title	CMCaller waiting for new flows
Description	The CMCaller is waiting for flows to be started by periodically requesting the contents of the flow table in the measurement database.
Preconditions	CMCaller running
Postconditions	If an actual entry is found in the database, the Caller changes to state with activity no. phase3.end-to-end. phase3.end-to-end.measurement.it lasts in this state until the CMCaller is terminated.
Notes	

Activity No.	phase3.end-to-end.measurement.cm.2
Title	Reservation set-up request through CMCaller
Description	The user has the possibility to order a reservation for the measurement flows. The reservation is done via a script interface provided by the ACA.
Preconditions	Activity phase3.end-to-end. phase3.end-to-end.measurement.and a minimum of one flow with a specified reservation, that is supposed to be started.
Postconditions	The reservation is established, the ACA script interface returns OK.
Notes	

Activity No.	phase3.end-to-end.measurement.cm.3
Title	Distribution of flow information
Description	The CMCaller reads flow-specific-information out of 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 (is it still up?). After that, the flow-specific data is communicated firstly to the receiving agent and secondly to the sending agent.
Preconditions	Activity phase3.end-to-end.measurement.cm.1 and a minimum of one flow that is supposed 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, or else an according error state is set.
Notes	

Activity No.	phase3.end-to-end.measurement.cm.4
Title	Flow initialisation and start-up
Description	CMDaemon receives flow initialisation from CMCaller and response with success or an error number. CMDaemon starts the measurement flow as defined either as sender or receiver.
Preconditions	Activity phase3.end-to-end.measurement.cm.3
Postconditions	The running measurement flow
Notes	

Activity No.	phase3.end-to-end.measurement.cm.5
Title	Storage of measurement results
Description	When a measurement has finished or when aggregated results are supposed to be transmitted, the CMDaemon sends data to the Resultserver.
Preconditions	End of the running measurement flow or availability of aggregated results.
Postconditions	Flow-specific result data is inserted into the result-table of the database and the flowstate is updated.
Notes	

Activity No.	phase3.end-to-end.measurement.cm.6
Title	Kill a running measurement flow
Description	CMDaemon receives a request to kill a flow from CMCaller and sends as response a defined error number. CMDaemon kills the flow.
Preconditions	Measurement flow running, activity phase3.end-to-end.measurement.cm.4
Postconditions	Flow and error state of the measurement flow is updated
Notes	

Activity No.	phase3.end-to-end.measurement.cm.7
Title	Path discovery
Description	CMDaemons can be requested (via the GUI) to start a path discovery to a specified receiver. The CMDaemon gets the IP addresses of the hops along the path (using “traceroute”) to the receiver and reports them to the Resultserver.
Preconditions	Activity phase3.end-to-end.measurement.cm.1 and a minimum of one flow in the database, which requests a path discovery.
Postconditions	The hop-information for this flow is stored to the path-table.
Notes	

6.2.4.2 T-Nova Toolset

Activity No.	phase3.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	GUI available
Postconditions	Measurement clients are known to the system and measurement tests/flows can be defined by the configuration GUI. (For details see Chapter 8.1.5)
Notes	To create new flows it is necessary, that tests, traffic and hops are already available in the database.

Activity No.	phase3.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 GPS synchronisation is ok). After that, the master sends the flow-configuration data to the receiving agent and then to the sending agent.
Preconditions	Activity phase3.end-to-end.measurement.tnova.1 and a minimum of one flow that 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 eventlog of the database.
Notes	

Activity No.	phase3.end-to-end.measurement.tnova.3
Title	Flow initialisation and start-up
Description	Measurement clients 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.
Preconditions	Activities: phase3.end-to-end.measurement.tnova.1-2
Postconditions	Running measurement flow(s). Measurement results are stored in the database.
Notes	

Activity No.	phase3.end-to-end.measurement.tnova.4
Title	Kill a running measurement flow
Description	The master detects that a measurement flows has to be stopped. The master sends the request to the sender clients and after the sender is stopped to the receiving measurement client.
Preconditions	Measurement flow running Activity: phase3.end-to-end.measurement.tnova.3
Postconditions	Flow- and error state of the measurement flow is updated in the database
Notes	

6.2.4.3 Router QoS Monitoring Tool

Activity No.	phase3.end-to-end.measurement.router.1
Title	Starting router monitoring when a flow starts
Description	The QoS Monitoring Tool daemon checks the database for new flows in regular intervals. If a new flow is detected, the daemon starts the Monitoring Tool for every router in the flow path. The tool is run again after specified intervals, unless the flow is marked as completed.
Preconditions	The database is accessible, and flow information is available
Postconditions	Requested router parameter values are written to the database
Notes	

6.2.4.4 GUI

Activity No.	phase3.end-to-end.measurement.gui.1
Title	Configuration GUI
Description	The configuration GUI is used to specify new tests. Tests consist of flows between specified hosts.
Preconditions	GUI available
Postconditions	Tests scenarios are written to the database.

Notes	To create new flows it is necessary, that tests, traffic and hops are already available in the database.
-------	--

Activity No.	phase3.end-to-end.measurement.gui.2
Title	GUI for display of results
Description	One GUI component is the graphical display of the measurement results (e.g. one way delay, goodput, jitter per packet).
Preconditions	Result data available in the database.
Postconditions	GUI outputs graphical display of measurement results
Notes	

7 Software maintenance and bug report

In the course of the integration, a software library was created. In this FTP server, which was maintained by TUD, all partners uploaded new versions of their software components. For the efficient organisation of the integration process, a consistent naming scheme was maintained. All packages were compressed mainly in ZIP format and were named like the following:

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

A template was created for reporting new software versions (IDL and Java files):

Component Name (or file name)		
Version	Date	Comments

Finally, a template was also created for reporting important bugs to the WP 3.1 people:

Component Name	
Component Version	
Originator	
Description	
Status	
Comments	

8 Procedures for set-up of new trial sites

8.1 Installation and use of RCL components

8.1.1 General Instructions

8.1.1.1 Fetching the RCL components

The components of the Resource Control Layer are all written in Java. You should make sure that JDK 1.3.0 is installed in the PCs / Workstations. The Java classes that make up the RCL components are grouped in Java packages, according to which 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/  
  rcl/  
    rca/  
    aca/  
    network/  
    subscriber/  
    tc/  
    util/  
    eat/  
      eatManager/  
      api/  
      proxy/  
      jsp/  
      gui/  
      script/
```

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

For the inter-communication of the RCL components, CORBA is used. The IDL files of the RCL interfaces are also stored in the FTP server using the same convention:

nameOfComponent-xx.yy.zz.idl

8.1.1.2 Installation and compilation

The installation of the files and the operation of the RCL will be eased, if a directory structure like the following is used.

```
$HOME/  
  classes/  
  src/  
  log/  
  external/  
  support/  
    ldif/  
    rc/  
    scripts/
```

This structure is not mandatory, but will be very helpful. The \$HOME is the environment variable of the specific account on which this installation will be done, in the case of workstations. A similar structure may also be used in the case of PCs. The *src* directory is used for the source files, while the *classes* directory will host the Java classes.

First of all, the IDL files need to be compiled, so that the AQUILA package structure is formed and the CORBA skeleton and stub files are created.

The IDL files should be renamed, by erasing the date from their file name. For instance, *aca-00.09.15.idl* should be renamed into *aca.idl*. The files should then be placed in the *src* directory. In the same directory you should create a text file called *idl.config*, which will contain the following lines:

```
PkgPrefix.rca=aquila.rcl  
PkgPrefix.aca=aquila.rcl  
PkgPrefix.acaInternal=aquila.rcl  
PkgPrefix.eat=aquila.rcl  
PkgPrefix.network=aquila.rcl  
PkgPrefix.subscriber=aquila.rcl  
PkgPrefix.tc=aquila.rcl  
PkgPrefix.util=aquila.rcl  
PkgPrefix.utilInternal=aquila.rcl  
PkgPrefix.service=aquila.rcl  
PkgPrefix.api=aquila.rcl.eat  
PkgPrefix.converter=aquila.rcl.eat  
PkgPrefix.eatManager=aquila.rcl.eat  
PkgPrefix.proxy=aquila.rcl.eat
```

The compilation of the IDL files can be performed with this line:

```
idlj -fall *.idl
```

The compilation results in the formation of sub-directories, reflecting the package structure. Afterwards, you should unzip the Java packages and copy the Java files in their corresponding directory. For example, the ACA files should be copied to *src/Aquila/rc/aca*.

In general, for the compilation of a Java file that belongs in a package structure, you should switch to the *src* directory and then give the command:

```
javac -d ../classes aquila/rc/nameOfComponent/javaFile.java  
(SUN workstations)
```

```
javac -d ..\classes aquila\rcl\nameOfComponent\javaFile.java  
(PCs)
```

Note that more than one Java files **in the same package** can be compiled at once, by using wildcards.

8.1.1.3 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.rcl.nameOfComponent.nameOfClass
```

where:

- 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)

8.1.1.4 Property files

The operation of the RCL requires the presence of a configuration file that will provide information on the name of the particular RCL component, the Name Server used, the LDAP server and others. The default name of the property file is **aquila.rc**, but other names can also be used to differentiate between RCL entities. One property file for each RCA, ACA or EAT is needed. These property files should be stored in the *support/rc* directory. The property file includes the information:

- on which host / port the CORBA Naming Service (TnameServ) is running
- which LDAP factory is used
- the URL to the root entry of the LDAP Directory Server
- the ACA name (module dependent)
- the RCA name (module dependent)
- the EAT name (module dependent)
- the Proxy name (module dependent)
- the logging directory
- the info if the ldapNotify feature has been activated

With the term “module dependent” we mean that the inclusion of this information in the property file, depends on which AQUILA component is used

An example property file for an ACA would be:

```
org.omg.CORBA.ORBInitialHost:      sun1
org.omg.CORBA.ORBInitialPort:      1234
java.naming.factory.initial:        com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:           ldap://sun1:389/o=AQUILA,c=EU
aquila.rcl.aca.name:                aca.warsaw
aquila.rcl.rca.name:                rca.athens
aquila.rcl.trace.dir:               <home>/log
aquila.rcl.ldapNotify:              true
```

As can be seen, this ACA, named *aca.warsaw* connects to *rca.athens*. Also, *<home>* identifies the home directory of the installation. The name of the file could be *aca.warsaw.rc*

8.1.1.5 Name Server

The Name Server should run on one Workstation or PC with the following command:

```
tnameserv -ORBInitialPort <number>
```

where *<number>* is a selected port number. The same number should be used in the *aquila.rc* files of all components.

8.1.1.6 LDAP Server

This section assumes that the Netscape Directory Server is already installed on a SUN workstation and properly configured. Installation information is included in appendix. The present paragraph focuses only on the specific settings required for the AQUILA project

8.1.1.6.1 Configuration of the Database

From the console, open the Netscape Directory Server window. Click on the Configuration Tab, click on Database icon and click the Passwords Tab. In Password Entry set the password encryption to “clear text”.

8.1.1.6.2 Configuration of the Database Schema

From the same console, using the Schema icon and Attribute Tab, create a new attribute:

```
Name: authLoginService |OID:1.3.6.1.4.42.2.27.1.1.31| Syntax: Case Exact
String |MULTI set
```

From the Object Tab, a new object may be created:

```
Name: remoteUser | OID:1.3.6.1.4.1.42.2.27.1.2.8
with the required Attributes: objectclass $ uid
and with allowed Attributes: authLoginService $ userpassword
```

Also, before the configuration of the Resource Pools and their storage at the database, there are some additions that need to take place at the schema of the database. These additions comprise the creation of an object class and of its attributes.

The first step is the creation of the attributes of the aforementioned object class. The names and the types of these attributes are mentioned below.

Name	Syntax	Multi
poolID	Case Ignore String	No
fatherID	Case Ignore String	No
mID	Case Ignore String	No
rpoolType	Integer	No

The next step is the creation of the object class itself. The name and the required attributes of this class can be viewed at the following table. The attribute “cn” already exists at the schema of the database. To be noted here that these five attributes should be added to the list of the “Required” attributes and not to the list of the “Allowed” attributes of the object class.

Name	Required Attributes
poolob	cn
	poolID
	fatherID
	mID
	rpoolType

Furthermore, an entry should be created at the database for the storage of the Resource Pools. This entry’s object class could simply be a “javacontainer” and its attribute, “cn”, should take the value “Pool”. Finally, it should be stored under the entry with the distinguished name “o=Aquila, c=EU”.

8.1.1.6.3 Configuration of the Directory

A new user should be added to the “Directory Administrators” folder. Also, set the access permissions for the folder “Subscriber”, so that anybody can compare, read and search the content of this folder.

8.1.1.6.4 AQUILA LDAP Directory Structure

Now the Directory schema should be created. Note that the AQUILA root directory entry (o=aquila, c=eu) should have already been set up during the installation of the LDAP Server as stated in the appendix.

The structure *under the root* is then subdivided in the following way:

```
o=aquila, c=eu
  cn=AdmissionCtrlAgent
  cn=Pool
  cn=Router
  cn=Service
  cn=Subnet
  cn=Subscriber
```

The Router entry is subdivided in:

```
  cn=Telnet
  cn=COPS
```

The Service entry is subdivided in:

```
  cn=NetworkService
  cn=TrafficClasses
```

In order to store this information into the LDAP Directory Server Database use this command:

```
ldapadd -D"uid=xyz,ou=Directory Administrator,o=aquila,c=eu"
        -f aquilaLDAP.ldif -c
```

The content of the *aquilaLDAP.ldif* file is identical for all test beds:

```
dn: cn=AdmissionCtrlAgent,o=AQUILA,c=EU
objectClass: javaContainer
cn: AdmissionCtrlAgent

dn: cn=Service,o=AQUILA,c=EU
objectClass: javaContainer
cn: Service

dn: cn=NetworkService,cn=Service,o=AQUILA,c=EU
objectClass: javaContainer
cn: NetworkService

dn: cn=TrafficClass,cn=Service,o=AQUILA,c=EU
objectClass: javaContainer
cn: TrafficClass

dn: cn=Pool,o=AQUILA,c=EU
objectClass: javaContainer
cn: Pool

dn: cn=Subscriber,o=AQUILA,c=EU
objectClass: javaContainer
cn: Subscriber

dn: cn=Subnet,o=AQUILA,c=EU
objectClass: javaContainer
cn: Subnet

dn: cn=Router,o=AQUILA,c=EU
```

```
objectClass: javaContainer
cn: Router
```

```
dn: cn=telnet,cn=Router,o=AQUILA,c=EU
objectclass: javaContainer
cn: telnet
```

```
dn: cn=COPS,cn=Router,o=AQUILA,c=EU
objectclass: javaContainer
cn: COPS
```

8.1.1.6.5 Information to the AdmissionCtrlAgent entry (specific for each test bed):

For different ACAs to run in a test bed, the following command line has to be entered:

```
ldapadd -D"uid=xyz,ou=Directory Administrator,o=aquila,c=eu"
-f acaTrialSite.ldif -c
```

The content of the *acaTrialSite.ldif* file is (in this case for 4 ACAs):

```
dn: cn=aca.vienna,cn=AdmissionCtrlAgent,o=AQUILA,c=EU
objectclass: device
cn: aca.vienna
description: routerFactoryClass=aquila.rcl.aca.TelnetRouterFactory
description: routerType=ED1TAAios1201
description: routerHost=10.1.0.1
description: routerName=ED1TAA
description: routerAuthentication=cisco:null/cisco
```

```
dn: cn=aca.helsinki,cn=AdmissionCtrlAgent,o=AQUILA,c=EU
objectclass: device
cn: aca.helsinki
description: routerFactoryClass=aquila.rcl.aca.TelnetRouterFactory
description: routerType=ED2TAAios1201
description: routerHost=10.1.1.1
description: routerName=ED2TAA
description: routerAuthentication=cisco:null/cisco
```

```
dn: cn=aca.warsaw,cn=AdmissionCtrlAgent,o=AQUILA,c=EU
objectclass: device
cn: aca.warsaw
description: routerFactoryClass=aquila.rcl.aca.TelnetRouterFactory
description: routerType=ED3TAAios1201
description: routerHost=10.1.2.1
description: routerName=ED3TAA
description: routerAuthentication=cisco:null/cisco
```

```
dn: cn=aca.athens,cn=AdmissionCtrlAgent,o=AQUILA,c=EU
objectclass: device
cn: aca.athens
description: routerFactoryClass=aquila.rcl.aca.TelnetRouterFactory
description: routerType=CORE1TAAios1201
description: routerHost=10.1.0.254
description: routerName=CORE1TAA
description: routerAuthentication=aquila:null/aquila
```

8.1.1.6.6 Information to the Pool entry (specific for each test bed)

The pool structure is created and stored in the LDAP database with the QMTool. (See the section on QMTool).

8.1.1.6.7 Information to the Router entry (specific for each test bed)

The Router entry is described in detail in paragraph 8.1.3.3.

8.1.1.6.8 Information to the Service entry (identical for each test bed)

The network services and the traffic classes including the information that is bound to them have to be stored in the LDAP database. For that purpose two classes: `GenerateServices.java` and `GenerateTrafficClasses.java` process this operation. As a result, four network service software objects and four traffic class software objects are stored in the AQUILA LDAP database.

The network service and the traffic classes can be stored with a program in this way:

- Compile the source:

```
javac -d ../classes aquila/rcl/service/GenerateServices.java
```

- Change to directory `src` and start this command line to generate the Network Services:

```
java -classpath classes
  -Duser.dir=.
  -Daquila.rcl.propertyFile =${HOME}/support/rc/acaXyz.rc
  aquila.rcl.service.GenerateServices
```

Also generate the traffic classes:

- Compile the source:

```
javac -d ../classes aquila/rcl/service/GenerateTrafficClasses.java
```

- Change to `src` and start this command line to generate the Traffic Classes:

```
java -classpath classes
  -Duser.dir=.
  -Daquila.rcl.propertyFile =${HOME}/support/rc/acaXyz.rc
  aquila.rcl.tc.GenerateTrafficClasses
```

8.1.1.6.9 Information to the Subnet entry (specific for each test bed):

The subnet structure of the test bed should also be stored in the database:

```
ldapadd -D"uid=xyz,ou=Directory Administrator,o=aquila,c=eu"
  -f subnetTrialSite.ldif -c
```

The content of the `subnetTrialSite.ldif` file is specific for each test bed:

```
dn: cn=subnet.vienna.192.168.3, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
cn: subnet.helsinki.192.168.3
ipNetworkNumber: 192.168.3.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.vienna
```

```
dn: cn=subnet.vienna.192.168.5, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
```

```
cn: subnet.helsinki.192.168.5
ipNetworkNumber: 192.168.5.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.vienna
```

```
dn: cn=subnet.helsinki.192.168.2, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
cn: subnet.helsinki.192.168.2
ipNetworkNumber: 192.168.2.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.helsinki
```

```
dn: cn=subnet.helsinki.192.168.6, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
cn: subnet.helsinki.192.168.6
ipNetworkNumber: 192.168.6.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.helsinki
```

```
dn: cn=subnet.warsaw.192.168.1, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
cn: subnet.warsaw.192.168.1
ipNetworkNumber: 192.168.1.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.warsaw
```

```
dn: cn=subnet.warsaw.192.168.11, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
cn: subnet.warsaw.192.168.11
ipNetworkNumber: 192.168.11.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.warsaw
```

```
dn: cn=subnet.warsaw.192.168.12, cn=Subnet,o=AQUILA,c=EU
objectclass: ipNetwork
cn: subnet.warsaw.192.168.12
ipNetworkNumber: 192.168.12.0
ipNetmaskNumber: 255.255.255.0
localityName: aca.warsaw
```

The name of a subnet is structured in this way:

1. the identifier of the name: **subnet**
2. the **name of the ACA** that is responsible for the subnet
3. the **fixed part of the IP address** of this subnet

The *localityName* entry contains the name of the ACA that is responsible for this subnet.

8.1.1.6.10 Information to the Subscriber entry (specific for each test bed):

Subscriber information has to be stored in this way:

```
ldapadd -D"uid=xyz,ou=Directory Administrator,o=aquila,c=eu"
-f subscribTrialsite.ldif -c
```

An example content of the *suscriberTrialsite.ldif* file would be:


```
dn: uid=user1, cn=Subscriber,o=AQUILA,c=EU
objectclass: remoteUser
uid: user1
userPassword: pass1
authLoginService: pcbv
authLoginService: pvbr
authLoginService: pmm
```

```
dn: uid=user2, cn=Subscriber,o=AQUILA,c=EU
objectclass: remoteUser
uid: user2
userPassword: pass2
authLoginService: pcbv
authLoginService: pvbr
authLoginService: pmm
authLoginService: pmc
```

```
dn: uid=user3, cn=Subscriber,o=AQUILA,c=EU
objectclass: remoteUser
uid: user3
userPassword: pass3
authLoginService: pcbv
```

```
dn: uid=user4, cn=Subscriber,o=AQUILA,c=EU
objectclass: remoteUser
uid: user4
userPassword: pass4
authLoginService: pmm
authLoginService: pmc
```

8.1.1.7 Trace Server

The Trace Server does not need any specific installation procedure. It requires the same general steps as the other RCL modules (RCA, ACA, etc). After the compilation, the command line that starts the server is the following:

```
java -Duser.dir=.
      -Daquila.rcl.propertyFile =$support/rc/traceServer.rc
      aquila.rcl.util.TraceMain
```

The trace server property file includes the information:

- on which host / port the CORBA Naming Service (TNameServ) is running
- which LDAP factory is used
- the URL to the root entry of the LDAP Directory Server
- the logging directory
- the info if the ldapNotify feature has been activated

As an example, the traceServer.rc file would be:

```
org.omg.CORBA.ORBInitialHost:    sun1
org.omg.CORBA.ORBInitialPort:    1234
```

```
java.naming.factory.initial:    com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:      ldap://sun1:389/o=AQUILA,c=EU
aquila.rcl.trace.dir:          <home>/log
aquila.rcl.ldapNotify:         true
```

8.1.1.8 QMTool

8.1.1.8.1 Compilation

First of all, all the java files of the application contained in qmtool-yy.mm.dd.zip, except for the files RShareInfo.java and TrafficSpec.java, should be placed under a directory called “*Tool*” since they belong to a package named “*Tool*”. The files RShareInfo.java and TrafficSpec.java should be placed under the directories “*aquila\rcl\network*” and “*aquila\rcl\tc*” respectively. It would be more convenient if the directory “*Tool*” resided in parallel with the directory “*aquila*” so that only one CLASSPATH needs to be set.

Sequentially, the environment variable PATH should be updated by executing the following from the command line:

```
set path=c:\jdk1.3\bin;%path%
```

This variable should be set to the corresponding path of the JDK1.3 installation. Note that this application has been tested only with jdk1.3 under Windows NT Workstation 4.0.

Moreover, the environment variable CLASSPATH should be set to the directory *classes*, where the classes are stored.

```
set classpath=<aquila home dir>\classes;%classpath%
```

Additionally, the jar file “*jbcl3.1.jar*” should be added at the CLASSPATH. This jar file can be found at the “*lib*” directory of JBuilder.

```
set classpath=c:\jbuilder35\lib\jbcl3.1.jar;%classpath%
```

After the above settings, everything is prepared for the compilation of the java files. The compilation should be done with the following order. Initially, the compilation of the TrafficSpec.java file should take place. From the *source* directory, type:

```
javac -d ..\classes aquila\rcl\tc\TrafficSpec.java
```

Then, the compilation of the RShareInfo.java file should follow.

```
javac -d ..\classes aquila\rcl\network\RShareInfo.java
```

Finally, the compilation of the rest of the java files, which are stored under the “*Tool*” directory:

```
javac -d ..\classes\Tool\*.java
```

8.1.1.8.2 *Running the QMTool*

After the compilation is complete, the application could be run from any location. The file QMTool.java is the one that contains the method “main”.

```
java Tool.QMTool
```

After the application has started running, the first thing to be done is the establishment of a connection to the LDAP database for the storage of the Resource Pools to be created. This is achieved through the menu “Settings...” which is under the menu “Database”. The selection of this menu will cause a dialog box to appear. The fields “Host name” and “Port number” should be completed with the name of the host that the LDAP server is running at and the port number that it listens to (by default 389). At the field “Initial context”, the distinguished entry of the database for the storage of the Resource Pools should be placed, for example “cn=Pool,cn=Aquila,o=EU”. This entry should have already been created otherwise an error message will appear. In any case, if any of these three settings is not correct, the further use of QMTool is not attainable.

After this step the Resource Pool tree should be drawn. This is feasible with the help of the buttons (ResourcePool, Aca, Line) that reside at the toolbar. It should be noted that during this phase, nothing is stored into the database. For this to happen, we have to move on to the configuration of the Resource Pools and the ACAs. By right clicking on a Resource Pool or an ACA and by selecting “Properties...” from the popup menu, a dialog box appears. The two fields “poolID” and “machineID” of the dialog box should be filled in with the ID of the pool and the machine it is running on respectively, for example “pool1” and “rca”. After this information is saved, the pool will be created at the database.

The creation of the Resource Shares of each Pool can be realised in the same way, by selecting “RShare” from the popup menu and by filling in the fields of the dialog box that appears. The calculation of those values depends on the topology of the testbed as well as the distribution of resources among the Network Services. A detailed description of the way those parameters are calculated can be found at [D1301]. The results can then be entered with the QMTool.

The creation of the Resource Shares is allowed only after the creation of the corresponding Resource Pool since they are stored at the LDAP database under its entry. The selection of the Resource Share to be created can be realized from a drop-down list that contains all the Resource Shares of a Pool.

When the Resource Pool tree is complete, or even at the start of its creation, the “root” of the tree should be defined by right clicking on a Resource Pool and selecting “MakeRoot” from the popup menu. At this point every Resource Pool is configured correctly and stored at the database.

8.1.1.9 Operation of the Resource Control Layer

After the Java sources have been installed and compiled, the Java classes must be in the *classes* directory. In order to start the Resource Control Layer the servers and the RCL components should be started in the following way:

1. LDAP Server
2. Name Server
3. Trace Server
4. RCA
5. ACA(s)
6. Proxy(-ies)
7. EAT(s)
8. Tomcat (Reservation GUI)

Information on installation (content of property files, other settings) and operation pertaining to a specific component is given in subsequent paragraphs.

8.1.2 RCA

The RCA component consists of only one component, the RCA package. However, it depends on the packages Network, TC, Util.

8.1.2.1 Required packages and property files

The following files and software packages are needed for a successful installation of the RCA:

1. The IDL file, that defines the interfaces to the RCA component:
rca-xx.yy.zz.idl
2. The RCA software package, which implements the RCA components:
rca-xx.yy.zz.zip
3. The property file for the RCA component. The name of the property file may be *rcaXyz.rc*, where Xyz is a unique name for the RCA identification (e.g. rcaAthens.rc)

The property file includes the information:

- on which host / port the CORBA Naming Service (TnameServ) is running

- which LDAP factory is used
- the URL to the root entry of the LDAP Directory Server
- the RCA name
- the logging directory
- the info if the ldapNotify feature has been activated

As an example, the rcaAthens.rc file would be:

```
org.omg.CORBA.ORBInitialHost:    sun1
org.omg.CORBA.ORBInitialPort:    1234
java.naming.factory.initial:     com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:        ldap://sun1:389/o=AQUILA,c=EU
aquila.rcl.rca.name:             rca.athens
aquila.rcl.trace.dir:            <home>/log
aquila.rcl.ldapNotify:           true
```

8.1.2.2 Operation

The RCA module is stated with the following command:

```
java -classpath classes/external/jta20.jar
     -Duser.dir=.
     -Daquila.rcl.propertyFile=$HOME/support/rc/rcaXyz.rc
     aquila.rcl.rca.Rca
```

where rcaXyz.rc is the corresponding property file for the RCA.

8.1.3 ACA

The ACA component, a part of the Resource Control Layer (RCL), only consists of one component, the ACA package. But the ACA component is dependent on the packages RCA, Network, Service, Subscriber, TC, and Util and uses various interfaces of other software packages e.g. to the EAT package. For that purpose different interface description files and other software classes of the RCL are necessary for a successful installation.

The installation of the ACA, the needed components and the start up procedure is described in the following.

8.1.3.1 Required packages and property files

The following files and software packages are needed for a successful installation of the ACA itself:

1. The IDL files that describe the interfaces to the ACA component:

```
acaInternal-xx.yy.zz.idl
```

aca-xx.yy.zz.idl

2. The ACA software package, which implements the ACA components:

aca-xx.yy.zz.zip

3. The external software components, that is needed to set up a telnet connection to a router device:

jta20.jar

Store this package in the *external* directory.

4. The property file for the ACA component. aquila.rc is the default name or something like acaXyz.rc ca also be used.

The property file includes the information:

- on which host / port the CORBA Naming Service (TNameServ) is running
- which LDAP factory is used
- the URL to the root entry of the LDAP Directory Server
- the ACA name
- the RCA name
- the logging directory
- the info if the ldapNotify feature has been activated

As an example, the acaWarsaw.rc file would be:

```
org.omg.CORBA.ORBInitialHost:    sun1
org.omg.CORBA.ORBInitialPort:    1234
java.naming.factory.initial:     com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url:        ldap://sun1:389/o=AQUILA,c=EU
aquila.rcl.aca.name:             aca.warsaw
aquila.rcl.rca.name:             rca.athens
aquila.rcl.trace.dir:            <home>/log
aquila.rcl.ldapNotify:           true
```

8.1.3.2 Operation

The ACA module is stated with the following command:

```
java -classpath classes/external/jta20.jar
     -Duser.dir=.
     -Daquila.rcl.propertyFile=$HOME/support/rc/acaXyz.rc
     aquila.rcl.aca.Aca
```

where acaXyz.rc is the corresponding property file for the specific ACA.

8.1.3.3 EDAdapter

The EDAdapter is the part of the ACA that connects to the routers and configures them by using the Command Line Interface (CLI). For the first trial, one ACA controls exactly one edge router. The EDAdapter is configured with two property files. Every edge device within the trial network needs those two property files. The files have to be adapted to each device.

The property files are LDIF files (LDAP Data Interchange Format) for the LDAP database. The ACA and the EDAdapter get the information from the LDAP server.

8.1.3.3.1 ACA file

Each ACA is connected to one router. In the following example:

```
dn: cn=aca.helsinki,cn=AdmissionCtrlAgent,o=AQUILA,c=EU
objectclass: device
cn: aca.helsinki
description: routerFactoryClass=aquila.rcl.aca.TelnetRouterFactory
description: routerType= VXR2ios1201
description: routerHost= 194.240.200.135
description: routerName= vxr2
description: routerAuthentication=secure:null/aquila
```

The name of the ACA is *aca.helsinki*. With this name the properties can be found in the LDAP database. The name of the ACA also stands in the third line behind “cn:”.

The ACA configures the router with the CLI over a Telnet connection. Another possibility is to use COPS. Therefore we have 2 different routerFactory Classes. For the first trial we only use the TelnetRouterFactory. The prefix *aquila.rcl.aca* reflects the structure of the subdirectories.

The entry *routerType* leads to the second LDAP object. This object includes router specific information. The structure of this object is described in the following paragraph. The entry *VXR2ios1201* is the name of the object. There is no special rule to generate the name. In this example it is a combination of the *routerName* and the IOS version.

The IP number of the router is stored at *routerHost*. The IP number is used to start the Telnet connection. One can also use a symbolic name.

The *routerName* is the name of the router. The name of the router is visible in the prompt of the router:

```
vxr2>
vxr2#configure terminal
vxr2(config)#
```

The EDAdapter reads the prompt of the router.

The *routerAuthentication* string is a combination of passwords. The first string (secure in the example) is the password to “enable” the router. The second string behind the “:” is the Username to log on to the router. For some routers no Username is set. Therefore you can use “null”. The third string behind the “/” is the password for the router (in the example: aquila).

8.1.3.3.2 Router file

The second property file is also an LDIF file. This file stores more router specific commands as well as the configuration of the scheduler. See the example for vxr2ios1201:

```
dn: cn=VXR2ios1201,cn=telnet,cn=Router,o=AQUILA,c=EU
objectclass: device
objectclass: javaObject
cn: VXR2ios1201
javaclassname: aquila.rcl.aca.Cisco7206
description: number_of_commands=48
description: cmd1=configure terminal
description: prompt1=(config)#
description: cmd2=class-map TCL1
description: prompt2=(config-cmap)#
description: cmd3=match ip dscp 48
description: prompt3=(config-cmap)#
description: cmd4=exit
description: prompt4=(config)#
description: cmd5=class-map TCL2
description: prompt5=(config-cmap)#
description: cmd6=match ip dscp 40
description: prompt6=(config-cmap)#
description: cmd7=exit
description: prompt7=(config)#
description: cmd8=class-map TCL3
description: prompt8=(config-cmap)#
description: cmd9=match ip dscp 32 24
description: prompt9=(config-cmap)#
description: cmd10=exit
description: prompt10=(config)#
description: cmd11=class-map TCL4
description: prompt11=(config-cmap)#
description: cmd12=match ip dscp 16 8
description: prompt12=(config-cmap)#
description: cmd13=exit
description: prompt13=(config)#
description: cmd14=policy-map aquila1
description: prompt14=(config-pmap)#
description: cmd15=class TCL1
description: prompt15=(config-pmap-c)#
description: cmd16=priority percent 10
description: prompt16=(config-pmap-c)#
description: cmd17=exit
description: prompt17=(config-pmap)#
description: cmd18=class TCL2
description: prompt18=(config-pmap-c)#
description: cmd19=bandwidth percent 15
description: prompt19=(config-pmap-c)#
description: cmd20=queue-limit 100
description: prompt20=(config-pmap-c)#
description: cmd21=exit
description: prompt21=(config-pmap)#
description: cmd22=class TCL3
description: prompt22=(config-pmap-c)#
description: cmd23=bandwidth percent 30
description: prompt23=(config-pmap-c)#
description: cmd24=queue-limit 100
description: prompt24=(config-pmap-c)#
description: cmd25=random-detect
description: prompt25=(config-pmap-c)#
```



```
description: cmd26=random-detect precedence 4 30 40 10
description: prompt26=(config-pmap-c)#
description: cmd27=random-detect precedence 3 10 20 10
description: prompt27=(config-pmap-c)#
description: cmd28=exit
description: prompt28=(config-pmap)#
description: cmd29=class TCL4
description: prompt29=(config-pmap-c)#
description: cmd30=bandwidth percent 5
description: prompt30=(config-pmap-c)#
description: cmd31=queue-limit 100
description: prompt31=(config-pmap-c)#
description: cmd32=random-detect
description: prompt32=(config-pmap-c)#
description: cmd33=random-detect precedence 2 25 35 10
description: prompt33=(config-pmap-c)#
description: cmd34=random-detect precedence 1 10 20 10
description: prompt34=(config-pmap-c)#
description: cmd35=exit
description: prompt35=(config-pmap)#
description: cmd36=class class-default
description: prompt36=(config-pmap-c)#
description: cmd37=
description: prompt37=(config-pmap-c)#
description: cmd38=random-detect
description: prompt38=(config-pmap-c)#
description: cmd39=exit
description: prompt39=(config-pmap)#
description: cmd40=exit
description: prompt40=(config)#
description: cmd41=interface Serial4/0
description: prompt41=(config-if)#
description: cmd42=max-reserved-bandwidth 100
description: prompt42=(config-if)#
description: cmd43=service-policy output aquila1
description: prompt43=(config-if)#
description: cmd44=exit
description: prompt44=(config)#
description: cmd45=interface POS3/0
description: prompt45=(config-if)#
description: cmd46=max-reserved-bandwidth 100
description: prompt46=(config-if)#
description: cmd47=service-policy output aquila1
description: prompt47=(config-if)#
description: cmd48=exit
description: prompt48=(config)#
```

The name of this LDAP object is stored in the first and the fourth line (in this example: vxr2ios1201). In the LDAP database, you can find this object under the entries *telnet* and *router*.

The *javaclassname* is the name of the Java class used for the configuration of the router. For each router type (Cisco 7507, 2620, 1650, 3640, etc.) a specific class exists. The names of those classes are build like this: Cisco7206 or Cisco7507. Also, the prefix *aquila.rcl.aca* reflects the structure of the subdirectories.

The *number_of_commands* parameter gives the number of commands used for the configuration of the scheduler. Be careful to **update** it if you include some additional commands.

In the example above all the commands are presented. In command 42 and 45 we combine the service-policy to the interface. In some router types it is not possible to use the command:

```
vxr2(config-cmap)# match ip dscp 40
```

therefore we have to use access-lists. Those access-lists are generated in the Java class for the router. The access-lists have the numbers from 100 to 103.

8.1.3.3.3 Search for failures

To check, whether the configuration was successful or not, you have to look into the log-files of the ACA. A different log file exists for every ACA.

8.1.3.3.3.1 Failure during start up

After the successful start-up there are two entries in the log-file from the EDAdapter:

```
2001-02-16 17:47:49.786 aca.warsaw INFO aquila.rcl.aca.Cisco1605_1.init:
accessListNumbers
2001-02-16 17:47:57.059 aca.warsaw INFO aquila.rcl.aca.Cisco1605_1.init:
initConfiguration
```

The first entry is logged after the successful connection to the router and the login on the router. If this command does not exist, you have to search in the LDIF files for errors. Maybe a password, or the IP address of the router or the name of the router is wrong. These entries belong to the LDIF file for the ACA.

The second entry indicates the successful configuration of the scheduler. If a failure occurs in this part, one of the entries in the LDIF file for the router may be wrong. The EDAdapter stops to work with a wrong prompt entry. The EDAdapter is waiting for the prompt string, which the router will not send. There is no mechanism to handle this failure. You can notice this kind of failure, when the RCL stops to work.

8.1.3.3.3.2 Failure during runtime

The EDAdapter also logs the successful configuration of the classifier, marker and policer on the router. See the example below:

```
2001-02-16 13:18:43.549 aca.warsaw INFO
aquila.rcl.aca.Cisco1605_1.setExtendedClassifierList: access-list 101 per-
mit ip 10.0.5.2 0.0.0.0 10.0.10 0.2 0.0.0.0 precedence 0
2001-02-16 13:18:45.249 aca.warsaw INFO
aquila.rcl.aca.Cisco1605_1.setRateLimit: rate-limit input access-group 101
53000 2005 2005 conform-action set-prec-transmit 6 exceed-action drop
```

The first entry is for the classifier and the second is for the policer and marker. To check the successful configuration you just have to control the current configuration of the router.

The EDAdapter also logs the release of a flow. The entries for this action are:

```
2001-02-16 13:20:11.192 aca.warsaw INFO
aquila.rcl.aca.Cisco1605_1.removeClassifierList: no access-list 100
2001-02-16 13:20:12.749 aca.warsaw INFO
aquila.rcl.aca.Cisco1605_1.removeRateLimit: no rate-limit input access-
```

```
group 100 50000 2005 2005 conform-action set-prec-transmit 6 exceed-action drop
```

If an entered parameter exceeds the usable range for the router, the EDAdapter refuses the request. The EDAdapter logs the refusing of the request with:

```
2001-02-16 17:49:21.137 aca.warsaw ERROR
aquila.rcl.aca.Cisco1605_1.setRateLimitProfil: invalid SR
```

The range of the parameters is:

Traffic Class	Ranges described in [D1301]	Actual ranges in Cisco routers	Actual ranges in Unisphere routers
Premium CBR	(Page 57) Traffic model: single rate	PR: 8000 - 2000000000 bits/sec BSP: 2000 - 512000000 bytes	PR: 0 - 4294967295 bits/sec BSP: 0 - 4294967295 bytes
	PR: max = 200 Kb/s m: min = 40B, max = 256B M: n.a BSP: n.a.		
Premium VBR	(Page 58) Traffic model: dual token bucket	PR: 8000 - 2000000000 bits/sec BSP: 2000 - 512000000 bytes SR: 8000 - 2000000000 bits/sec BSS: 2000 - 512000000 bytes	PR: 0 - 4294967295 bits/sec BSP: 0 - 4294967295 bytes SR: 0 - 4294967295 bits/sec BSS: 0 - 4294967295 bytes
	PR: max = 1 Mb/s SR: max = PR BSS: min = M m: min = 40B, max = M M: n.a BSP: n.a.		
Premium Multimedia	(Page 58) Traffic model: single token bucket	SR: 8000 - 2000000000 bits/sec BSS: 2000 - 512000000 bytes	SR: 0 - 4294967295 bits/sec BSS: 0 - 4294967295 bytes
	SR: max = 250Kb/s BSS: min = M m: min = 40 B, max = M M: n.a		
Premium Mission	(Page 59) Traffic model: dual token bucket	PR: 8000 - 2000000000 bits/sec BSP: 2000 - 512000000 bytes	PR: 0 - 4294967295 bits/sec BSP: 0 - 4294967295 bytes

Critical	SR: 8000 - 2000000000 bits/sec	SR: 0 - 4294967295 bits/sec
	PR: max = 50 Kb/s	
	SR: max = 5Kb/s (For the integration in Warsaw 16 Kb/s was used)	BSS: 0 - 4294967295 bytes
	BSS: min = 10.000 B	
	m: min = 40 B, max = M	
	M: n.a	
	BSP: n.a.	

Table 19: Range of parameters for different traffic classes

The EDAdapter releases the configured classifier after the occurred error. The event is logged by the EDAdapter.

Sometimes the telnet connection between the EDAdapter and the router breaks down. The EDAdapter reconnects the router again and logs the event:

```
2001-02-16 13:18:45.249 aca.warsaw INFO
aquila.rcl.aca.Cisco1605_1.setRateLimit: Reconnect
```

In some case, the EDAdapter can't reconnect to the router. The RCL does not proceed in this failure case. In this case there is a serious problem in the network between the ACA and the router.

8.1.3.3.3 Used access lists

The new version of the EDAdapter will remove all CAR, which are combined to an access-list. The EDAdapter should have control over all CAR. Otherwise no guaranties can be given for each flow. This is done during the start-up of the EDAdapter. When the RCL is started, some CAR configurations of the router may be lost.

8.1.3.4 System requirements

Note that for correct debugging information, the time of all involved SUN platforms has to be synchronized. For that purpose the NTP (Network Time Protocol) process should be run on each SUN machine. One of the machines should be configured as an ntp-time-server that sets up the reference time for all ntp-clients.

8.1.4 EAT

The EAT for the first trial consists of the following components:

- The EAT Manager
- The "internal" API

- The EAT Script
- The GUIs
- The Proxy

Their installation and operation is described in the following sections

8.1.4.1 EAT Manager, API and EAT Script

8.1.4.1.1 Required Installation files

For the EAT itself, the following ZIP files are needed:

- eat_idls-yy.mm.dd.zip, containing the actual IDL files of the EAT:
 - api.idl
 - converter.idl
 - eat.idl
 - eatManager.idl
 - proxy.idl
- eat-yy.mm.dd.zip, containing the actual JAVA files of the following packages:
 - aquila/rcl/eat
 - aquila/rcl/eat/api
 - aquila/rcl/eat/gui (Only the GUI.java dummy implementation)
 - aquila/rcl/eat/eatManager
 - aquila/rcl/eat/script

For the EAT Script, additional files are needed:

- EATScript-YY.MM.DD.zip, containing the actual DTD and XML files/examples
- jaxp-1_0_1.zip, Sun's Java API for XML Parsing

8.1.4.1.2 Installation procedure

In order to create the EAT package structure the procedures listed in paragraph 8.1.1 should be followed. Additionally, the following steps are required:

- Create manually the additional subdirectories: *aquila/rcl/eat/gui* and *aquila/rcl/eat/script*
- Unpack eat-yy.mm.dd.zip into the *aquila/rcl/eat* directory. (Since the ZIP file contains the appropriate, relative paths, the JAVA files are extracted in the right packages.)
- Make sure that also the following packages are installed:
 - aquila.rcl.aca

```
aquila.rcl.eat.proxy
aquila.rcl.service
aquila.rcl.subscriber
aquila.rcl.tc
aquila.rcl.util
```

- Compile the EAT, and the dummy GUI:

```
javac aquila/rcl/eat/eatManager/EAT.java
javac aquila/rcl/eat/gui/GUI.java
```
- Before you compile the EAT Script, you have to unzip jaxp-1_0_1.zip first (into a directory of your choice, here: *JAXP_HOME*), and to modify your *CLASSPATH*:

```
setenv CLASSPATH ${CLASSPATH} : $JAXP_HOME/jaxp.jar :
    $JAXP_HOME / parser.jar
javac aquila/rcl/eat/script/EATScript.java
```

8.1.4.1.3 How to run

The EAT Manager, the GUI, and the EAT Script make use of the property file, such as *aquila.rc*. Add the following *sample* entries:

```
org.omg.CORBA.ORBInitialHost: <hostname>
org.omg.CORBA.ORBInitialPort: <port>
java.naming.factory.initial: com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url: ldap://<hostname>:389/o=AQUILA,c=EU
aquila.rcl.aca.name: aca.Warsaw
aquila.rcl.eat.eatManager.name: eat.Warsaw
aquila.rcl.eat.proxy.name: proxy.Warsaw
aquila.rcl.eat.proxy.1: NetMeeting
aquila.rcl.eat.proxy.1.name: NetMeeting 3.0 Proxy
aquila.rcl.eat.proxy.1.port: 9988
aquila.rcl.trace.dir: trace
aquila.rcl.ldapNotify: false
```

The EAT should be started **after** the corresponding ACA (here: *aca.Warsaw*) and **after** the corresponding Proxy (here: *proxy.Warsaw*).

Start the EAT Manager:

```
java aquila.rcl.eat.eatManager.EAT
```

Start the Dummy GUI and see the messages:

```
java aquila.rcl.eat.gui.GUI
```

Or start the EAT Script:

```
java aquila.rcl.eat.script.EATScript Example.xml
```

8.1.4.2 Reservation GUI and Tomcat

8.1.4.2.1 Installation files

For the installation of the Tomcat web server and the Reservation GUI, the latest versions of the following files are required:

- The TOMCAT installation (tomcat.zip)
- The GUI beans (guis-yy.mm.dd.zip)
- The JSP files (jsps-yy.mm.dd.zip)

8.1.4.2.2 *Installation procedure*

The tomcat.zip file should be unzipped to a suitable folder. Afterwards, some environment variables should be set: Right click “My Computer” and select Properties. Go to “advanced” and press “Environment Variables”. Add the following **system** variables:

- JAVA_HOME (JDK 1.3 home directory)
- TOMCAT_HOME (tomcat home directory)
- CLASSPATH (the root of the AQUILA structure: *<AQUILA home>/classes*)

Afterwards, create (if not existing) a directory “gui” in the AQUILA structure: aquila.rcl.eat.gui and put the java GUI beans inside. Also, create a directory “jsp” in the AQUILA structure: aquila.rcl.eat.jsp and put the JSP files inside. Modify the server.xml file in the `\tomcat\conf\` folder. The following lines should be added (near the end of the file, after the other “context path” assignments):

```
<Context path="/aquila"
    docBase="c://aquila-installation-dir//aquila//rcl//eat"
    debug="0" reloadable="true" >
</Context>
```

This will create a new context *Aquila* that points to the docBase:
`c:\mydirectory\Aquila\rcl\eat.`

A **new** aquila.rc configuration file should be placed in `\tomcat\bin` directory, with values like the following:

```
org.omg.CORBA.ORBInitialHost: sun1
org.omg.CORBA.ORBInitialPort: 1234
java.naming.factory.initial: com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url: ldap://sun1:389/o=aquila,c=EU,o=telecom.ntua.gr
aquila.rcl.eat.eatManager.name: eat.warsaw
```

Then, using the Command Prompt, go to `\tomcat\bin` directory and run: **tomcat run**

In order to load a .jsp file from a Web Browser, the following syntax should be used:

```
http://PC-with-tomcat:8080/context/pathToFiles
```

which in our case translates in this:

```
http://PC-with-tomcat:8080/aquila/jsp/LoginGUI.jsp.
```

To use the Reservation GUI, the LoginGUI.jsp should first be loaded and the user should login to the network. After successful reservation, the ReservationGUI.jsp page can be loaded, in order to make a reservation.

In the general case where a reservation for a flow other than NetMeeting is needed, **no** proxy should be selected. The following parameters should be entered:

- Name of the application (a string) and the kind of data for this flow (audio, video, data)
- Source and destination addresses, network masks, transport protocol and ports. If one port is needed, the lower port filed should only be used. If a range of ports is specified, you should use both lower and upper port.
- Traffic parameters

The DSCP and the duration parameters are not needed for the first trial.

In the case of the Proxy, you should select the name of the Proxy (only NetMeeting available for the first trial). In this case, *you don't need to enter the Protocol ID. Although ports are not needed you should fill in the lower port number with some random numbers, or else an error will appear.*

8.1.4.2.3 Important notes

- The GUI beans make use of some other classes in the AQUILA packages. It is therefore essential to have those classes installed in the PC. It would be a good idea to have the full AQUILA packages installed in the system.
- Make sure that you don't use a page that is cached in the browser, since it may cause problems. Remember to use "refresh" when trying to access an already accessed page.
- When such problems arise, it may be better to restart tomcat.

8.1.4.3 Proxy

The proxy module to be used for the first trial is a limited version of the proxy described in D2201. It is specifically designed to support the operation of NetMeeting for the first trial. The Proxy module uses a port of the Berkeley Packet Filter (BPF) for Windows NT. Therefore is portable to other platforms, like Solaris and Linux. However, for the purposes of the first trial, the Proxy will run on WinNT or Win2k.

8.1.4.3.1 Installation files

For the installation of the proxy, three .zip files are provided:

- The Windows Packet Capture Library from Torino Polytechnic (Wpdpack.zip). It includes the WinPcap NDIS driver (for WinNT and Win2k) and the appropriate libraries. It can also be downloaded from <http://netgroup-serv.polito.it/winpcap>

- The Java files of the Proxy module. They are included in the `aquila.rcl.eat.proxy` package (`proxy-yy.mm.dd.zip`)
- The Java Native Interface (JNI) between the Java classes and the WinPcap library (written in C). A Windows Dynamic Link Library (DLL) has been created for the java files to access the WinPcap driver (`Filter-yy.m.dd.zip`).

8.1.4.3.2 Installation procedure

First the driver and its libraries should be installed. Instructions are very simple and can be found at the WinPcap site at: <http://netgroup-serv.polito.it/winpcap/>. In brief, from Control Panel's Network settings a new Protocol should be installed. The location of the driver (`<install-dir>\wpdpack\drivers\packetNT`) should be provided. If the installation is successful, a new entry will appear at the "Services" tab and **not** at the "Protocol" tab, from where the installation started. It is quite strange, but it should not alarm you.

The Java files should be placed at the aquila hierarchy: `aquila.rcl.eat.proxy`

The C project (made with Visual C++ 6.0) may be placed at any folder. Project Settings will certainly need changes as far as the *include* directories are concerned. The "include directory" should be changed to the directory where the include files from the `wpdpack.zip` archive were unzipped. Also, the static library `libpcap.lib` should be placed in the directory of the project (it already exists there). The compilation and linking should then be successful. Note that the output DLL (`filter.dll`) will directly appear at the `c:\winnt\system32` folder (the natural place for DLLs).

An **already compiled DLL** is included in `Filter.zip` (`filter.dll`). This DLL should be placed either in `winnt\system32`, or in the starting directory from where the proxy will be run (`src`).

Finally, the `aquila.rc` should contain the following lines:

```
aquila.rcl.eat.proxy.name: proxy name
```

This is the name, the EAT Manager will search for in the Naming Service during start up.

8.1.4.3.3 Operation

For the instantiation of the Proxy module, the usual procedures of running an RCL module should be followed: from the root of the AQUILA structure, the following command should be given:

```
java aquila.rcl.eat.proxy.Proxy
```

The host and the port of the Name Server are already specified in the `aquila.rc` file.

Important Note: The Proxy should be already running on the client's PC, before the EAT Manager starts up. That's because the EAT Manager tries to locate the Proxy in the Name Service.

When there is a reservation request from Tomcat for a NetMeeting flow, the EAT Manager contacts the Proxy and requests the NetMeeting port numbers. When the Proxy detects them, it asynchronously notifies the EAT Manager.

In the event that the Proxy crashes (there seems to be a bug in the implementation of the Java Native Interface of JDK 1.3), the Proxy **as well as the EAT** should be restarted.

8.1.4.3.4 Topology for the first trial

In essence, the Proxy acts as a packet filter that sniffs all the packets traversing the LAN, and by analysing them it tries to discover new flows that have multimedia content specific to NetMeeting. For the filter to work properly, it must run on a PC that takes part in this LAN, which is made up:

- Either of a coaxial cable
- Or, a passive hub

It is evident that when **Ethernet switches** are used, the filter will only be able to see packet that are destined to it or are sent by it. For the first trial, the Proxy can run on the same PC as NetMeeting, or in another PC that is connected to the same hub with the PC running Net-Meeting.

8.1.5 Measurement tools

Before the installation of the measurement tools, the following software has to be installed and running at the measurement server. Version numbers are recommended, but not restricted to:

- SuSE Linux 6.4
- MySQL Database 3.22.23, myODBC 2.50.23, unixODBC 1.8.12
- Apache 1.3.14 with modules mysql_auth 2.20 and php4.0.4p11 (with gd-support)
- Gnuplot 3.7
- Xntp 4.0.99

8.1.5.1 Creation of the AQUILA database and loading some initial data

For the installation of the AQUILA database, two scripts are needed. Unpack and load them into the database by executing

```
mysql < db-aquila-model.sql
mysql < db-aquila-initdata.sql
```

The scripts contain the database model, information about access rights to the database and initial user, traffic and reservation data.

8.1.5.2 Update of web server configuration

The configuration file from the apache web server (usually `httpd.conf`) has to be updated. The following lines have to be included (path information and access directives can be adjusted):

```
Auth_MySQL_Info localhost apache geheim

<Directory "/usr/local/apache/htdocs/aquiladma">
    Options All MultiViews
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

The server port, which defaults to port 80, should be changed to port 8089 (around line 290) by changing `httpd.conf`:

```
Port 8089
```

To apply changes in the configuration file, the web server has to be restarted.

8.1.5.3 Installation of the GUI to the measurement tools

The php-files for the web based graphical user interface have to be installed to the `htdocs` directory of the Apache web server. The files are located in `aquiladmYYYYMMDD.tar.gz`.

```
cd /usr/local/apache
tar -xzf aquiladmaYYYYMMDD.tar.gz
```

the files will be extracted to directory `htdocs/aquiladma/` and below.

To test the GUI, start browser with `http://hostname:8089/aquiladma/`. If the AQUILA database is up, you should get an identification request, which can be answered with `aquila/aquila`

8.1.5.4 Installation of the DTA masterstation software module

The following procedure should be used:

- Copy the masterstation software module “master” and the configuration file “master.rc” to the destination directory (`/usr/local/bin`).
- Edit the configuration file `master.rc`, set the correct values for `DBHOST`, `DBNAME`, `DBUSER` and `DBPASS`

```
DBHOST = name or IP-address of the MySQL-Server
DBNAME = name of the AQUILA database ("aquila")
DBUSER = name of the MySQL-user ("aquila")
DBPASS = password ("aquila" )
```

- Run the masterstation module, a menu will be displayed.

- Switch to screen “Show status” (key 5). The display shows something like “Connected to ... and Master connected to DB”, the masterstation is ready to start measurement flows.

8.1.5.5 Installation of the SPU CMCaller software module

Before the CMCaller can be started, unixODBC has to be configured properly. This can be done by either copying the files `odbc.ini` and `odbcinst.ini` into the directory `/usr/local/etc` or by configuring the data source via

```
/usr/local/bin/ODBCConfig
```

This is only possible, if the GUI has been compiled into the unixODBC-distribution. But only, if the qt-and X-libraries are installed this is default.

With the text-based tool

```
/usr/local/bin/odbcinst
```

configuration is also possible.

Now, the CMCaller can be configured and compiled

```
cd /usr/local
tar xvzf /path/to/cmcaller.tar.gz
cd cmcaller
./configure
make
```

The CMCaller can be started with

```
cd /usr/local/cmcaller/cmcaller
./cmcaller
```

If the `cmcaller` should automatically start at boot time, copy the startscript (`cmcaller`) to the `init` directory.

```
cp cmcaller /path/to/initscripts/ (e.g. /etc/rc.d/)
ln -s /path/to/initscripts/cmcaller
    /path/to/startrunlevel/S99cmcaller
ln -s /path/to/initscripts/cmcaller
    /path/to/startrunlevel/K10cmcaller
```

After that the CMCaller can be started and stopped by executing.

```
/etc/rc.d/cmcaller { start | stop }
```

8.1.5.6 Installation of the DTA measurement agents

The following description shows the installation procedure for a measurement agent on a PC.

8.1.5.6.1 Hardware requirements

Minimum hardware configuration:

- CPU: Pentium > 400MHz
- RAM >128MByte

- IDE hard disk controller (SCSI in not supported yet)
- Network interface:
 - Intel EtherPro100 (8255x chipset)
 - 3COM 3c59x
(You can check the NIC after installing Linux; type the command “lsmod”, there must be “eepro100” or “3c59x” listed!)
- Meinberg GPS167PCI card

Note: Be sure that the APM (advanced power management) is switched **OFF** in the BIOS!

8.1.5.6.2 Software

The measurement agent was compiled under SuSE 6.4 Linux distribution with kernel 2.2.14.

Install “standard software” package. **Do not** create a SWAP partition and use only ext2 file system. The /boot directory must be under the 1023 cylinder limit, because of the LILO. Please refer the SuSE manual. If a swap space is already in use, the commands “swapoff” and “swapon” can be used to turn the swap on and off.

After installation is finished login as “root”. Copy the tar file “nettest.tgz” in the /root directory and unpack with “tar -xvzf nettest.tgz”. Change to the new directory “nettest” and run the installation script “./install”. Perhaps you have to change the file properties to executable (chmod u+x install).

What does the install script do:

- Copy meinberg.o to /lib/modules/2.2.14/net/
- Make node meinberg c 42 0 in /dev/
- Copy start script nettest_s to /sbin/init.d/nettest
- Make link /sbin/init.d/rc2.d/S90nettest to /sbin/init.d/nettest
- Make link /sbin/init.d/rc2.d/K90nettest to /sbin/init.d/nettest
- Make link /sbin/init.d/rc3.d/S90nettest to /sbin/init.d/nettest
- Make link /sbin/init.d/rc3.d/K90nettest to /sbin/init.d/nettest
- Copy program nettest to /root
- Copy program ndm to /root
- Check the NIC (Intel or 3COM)
- Copy the suitable network card driver and kernel

After all files are copied and links are made, you have to register the new kernel (“/boot/vmlinuz.e100” or “/boot/vmlinuz.3c”) in the LILO. Therefore you have to edit the file “/etc/lilo.conf” and then run “/etc/lilo”. The new kernel must be the **first** in the file “/etc/lilo.conf”.

Make sure that the login is ASCII and **not** Graphical. You can change this in YAST.

8.1.5.6.3 Start and test

After finishing the installation the PC **must** be rebooted.

Be sure the green LED on the Meinberg GPS card is on.

After rebooting one of the last lines before the login must be the message that nettest was started (but the green “done” message will not be printed):

```
Starting nettest
```

Change to /root directory and run the test program ./ndm. This program checks some functions of the measurement agent. Be sure that no “failed” message is listed!

If there are failures, check:

- `ps -A | grep nettest` is the program running? Min 4 entries must be listed
- if kernel test failed check LILO
- `lsmod` check if meinberg module is loaded with NIC

8.1.5.7 Installation of the SPU measurement daemons

8.1.5.7.1 Installing the Meinberg GPS modules

1. Compilation and installation of the time module (this step can be skipped, if DTA Measurement Agent is already installed):

```
cp meinbergtime.tar /root/meinbergtime
cd /root/meinbergtime
tar xvf meinbergtime.tar
make
mknod /dev/meinberg c 42 0
insmod /root/meinbergtime/meinberg.o
```

Check if the module is running by compiling the read_gps.c file with "gcc -o read_gps read_gps.c" and run "./read_gps".

2. Compilation and installation of the co-ordinates module:

```
cp meinbergcoord.tar.gz /root/meinbergcoord
cd /root/meinbergcoord
tar zxvf meinbergcoord.tar.gz
make
mknod /dev/mbg_cor c 42 0
insmod /root/meinbergtime/mbg_cor.o
```

Check if the module is running by compiling the coord.c file with "gcc -o coord coord.c" and run "./coord".

8.1.5.7.2 Installing the SPU Agent

Copy the file cmdaemon-0.1.tar.gz to the /usr/local directory. Unpack with "tar zxvf cmdaemon-0.1.tar.gz". Change to the cmdaemon-0.1/ directory and run "./configure" and "./make".

```
cp cmdaemon-0.1.tar.gz /usr/local/cmdaemon-0.1.tar.gz
tar zxvf cmdaemon-0.1.tar.gz
cd cmdaemon-0.1
./configure
make
```

Copy the cmdaemon startscript to /etc/rc.d and create a symbolic link in /etc/rc.d/rc2.d/ with "ln -s /etc/rc.d/cmdaemon /etc/rc.d/rc2.d/S99cmdaemon" for starting at boot time.

```
cp cmdaemon /etc/rc.d/cmdaemon
ln -s /etc/rc.d/cmdaemon /etc/rc.d/rc2.d/S99cmdaemon
ln -s /etc/rc.d/cmdaemon /etc/rc.d/rc2.d/K10cmdaemon
```

Start and Test: Start and stop cmdaemon use

```
/etc/rc.d/cmdaemon { start | stop }.
```

To check the GPS modules use "lsmod".

The logmessages for errors and events of the cmdaemon are in the /var/log/cmd.event and /var/log/cmd.error files, check with "tail -f /var/log/cmd.event" or "tail -f /var/log/cmd.error".

```
/etc/rc.d/cmdaemon start
lsmod
tail -f /var/log/cmd.event
tail -f /var/log/cmd.error
```

8.1.5.8 Installation of the ELI router QoS monitoring tool

8.1.5.8.1 Initial installation

The router QoS monitoring tool is distributed in routerstats.tar.gz. The tool should be installed at the master measurement station:

```
cd /usr/local
tar xzf routerstats.tar.gz
```

The monitoring tool has to know the interfaces of every router in the network. This is done by running the tool **rinit** for every router in the network:

```
cd /usr/local/routerstats
rinit <router IP address> <user name> <access password> <enable
password> <session handler>
```

Session handler has two options: telnetsession and u_telnetsession. This depends on the router model that is used. **rinit** creates a configuration file for the router, which is used by the actual monitoring tool.

8.1.5.8.2 Preparation of the parameter definition files

Parameter definition files contain definitions for each parameter type that is requested from the router. Parameter definition files are needed, because router output depends on the router model and software version it runs.

Parameter definition files have the filename <IP address>.def. The parameter definition files are in /usr/local/routerstats/parameters directory, and it has the following format:

```
<Parameter number> <Parameter group number> <line number> <word
number> <command>
```

Parameter number is a unique number that is used within the parameter files. Parameter group number should be the same for parameter values that are displayed with the same command. For example, in Cisco the output interface QoS statistics are shown with **show policy-map interfaces**. This command shows several parameters. All parameters have a unique number, and all those parameters should belong to the same parameter group.

Line number and word number specify the location of the parameter in the command output. The last entry in a line is the command that is sent to the router for getting the parameter value.

8.1.5.8.3 Preparation of the parameter files

The actual parameters that are fetched from the routers are specified in so-called parameter files. These files are in `/usr/local/routerstats/parameters` directory, and the file name is `<IP address>.param`. The parameter file contains several lines, which have the following format:

```
<Parameter number> <Access list number> <IP Address of the interface>
```

Parameter number is the number of the parameter that is needed from the router. Access list number defines the traffic class when core router parameters are queried. The access list numbers are needed only for parameter files that concern the core routers. Access list numbers are defined as follows: 106 for TCL1, 105 for TCL2, 104 for TCL3, 102 for TCL4 and 100 for Best Effort traffic. IP Address is the interface where the parameter is queried.

For parameters that aren't related to any specific access list, like CPU Utilisation, access list number can be anything, but it has to be specified. The parameter file must contain entries for each interface that is to be monitored.

8.1.5.8.4 Running the daemon

After the router database is initialised and the parameter files are defined, the daemon needs to be started. The daemon is started in the following way:

```
/usr/local/routerstats/launcher &
```

This starts the process to background, where it monitors the database for new flows, and starts router monitoring whenever a new flow starts.

8.2 Installation and use of applications

The installation of the applications for the first trial is covered in the following tables.

8.2.1 Siemens WinSIP

APPLICATION NAME	WinSip
Version	0.2.0
Operating system (incl. Service packs)	Windows NT, service pack 5,6

CPU	
Memory	
License required? No	Freeware? No
If yes: license price? Propriety of Siemens AG	If yes: for how many clients? Propriety of Siemens AG
Network requirements	
Protocol	UDP, (TCP)
Ports	Local port 5060 Server port 0
Access	
Comments	
Other required equipment	
Equipment 1	soundcard
Characteristic a	full duplex
Equipment 2	Headset
Characteristic a	
Other required software	
special installation procedure	
special configuration files	
servers to be set up	
possible troubleshooting	
Comments	<p>Automatic installation. WinSip configuration:</p> <ul style="list-style-type: none"> • Option ->Preferences: <ul style="list-style-type: none"> • sip address: sip:UserName@ComputerName(.Domain) i.e. "sip:at7@st72116.inf.tu-dresden.de" • Local port = 5060 • Server port = 0 <p>To call</p> <ul style="list-style-type: none"> • Create a buddy: <ul style="list-style-type: none"> • UserName = NT login name i.e. sip:ff1@st72113.inf.tu-dresden.de • Create a new session

8.2.2 Unreal Tournament Client Epic Games & Digital Extremes

APPLICATION NAME	Unreal Tournament (client with integrated server)
Version	
Operating system (incl. Service packs)	Windows 95, 98, or NT (or Unix, Macintosh)
CPU	<ul style="list-style-type: none"> • Min: Pentium 233 MHz MMX / AMD K6 • Typical: Pentium II 300MHz AMD K6-3 • Awesome: Pentium III 500 / AMD Athlon 550
Memory	<ul style="list-style-type: none"> • Min: 32 Mb RAM • Typical: 64 Mb RAM • Awesome: 128 Mb RAM
License required? Yes	Freeware? Yes - Trial version
If yes: license price? www.amazon.com : 20\$	If yes: for how many clients?
Network requirements	
Protocol	UDP

Ports	Default 7777
Access	Min. 28.8 K modem
Comments	
Other required equipment	
Equipment 1	Video card
Characteristic a	<ul style="list-style-type: none"> • Min: 4MB • Typical: 3dfx Voodoo 2 / Riva TNT class 3d accelerator • Awesome: 3dfx Voodoo 3 / Riva TNT2 class 3d accelerator
Equipment 2	Sound card
Other required software	
DirectX (Direct3D)	
special installation procedure	
special configuration files	
servers to be set up	
Possible troubleshooting	
Comments	<p>For optimal network play performance, you can launch a dedicated copy of the Unreal Tournament server on a computer. This improves performance compared to using a non-dedicated server but, of course, it ties up a PC.</p> <p>Anybody may freely run dedicated servers on the Internet; you don't need to get permission or fill out any paperwork.</p>

8.2.3 NetMeeting

APPLICATION NAME	NetMeeting
Version	3.01
Operating system (incl. Service packs)	Windows 98, Windows NT4.0, Windows 2000. Already installed in Win2000
CPU	a Pentium 133 or later processor recommended
Memory	At least 16MB of RAM for Win98 At least 32MB of RAM for WinNT4, Win2000
Hard Disk	At least 5MB (plus another 10MB during installation)
License required? no	Freeware? Yes
	No restrictions
Network requirements	
Protocol	TCP/IP for H.323 (and other control protocols) TCP/IP for data streams (whiteboard, file transfer, program sharing, chat) UDP/IP (RTP + RTSP) for audio and video streams
Ports	1720 (H.323 connect) All other ports are dynamically negotiated.
Access	A 28.8 Kbps or faster modem, ISDN, or LAN connection
Comments	
Other required equipment	
Equipment 1	Sound card with microphone and speakers
Equipment 2	Cameras with a video capture card, universal serial bus (USB) cameras, or a parallel port camera that provides a Video for Windows driver.
Characteristic a	USB cameras and cameras that require a video capture card typically offer better performance than cameras that plug into

	parallel ports. Parallel port cameras can require more of the CPU, resulting in decreased video performance.
Characteristic b	A black-and-white camera typically offers better performance than an equivalent colour camera because fewer bits are sent across the connection.
Other required software	
	Microsoft Internet Explorer 4.01 (or later) – not further analysed.
Special installation procedure	During installation, one of the parameters to be setup is the underlying Network Bandwidth. There are 4 choices: 14400 bps modem 28800 bps or faster modem Cable, xDSL, ISDN Local Area Network
Special configuration files	
Servers to be set up	
Possible troubleshooting	
Comments	Uninstall earlier versions of Microsoft NetMeeting before you install NetMeeting 3 The Audio Tuning Wizard: (a) detects if a microphone is attached to the computer, (b) If the computer has more than one sound card, the user can select which sound card to use for recording and playback, (c) configures a voice modem, which allows the user to talk and transmit data at the same time.

8.2.4 RealSystem

APPLICATION NAME	Streaming Media – Real System RealServer is the Server Application
Version	8.0 or upper
Operating system (incl. Service packs)	Linux 2.2 glibc
CPU	minimum Penium III/600 for RealServer
Memory	128MB for the Server
License required? Yes - no	No – for limited number of streams up to 25
Network requirements	
Protocol	TCP/IP UDP/IP RTSP PNA
Ports	554 for RTSP requests 7070 for PNA requests 8080 TCP HTTP requests 6970-6999 UDP Data port (port numbers are not configurable)
Access	28.8 KB Modem or faster ISDN LAN- Connection
Comments	Those network requirements are for the communication between the RealServer and the RealPlayer. There are some other requirements to administrate the RealServer or to encode some new streams.
Other required equipment	
Equipment 1	video card
Characteristic a	min. 4MB

Equipment 2	sound card
Characteristic a	16 bit
Other required software	
	IE 4.0.1 or Netscape 4.0 or higher
special installation procedure	Download of the file RealServer Basic Version Change user rights: <code>chmod a+x g2_7_0update2-linux-c6.bin</code> execute the file: <code>./g2_7_0update2-linux-c6.bin</code> answer the questions at the screen, the default values have to confirm, the destination directory is e.g.: <code>/usr/local/real</code> after the correct installation the RealServer will start
special configuration files	
servers to be set up	
possible troubleshooting	
Comments	
APPLICATION NAME	Streaming Media – Real System RealPlayer is the End-User Application
Version	8.0 or upper
Operating system (incl. Service packs)	Linux or Windows for the RealPlayer
CPU	minimum Pentium 233 for RealPlayer
Memory	minimum 64 MB
License required? Yes - no	No (Basic Version)
Network requirements	
Protocol	TCP/IP UDP/IP RTSP PNA
Ports	
Access	28.8 KB Modem or faster ISDN LAN- Connection
Comments	Those network requirements are for the communication between the RealServer and the RealPlayer. There are some other requirements to administrate the RealServer or to encode some new streams.
Other required equipment	
Equipment 1	video card
Characteristic a	min. 4MB
Equipment 2	sound card
Characteristic a	16 bit
Other required software	
	IE 4.0.1 or Netscape 4.0 or higher
Special installation procedure	Download of the file RealPlayer Basic Version execute the file during the installation it is necessary to specify the connection speed.
Special configuration files	
Servers to be set up	
Possible troubleshooting	
Comments	

9 Results and experiences

For the integration two meetings were held. The first one took place in Dresden, where the functionality of the Resource Control Layer was tested. The interoperability of the RCL components was tested and at the end of the meeting we were able to establish end-to-end reservations over a “virtual” DiffServ network. The term virtual signifies the absence of any routers in this integration phase.

The second integration meeting was held in Warsaw, which hosts one of the trial sites of the AQUILA project. The integration continued the work of the first meeting, 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 Vienna 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. Many experiences were gained from this process on how to conduct an integration process and what to avoid. The most important points are outlined in the following paragraphs.

It is of utmost importance to plan carefully the objectives of the integration, the time plan to be followed, the testbed sites to be used and the tests to be performed. The maintenance of a software library and a versioning method will also ensure the integrity and consistency of the produced software.

For the planning of integration meetings, it is very important to have all the equipment available and all the necessary software already installed. It can be a great waste of time, if supporting software should be set up during the meeting. As far as the installation of the prototypes to other sites is concerned, it is not as straightforward a task as it seems. It does not just involve copying software from a site to another, but also setting up this site, which is often a complicated procedure. Therefore, enough time should be foreseen for this work, but also such installation should better be performed by the same people that did it in the first place.

10 Abbreviations

ACA	Admission Control Agent
API	Application Programming Interface
BSS	Bucket Size for Sustainable Rate
BSP	Bucket Size for Peak Rate
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
EAT	End-user Application Toolkit
EDAdapter	Edge Device Adapter
GUI	Graphical User Interface
GPS	Global Positioning System
IDL	Interface Definition Language
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LLQ	Low Latency Queuing
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
RUT	Router Under Test
SLS	Service Level Specification
SR	Sustainable Rate
TC	Traffic Class
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Detection
XML	Extensible Mark-up Language

11 References

- [D1301] IST-1999-10077-WP1.3-COR-1301-PU-O/b0, Specification of traffic handling for the first trial
- [D2101] IST-1999-10077-WP2.1-SAG-2101-RE-O/b0, Design and functional specification of the Resource Control Agent for the first trial
- [D2201] IST-1999-10077-WP2.2-TUD-2201-RE-O/b0, Specification of End-user Application Toolkit
- [D2301] IST-1999-10077-WP2.3-SPU-2301-PU-R/b0, Report on the development of measurement utilities for the first trial

12 Appendix

12.1 LDAP Installation

12.1.1 *Evaluation of Netscape Directory Servers*

There are two versions of Netscape Directory Server available in the market, 4.11 and 4.12. Version 4.12 of Netscape Directory Server is included in the new version of Solaris (version 8) with a license for 200.000 users.

For each of the above versions the following platforms and requirements are supported.

12.1.1.1 **Netscape Directory Server 4.11**

This release of the directory server is supported on the following platforms with the noted requirements:

- Solaris 2.5.1, 2.6, or 7 (With the Sun recommended patch set).
- HP-UX 11
- IBM AIX 4.3.2.
- Digital Unix 4.0D.
- SGI IRIX 6.5.
- Redhat Linux 6.0 (Including kernel 2.2.5 and glibc 2.1.1).
- NT 4.0 with service pack 4. The server must be installed on an NTFS file system.

On all platforms, 32 MB of RAM are required to run Directory Server 4.11.

64 MB are recommended (more memory is required for large directory sizes).

A minimal installation of the Directory Server requires 200 MB of disk space. More disk space is required as your directory size (and therefore your database) grows.

As of this release, the graphical user interface for the server is provided through a Java application called the Netscape Console. The Netscape Console runs on all platforms supported by the server, as well as on Windows 95.

A stand-alone installation of the console requires 64 MB of memory and 40 MB of disk space.

12.1.1.2 Netscape Directory Server 4.12

This release of the directory server is supported on the following platforms with the noted requirements:

- Solaris 2.6, 7, 8 (32-bit OS mode with the Sun recommended patch set).
- HP-UX 11
- IBM AIX 4.3.2.
- Digital Unix 4.0D.
- Redhat Linux 6.0 (Including kernel 2.2.5 and glibc 2.1.1).
- NT 4.0 with service pack 5.

On all platforms, 64 MB of RAM are required to run Directory Server 4.12. More memory is required for large directory sizes.

A minimal installation of the Directory Server requires 200 MB of disk space. More disk space is required as your directory size (and therefore your database) grows.

As of this release, the graphical user interface for the server is provided through a Java application called the Netscape Console. The Netscape Console runs on all platforms supported by the server, as well as on Windows 95.

A stand-alone installation of the console requires 64 MB of memory and 40 MB of disk space.

12.1.1.3 Conclusion

According to the above and considering AQUILA requirements for the persistence layer the most suitable choice will be Netscape Directory Service 4.12 for all the cases of operating systems. In case of using Solaris operating system for the trials that should be Solaris 8 as it includes the Directory Server in it.

12.1.2 Installation Procedures

The installations tested so far are:

- Win NT 4.0 with SP5 + Netscape Directory Server 4.11
- Solaris 2.5.1 + Netscape Directory Server 4.11

The installation procedure for version 4.12 is exactly the same as version 4.11 with an exception on Solaris 8 workstations. In that case the Directory Server is either pre-installed or – if not already installed – the installation procedure is different.

Before you can install the Netscape Directory Server, you must make sure you have met the minimum hardware and operating system requirements, which are as follows:

- Roughly 200 MB of disk space for a bare-bones installation. For production systems, you should plan at least 1 GB to support the product binaries, database, and log files; 2 GB and greater may be required for very large directories.
- 32 MB of RAM. However, you should plan from 256 MB to 1 GB of RAM for best performance on large production systems.
- One of the following operating systems at the appropriate version and patch levels:
 - Sun Solaris
 - Hewlett-Packard HP-UX
 - Microsoft Windows NT
 - IBM AIX
 - Digital Unix
 - Linux
 - SGI IRIX

DNS must be properly configured on your system, and a static IP address must be assigned to your machine.

It is recommended that you install Directory Server as root (under Unix) or administrator (under NT).

Root privileges are required for Directory Server installations if you plan to use the default port numbers (which are less than 1024).

12.1.2.1 Netscape Directory Server Software Components

The Netscape Directory Server product contains four separate software components as follows:

12.1.2.1.1 Netscape Console

The Netscape Console provides the common user interface for all Netscape server products. From there you can perform common server administration functions, such as stopping and starting servers, installing new server instances, and managing user and group information. Netscape Console can be installed stand - alone on any machine on your network and used to manage remote servers.

12.1.2.1.2 Administration Server

The Administration Server is a common front end to all Netscape servers. It receives communications from Netscape Console and passes those communications on to the appropriate Netscape server. Your site will have at least one Administration Server for each server root in which you have installed a Netscape server.

12.1.2.1.3 Directory Server

The Directory Server is Netscape's LDAP implementation. The Directory Server runs as the ns-slaped process (Unix) or slapd service (Windows NT) on your machine. This is the server that manages the directory databases and responds to client requests. The Directory Server is a required component.

For fault-tolerance, you may choose to install additional instances of the Directory Server on different machines throughout your organization to store copies of all or part of the directory tree. This process is called replication. The server that contains the master copy of the directory data is called the supplier server. The server(s) that contain the replicated directory data are called consumer servers. The order in which you install supplier and consumer servers depends on whether you are performing a new installation or an upgrade.

12.1.2.1.4 Directory Server gateway

The Directory Server gateway is an LDAP client that you can access from a web browser. You use LDAP clients to access or change directory information. The Directory Server gateway is automatically installed when you install a Directory Server instance. You can access the gateway either from the Administration Server or you can configure a web server to manage the gateway.

12.1.2.1.5 Directory Express

Directory Express is a customized version of the Directory Server gateway. It is intended for read-only directory access such as might be required for corporate phonebook usage. Directory Express is installed and managed in the same way as the Directory Server gateway.

12.1.2.1.6 NT Synchronization Service

The NT Synchronization Service allows you to synchronize the entries in your Windows NT directory with your Netscape Directory Server entries. Install this component only if you want to synchronize your Netscape directory with your Windows NT directory so that when entries are created, modified, or deleted in one directory, the Synchronization Service makes the corresponding change to the other directory.

12.1.2.2 Basic Configuration Information

During Directory Server installation, you will be prompted for basic configuration information. You should decide how you are going to configure these basic parameters before you begin the installation process. You will be prompted for some or all of following information (depending on the type of installation that you decide to perform):

12.1.2.2.1 Choose Unique Port Numbers

Port numbers can be any number from 1 to 65535. Keep the following in mind when choosing a port number:

The standard Directory Server (LDAP) port number is 389.

The standard encrypted (LDAPS) port number is 636. Do not use this port number for the standard port even if 636 is not already in use.

Under Unix, the Administration Server must run as root if the Directory Server is going to use standard port numbers (only root authenticated user accounts can access ports lower than 1024.)

Under Windows NT, the Administration Server must run with administrator privileges if the Directory Server is going to use standard port numbers.

Make sure the ports you choose are not already in use. Further, if you are using both LDAP and LDAPS communications, make sure the port numbers chosen for these two types of access are not identical.

12.1.2.2.2 Create a New Server Root

Your server root is the directory where you install your Netscape servers. In the Netscape Directory Server documentation, it is referenced as <NSHOME>.

The server root must meet the following requirements:

The directory must be on a local disk drive -- you cannot use a networked drive. For installation purposes the directory must be empty, or contain only Netscape 4.x servers. The server root directory must not be the same as the directory from which you are running the set up program.

By default, the server root directory is:

- /usr/netscape/server4 on Unix systems
- c:\netscape\server4 on Windows NT systems

12.1.2.2.3 *Decide Which User and Group to Run Netscape Servers As (Unix only)*

For security reasons, it is always best to run Unix-based production servers with normal privileges. That is, you do not want to run the Directory Server with root privileges. However, you will have to run the Administration Server with root privileges if you want to use the standard Directory Server ports.

You must therefore decide what user accounts you will use for the following purposes:

The user and group to run the Directory Server as:

For Unix systems that support it, the special user account nobody is recommended. Also use group *nobody* if it is allowed by your operating system.

The user and group to run the Administration Server as:

For installations that use the default port numbers, this must be root. However, if you can use ports over 1024, then you should run the Administration Server as a normal user and group (use nobody if it is allowed by your operating system).

For sites that are installing multiple Netscape Servers, consider installing each server under a unique user name (such as uid slapd for Directory Server, or uid msg for Messaging Server). Doing this can help you with general system administration. However, you should use a common group for all Netscape servers, such as gid Netscape, to ensure that files can be shared between servers when necessary.

Before you can install the directory and the Administration Servers, you must make sure that these user and group accounts exist on your system.

12.1.2.2.4 *Defining Authentication Entities*

As you install the Netscape Directory Server and the Administration Server, you will be asked for various user names, distinguished names (DN), and passwords. This list of login and bind entities will differ depending on the type of installation that you are performing:

12.1.2.2.5 *Directory Manager DN and password*

The Directory Manager DN is the special directory entry to which access control does not apply. Think of the directory manager as your directory's super user. (In the 1.x and 3.x release of the Directory Server, the Directory Manager DN was known as the root DN). The default Directory Manager DN is cn=Directory Manager. Because the Directory Manager DN is a special entry that is not stored in the directory tree (instead, it is stored in slapd.conf), the Directory Manager DN does not have to conform to any suffix configured for your Directory Server. Also, you should not create an actual Directory Server entry to use with the directory manager DN.

The Directory Manager password must be at least 8 characters long.

12.1.2.2.6 Configuration Directory Administrator ID and password

The configuration directory administrator is the person responsible for managing all the Netscape Servers accessible through the Netscape console. If you log in with this user ID, then you can administer any Netscape server that you can see in the server topology area of the Netscape console.

For security, the configuration directory administrator should not be the same as the directory manager. The default configuration directory administrator ID is admin.

Administration Server User and password

You are prompted for this only during custom installations. The Administration Server user is the special user that has "root" privileges for the local Administration Server. Authentication as this person allows you to administer all the Netscape servers stored in the local server root.

The Administration Server user ID and password is used only if the Directory Server is down and so you are unable to log in as the configuration directory administrator. The existence of this user ID means that you can access the Administration Server and perform disaster recovery activities such as starting the Directory Server, reading log files, and so forth.

For most situations, the Administration Server user and password should be identical to the configuration directory administrator ID and password

12.1.2.2.7 Determine your Directory Suffix

A directory suffix is the directory entry that represents the first entry in a directory tree. You will need at least one directory suffix for the tree that will contain your enterprise's data. Netscape recommends that you select a directory suffix that corresponds to the DNS host name used by your enterprise. For example, if your organization uses the DNS name of airius.com, then select a suffix of o=airius.com. However, in AQUILA the root directory is o=AQUILA, c=EU.

12.1.2.2.8 Determine the Location of the Configuration Directory

All 4.x Netscape servers use an instance of the Directory Server to store configuration information. This information is stored in the o=NetscapeRoot directory tree. Your configuration directory is the Directory Server that contains the o=NetscapeRoot tree used by your Netscape servers.

If you are installing the Directory Server only to support other Netscape servers, then that Directory Server is your configuration directory. If you are installing the Directory Server to use as part of a general directory service, then you will have multiple Directory Servers installed in your enterprise and you must decide which one will host the o=NetscapeRoot tree. You must make this decision before you install any 4.x Netscape servers (including Netscape Directory Server).

For ease of upgrades, it is recommended you use a Directory Server instance that is dedicated to supporting the o=NetscapeRoot tree; this server instance should perform no other function with regard to managing your enterprise's directory data. Also, do not use port 389 for this server instance because doing so could prevent you from installing a Directory Server on that host that can be used for management of your enterprise's directory data.

Because the configuration directory normally experiences very little traffic, you can allow its server instance to coexist on a machine with another, more heavily loaded, Directory Server instance. However, for very large sites that are installing a large number of Netscape servers, you may want to dedicate a low-end machine to the configuration directory so as to not hurt the performance of your other production servers. Netscape server installations result in write activities to the configuration directory. For large enough sites, this write activity could result in a short-term performance hit to your other directory activities.

Also, as with any directory installation, consider replicating the configuration directory to increase availability and reliability. See the Netscape Directory Server Deployment Guide for information on using replication and DNS round robins to increase directory availability.

WARNING. Corrupting the configuration directory tree can result in the necessity of reinstalling all other Netscape servers that are registered in that configuration directory. Remember the following guidelines when dealing with the configuration directory:

- Always back up your configuration directory after you install a new Netscape server.
- Never change the hostname or port number used by the configuration directory.
- Never directly modify the configuration directory tree. Only the set-up program for the various Netscape servers should ever modify the configuration.

12.1.2.2.9 Determine the Location of the User Directory

Just as the configuration directory is the Directory Server that is used for Netscape server administration, the user directory is the Directory Server that contains your enterprise's data. That is, this is the Directory Server that contains the information that you want the Directory Server to store.

For most directory installations, the user directory and the configuration directory should be two separate server instances. These server instances can be installed on the same machine, but for best results you should consider placing the configuration directory on separate physical machine.

Between your user directory and your configuration directory, it is your user directory that will receive the overwhelming percentage of the directory traffic. For this reason, you should give the user directory the greatest computing resources. Because the configuration directory

should receive very little traffic, it can be installed on a machine with very low-end resources (such as a minimally-equipped Pentium).

Also, you should use the default directory ports (389 and 636) for the user directory. If your configuration directory is managed by a server instance dedicated to that purpose, you should use some non-standard port for the configuration directory.

You cannot install a user directory until you have installed a configuration directory somewhere on your network.

12.1.2.2.10 Determine the Administration Domain

The administration domain allows you to logically group Netscape servers together so that you can more easily distribute server administrative tasks. A common scenario is for two divisions in a company to each want control of their individual Netscape servers. However, you may still want some centralized control of all the servers in your enterprise. Administration domains allow you to meet these conflicting goals.

Administration domains have the following qualities:

- All servers share the same configuration directory, regardless of the domain that they belong to.
- Servers in two different domains may use two different user directories for authentication and user management.
- The configuration directory administrator has complete access to all installed Netscape servers, regardless of the domain that they belong to.
- Each administration domain can be configured with an administration domain owner. This owner has complete access to all the servers in the domain but does not have access to the servers in any other administration domain.
- The administration domain owner can grant individual users administrative access on a server-by-server basis within the domain.

Typical and Custom Installation ask you to identify the administration domain that the server will belong to. For many installations, you can have just one administration domain. In this case, pick a name that is representative of your organization.

For other installations, you may want different domains because of the demands at your site. In this latter case, try to name your administration domains after the organizations that will control the servers in that domain.

For example, if you are an ISP and you have three customers for whom you are installing and managing Netscape servers create three administration domains each named after a different customer.

12.1.2.3 Common Problems during installation

12.1.2.3.1 Clients cannot locate the server

First, try using the host name. If that does not work, use the fully qualified name (such as www.domain.com), and make sure the server is listed in the DNS. If that does not work, use the IP address.

12.1.2.3.2 The port is in use

You probably did not shut down a server before you upgraded it. Shut down the old server, and then manually start the upgraded one.

Another installed server might be using the port. Make sure the port you have chosen is not already being used by another server.