




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

Deliverable Number:	IST-1999-10077-WP1.2-SAG-1203-PU-O/b1
Contractual Date of Delivery to the CEC:	March 31, 2002
Actual Date of Delivery to the CEC:	April 26, 2002: version b0 February 26, 2003: version b1
Title of Deliverable:	Final system specification
Workpackage contributing to the Deliverable:	WP 1.2
Nature of the Deliverable:	O – Other (Specification)
Editor:	Martin Winter (SAG)
Author(s):	Andrzej Bak (WUT); Estíbaliz Bear (SAG); Marek Dabrowski (WUT); Lila Dimopoulou (NTU); Gerald Eichler (DTA); Reinhard Frank (SAG); Falk Fünfstück (TUD); John Karadimas (QSY); Eugenia Nikolouzou (NTU); Stefano Salsano (COR); Petros Sampatakos (NTU); Anne Thomas (TUD); Haris Tsetsekas (NTU); Ralf Widera (DTA); Martin Winter (SAG)

Abstract:	This deliverable D1203 specifies the final system architecture of the resource control agent. It may serve as a reference to the structure of the AQUILA approach to QoS. It also includes the software architecture of the resource control layer.
Keyword List:	AQUILA, IST, architecture, resource control layer, QoS

Executive Summary

The AQUILA project aims at the dynamic provision of Quality of Services features for end-users over the existing Internet. AQUILA has developed an architecture that allows end-users to have application sessions where the communication is of higher quality than nowadays, and to explicitly request for such QoS sessions.

This deliverable describes approach of the project and documents the architecture of the chosen solution on the level of the network and offered services as well as the structure of the components of the control layers.

The document serves as a general description of the AQUILA solution. As such it tries to be self-contained and should be readable without prior detailed knowledge of the project. However it is helpful to consult other documents of the project, especially from workpackage 1.3.

Table of Contents

1	INTRODUCTION.....	7
2	APPROACH AND DEFINITIONS	8
2.1	APPROACH	8
2.1.1	Resource distribution & pools	10
2.1.2	Inter-domain resource control.....	11
2.1.3	Network topology	11
2.1.4	Control loops	12
2.1.5	Reservation groups.....	12
2.1.6	Prioritised signalling	12
2.1.7	End-user application support	13
2.1.8	Management	13
2.2	DEFINITIONS	14
3	SERVICES AND NETWORK ARCHITECTURE.....	21
3.1	SERVICES.....	21
3.1.1	Network Services.....	21
3.1.2	Characterisation of a Network Service.....	21
3.1.3	Implementation of Network Services.....	22
3.1.4	Traffic Classes	22
3.1.5	Mapping of Network Services to Traffic Classes.....	22
3.2	NETWORK ARCHITECTURE	23
3.2.1	Overview.....	23
3.2.2	Access Network Architecture	23
3.2.3	Core network architecture	24
3.2.4	Network Topology	26
4	RESOURCE CONTROL AND INTER-DOMAIN LAYER ARCHITECTURE.....	32
4.1	RESOURCE CONTROL LAYER.....	32

4.1.1	Admission control.....	33
4.1.2	Resource distribution	36
4.1.3	Small Bandwidth Links	40
4.1.4	Deployment of logical entities.....	41
4.1.5	Roles.....	42
4.1.6	Reservation groups.....	43
4.1.7	Prioritised signalling and control traffic.....	45
4.2	INTER-DOMAIN RESOURCE ALLOCATION	47
4.2.1	State of the art.....	48
4.2.2	Requirements	51
4.2.3	Architecture	51
4.2.4	Detailed aspects	52
4.2.5	Scalability	61
4.3	END-USER APPLICATION TOOLKIT	67
4.3.1	Overall architecture	67
4.3.2	Non QoS-aware application support	69
4.3.3	End-user Application API.....	74
4.3.4	GUIs, Converter, and Application Profiles	79
4.3.5	Protocol gateways.....	80
4.3.6	Deployment Scenario.....	89
4.4	MANAGEMENT	91
4.5	SUPPORTING ENTITIES	93
4.5.1	Router control.....	93
4.5.2	Pluggable algorithms	94
5	ABBREVIATIONS.....	97
6	REFERENCES	100

Table of Figures

FIGURE 2-1: THE FINAL AQUILA ARCHITECTURE.....	10
FIGURE 3-1: GENERAL AQUILA NETWORK ARCHITECTURE	23
FIGURE 3-2: PACKETS FORWARDING IN EDGE DEVICE.....	25
FIGURE 3-3: PACKETS FORWARDING IN CORE ROUTER.....	26
FIGURE 3-4: TYPICAL ACCESS NETWORK TOPOLOGY	26
FIGURE 3-5: EXAMPLE OF NETWORK TOPOLOGY	27
FIGURE 3-6: EXAMPLES OF CORE NETWORK STRUCTURES.....	28
FIGURE 3-7: ACCESS TO THE CORE NETWORK USING LOW-BANDWIDTH LINKS	29
FIGURE 3-8: EXAMPLE OF DISADVANTAGEOUS SITUATION WHEN LOCAL TRAFFIC INFLUENCES OTHER POOLS..	30
FIGURE 3-9: EXAMPLE OF UNDESIRE ROUTING BETWEEN TWO DIRECTLY LINKED SUB-AREAS.....	30
FIGURE 3-10: EXEMPLARY HIERARCHICAL NETWORK TOPOLOGY	31
FIGURE 4-1: MAPPING OF RCL ENTITIES TO THE UNDERLYING NETWORK ENTITIES	32
FIGURE 4-2: RESOURCE CONTROL (ADMISSION CONTROL LOOP).	34
FIGURE 4-3. MEASUREMENT ARCHITECTURE FOR ADMISSION CONTROL LOOPS.....	35
FIGURE 4-4. LINK MONITORED BY THE MBACCOLLECTOR ON THE INGRESS AND EGRESS SIDE OF THE ED.....	35
FIGURE 4-5: HIERARCHICAL RESOURCE POOLS	37
FIGURE 4-6: INTERACTIONS BETWEEN ACA AND RCA.....	39
FIGURE 4-7: EXAMPLE NETWORK WITH SMALL BANDWIDTH LINKS.....	40
FIGURE 4-8: BI-DIRECTIONAL RESERVATION REQUEST USING RESERVATION GROUPS.....	43
FIGURE 4-9: EXAMPLE OF RESERVATION GROUPS IN A CONFERENCE SCENARIO.....	44
FIGURE 4-10: EXAMPLE OF JOIN MESSAGE IN A CONFERENCE SCENARIO.....	45
FIGURE 4-11: AQUILA RESOURCE CONTROL LAYER.....	46
FIGURE 4-12: SIBBS: ARCHITECTURE AND MESSAGE FLOW	49
FIGURE 4-13: GENERAL INTER-DOMAIN ARCHITECTURE AND MESSAGE FLOW	52
FIGURE 4-14-UNIQUENESS OF A CIDR LABELLED SINK TREE.....	63
FIGURE 4-15-DISTRIBUTION OF DESTINATION CIDR ADDRESSES.....	65
FIGURE 4-16: BLOCK DIAGRAM OF THE OVERALL EAT ARCHITECTURE	68
FIGURE 4-17: RELATION BETWEEN THE RESERVATION GUI AND THE APPLICATION PROFILE	70
FIGURE 4-18: COMPLEX INTERNET SERVICE AND AQUILA	72
FIGURE 4-19: BLOCK DIAGRAM REPRESENTING THE EAT AND THE CIS.....	73
FIGURE 4-20: QOS APIS AT DIFFERENT LEVELS.....	76
FIGURE 4-21: THE INTERNAL EAT API, PART 1	78
FIGURE 4-22: THE INTERNAL EAT API, PART 2	79
FIGURE 4-23: SEQUENCE DIAGRAM FOR USUAL REQUEST	80

FIGURE 4-24: PROXY ARCHITECTURE	82
FIGURE 4-25: SIP SCENARIO	84
FIGURE 4-26: QOS SIP SCENARIO	85
FIGURE 4-27: SAMPLE MESSAGE FLOW	88
FIGURE 4-28: DEPLOYMENT SCENARIO OF THE EAT COMPONENTS.....	90
FIGURE 4-29: DEPLOYMENT SCENARIO FOR THE MEDIAZINE EXAMPLE.....	91
FIGURE 4-30: QMTOOL INTERFACES.....	93
FIGURE 4-31: RESOURCE POOL HIERARCHICAL STRUCTURE.....	95

Table of Tables

TABLE 4-1: POSSIBLE SOLUTIONS TO DEVELOP A QOS AWARE COMPLEX INTERNET SERVICE	72
---	----

1 Introduction

The document serves as a general description of the AQUILA solution. As such it tries to be self-contained and should be readable without prior detailed knowledge of the project. The document achieves this goal by choosing the following structure:

- Chapter 2 describes the overall approach of the project and provides an overview of the architecture of the AQUILA solution. Some important topics are shortly introduced. Furthermore, it contains definitions of terms used in this specification and other AQUILA documents.
- Chapter 3 copes with the structure of the network, which is the basis for AQUILA's QoS approach. It explains, how the services offered to the user are mapped to the technical transport characteristics of a DiffServ network.
- Chapter 4 specifies the architecture of the software components in the control layers. Intra-domain and inter-domain resource control are two fundamental building blocks of this architecture, which can provide a certain behaviour for packet transport across the Internet. Quality of service, however, it not just a matter of transport characteristics, but more important is the user perception of the quality. The AQUILA project defines an end-user application toolkit, which performs the mapping from the user's perceptive expectations to the technical parameters of the network services, which is also described here. Furthermore, a management tool for the AQUILA components is presented in this chapter. Finally, some supporting functions are described.
- Chapter 5 provides a list of acronyms used in this document.
- Chapter 6 lists the references.

2 Approach and Definitions

2.1 Approach

The AQUILA project aims at the *dynamic* provision of Quality of Services features for end-users over the existing Internet. AQUILA has developed an architecture that allows end-users to have application sessions where the communication is of higher quality than nowadays, and to explicitly *request* for such QoS sessions.

For that, the AQUILA network offers different **network services** with different *predefined* QoS characteristics to the customers of the network and implement them internally by different **traffic classes** [D1302]. Network services can be seen as products provided by the QoS-enabled AQUILA network and designed for typical application requirements.

Customers can subscribe network services in order to have the policy to request for them for their applications. More specifically, customers initiate QoS requests by firstly specifying the network service, secondly the traffic characteristics of their application, and finally the reservation scenario (such as the flow characteristics for bi-directional reservations, for example).

To support this dynamic QoS provision, the AQUILA project have developed a flexible, extendable and *scalable* Quality of Service architecture for the existing Internet based on existing technologies for delivering QoS. This architecture is introduced in this chapter and described in more detail in the rest of this document.

In particular, the AQUILA core network is an enhanced DiffServ network providing several dynamically manageable traffic classes with specific QoS parameters, per hop behaviours, and other “guidelines” that realise different traffic handling for different network services to be requested. (Customers are connected via access networks to the edges of the core network.)

The AQUILA architecture mainly relies on the new Resource Control Layer (RCL) that acts as distributed bandwidth broker [RFC2638], controlling and providing the resources of the underlying DiffServ-aware network.

For each domain, a separate instance of the RCL is created and configured. It consists of the following distributed components that take care of the control of this domain:

- One **Resource Control Agent (RCA)** which is responsible for the control and the management of the overall resources of one DiffServ domain. Therefore it acts as bandwidth broker distributing the resources via a hierarchical resource pool tree to the ACAs of the domain, which act as the leafs of that tree.
- An **Admission Control Agent (ACA)** manages the local resource of one (edge or border) router in order to perform local admission control and policy control for resources assigned to

this router. An ACA communicates with other ACAs to allow reservations across the domain. Moreover, an ACA communicates with EATs in order to process QoS requests initiated by the EATs.

- An **End-user Application Toolkit (EAT)** is a kind of middleware between end-user applications and the network infrastructure. On behalf of end-users it requests for network resources in order to support applications to get the proper QoS for their communication. Each EAT can communicate with its corresponding ACA (i.e. the ACA of the edge router of its access network) but is not aware of any RCA.

(Note that RCA, ACA, EAT components are not physical devices but logical components which may be placed anywhere within the network.)

The approach, however, is not restricted to a single domain scenario. It is also able to control resources between different domains that are not necessarily all controlled by AQUILA Resource Control Agents. For that reason an additional inter-domain layer exists on the top of the underlying for example RCL controlled domains. It consists of several instances of the following type:

- **BGRP Agent**, to communicate reservation requests between domains that are locally controlled by RCAs, for example. It interacts with the egress and ingress ACAs, respectively, of the neighbour domains.

All components together form a two-levelled logical overlay network above the underlying core network [Figure 2-1].

Further components of the architecture are legacy as well as new QoS-aware applications (not shown in the figure) running on the hosts. Both use the EAT middleware to benefit from the QoS capabilities of the AQUILA approach, i.e. the EAT is always the QoS portal to the RCL for them.

Moreover, the QoS Management Tool (QMTool) is a software for network operators providing access to the network (also not shown in the figure). It allows the management of resources and services, the establishment and the maintenance of a distributed database. This database is responsible to keep resource and service relevant information, for instance.

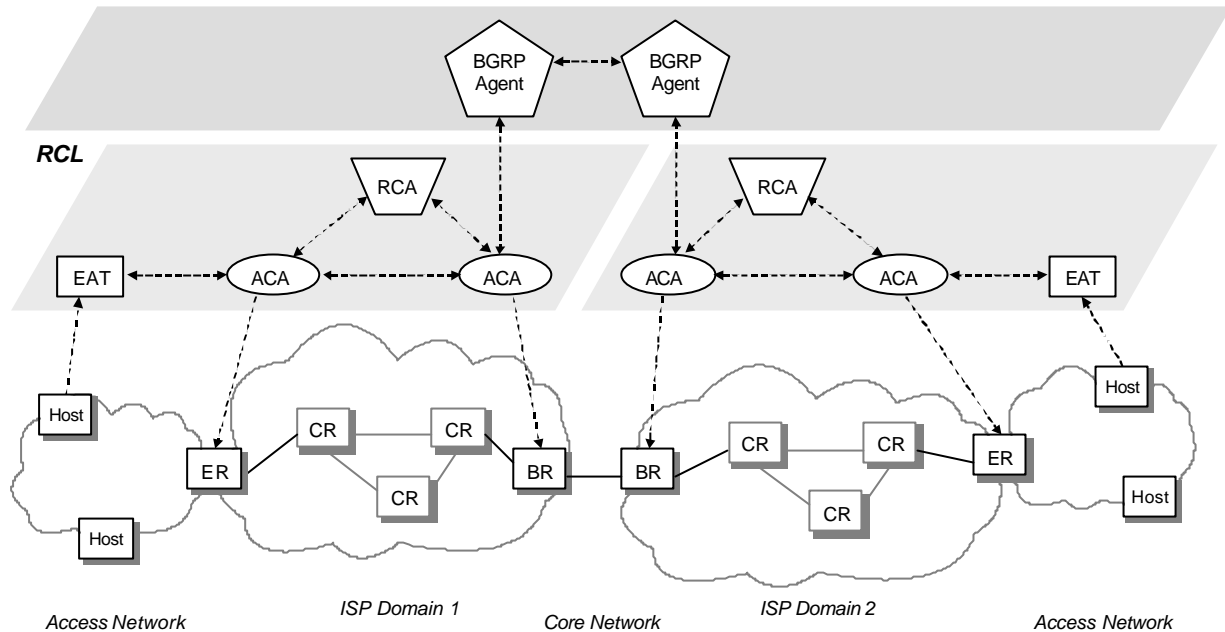


Figure 2-1: The final AQUILA architecture

One important goal of the AQUILA approach is to ensure platform independence. Therefore, the main components are realised by using the Java programming language, while the communication between the components is based on OMG's CORBA standard.

The following paragraphs shortly introduce the some important topics of the final architecture, which have been *implemented* by the project.

2.1.1 Resource distribution & pools

In AQUILA, resources are distributed by using the resource pool approach. The only resource handled by the resource pool tree is the *bandwidth*. That maximises the performance and makes the configuration of traffic classes in the hierarchical manner of the resource pool tree relatively easy.

In general, resource pools are configured in two phases: the tree creation phase based on static rules (initial resource provisioning), and the dynamic re-distribution phase. Latter may take place when an ACA asks for more resources, for example to admit a reservation request that exceeds its resource limit. On the other hand, resources may be returned if the sum of all reserved resources goes under a *low watermark*.

Resource pools are at intra-domain level, and the RCA is the component, which takes care of the creation of them. (Inter-domain issues are at BGRP level and handled there.)

More details can be found in the chapter 4.1.2.

2.1.2 Inter-domain resource control

Resource control between domains has some major differences to resource control within a domain. This enforces the need to establish new mechanisms to handle the inter-domain case.

- A domain's topology is built and controlled by a single network operator. Therefore, abstraction mechanisms like the resource pools can successfully be used to provide a coarse view of the topology. The Internet topology however, is based on a set of bilateral agreements between network operators.
- Scalability is an issue both for intra-domain and inter-domain resource control. One can handle single flows within a domain with an approach using distributed processing (one ACA per edge device). For a scalable inter-domain resource control however it is necessary to aggregate flows [BGRP].

Taking into account these differences, the project specifies and develops a different resource control mechanism used between domains, based on the proposal made in [BGRP]. A hop-by-hop reservation, based on the BGP routes, is used to follow the bilateral agreements of neighboured domains. Sink tree based aggregation of reservations is the base for scalability. Early reservation responses ("quiet grafting") are used to control the amount of signalling traffic. Open interfaces allow this reservation mechanism to be used in conjunction with any kind of intra-domain resource control, which can provide edge-to-edge QoS.

The project proposes a set of globally well known services to provide a common understanding of network services and to allow the creation of Internet-wide services.

More details can be found in the chapter 4.2.

2.1.3 Network topology

The AQUILA architecture considers – with regard to the practicability – characteristic Internet scenarios:

- Dual Hosting, where one host is connected to many edge routers. This influences the reservations, particularly the IP addresses.
- Dual Homing, where one edge router is connected to many core routers. The project has particularly to take care of situations where the links are not controlled by the same higher level pool.
- Dual Peering Points, where two autonomous systems (domains) are connected to many border routers. This issue has to be considered by the inter-domain resource control.

More details can be found in the chapter 3.2.

2.1.4 Control loops

Instead of using a rather conservative mechanism to distribute resources and admit resource requests (the so-called “open loop control”) – in which no feedback from the network is considered but mainly static mechanisms and rules – the final AQUILA approach *closes* the control loop by taking measured network parameters, such as utilisation, delay, etc., into account.

In AQUILA, control loops are relevant for:

- The measurement-based admission control (MBAC). At the (ingress or egress) router, passively measured (monitored) statistical parameters influence the decisions taken by the ACA in order to admit more requests and better utilise resources.
- The re-provisioning of resources. By using some feedback from the network, a more dynamic reconfiguration of the resource pools is possible, e.g. by adapting pool values, or by changing WFQ parameters in a longer time scale instead of directly shifting resources between the traffic classes.

More details can be found in the chapter 4.1.1.2.

2.1.5 Reservation groups

With the final architecture it is possible to group several reservation requests and send them together as a single request. The features are:

- Bi-directional services can be supported. Instead of requesting two independent, unidirectional reservations, a reservation bundle (or unit) is formed containing single reservations for both directions.
- Larger reservation groups are also supported, for example for multi-conference sessions. The end-user, moreover, had the possibility to join or to quit a former established group.
- The EAT offers reservation bundles and groups to end-user in a comfortable way. For example, only one reservation form for a bi-directional service has to be filled. The EAT will then internally form the bundle with the two single reservations and submit it to the ACA.

More details can be found in the chapter 4.1.6.

2.1.6 Prioritised signalling

The RCL in fact produces a significant signalling overhead. Signalling traffic occurs within the RCL (between the RCL components and the database) as well as between the RCL and external entities (such as end-users, applications, and DiffServ-aware routers).

Particularly at start-up, for reservation requests, database and router access, the RCL produces a sizeable message transfer [D3201]. This control traffic is, however, submitted by using the same physical links as the user traffic. It is thus important to prioritise the signalling traffic of the RCL in order to avoid situations where the user traffic affects the control traffic. Existing services for transaction-oriented applications might be reused for this purpose. Different solutions are proposed on how to use DSCPs for signalling traffic.

More details can be found in the chapter 4.1.7.

2.1.7 End-user application support

It is also the task of final architecture to support different types of end-user applications. This is done in a threefold way:

- For legacy applications that are not QoS-aware, the Regular Reservation GUI of the AQUILA QoS portal is a prototype that shows, how the integration of QoS offers in a Complex Internet Services (CIS) can be realised in a user-friendly way. The Application Profiles and the EAT's Converter are designed in a way that such services can easily be completed with the possibility to present network's QoS options and to allow reservation requests.
- For applications that are not EAT-based, additional protocol gateways (here also called: Proxies) are realised. These proxies will not only be able to detect flow information for the completion of reservation requests, but will directly translate QoS requests via signalling protocols into RCL-conform requests. So applications that use SIP, for example, to signal their QoS requirements can benefit from the RCL QoS capabilities as well.
- For new applications that are EAT-based, the EAT API offers direct access to the full AQUILA functionality regarding login, reservation requests at different levels (and reservation releases), reservation groups, network service and application profile retrieval etc.

More details can be found in the chapter 4.3.

2.1.8 Management

The final QMTool is an universal network management tool that is for example able to:

- design the resource pool tree in a user-friendly, visual manner,
- configure and monitor network elements,
- detect failures of RCL components,
- create and configure the network services, and
- create subscriber entries.

For the QMTool it is thus necessary to intensively inter-work with the LDAP database, on the one hand, and with the RCL and network elements, on the other hand.

More details can be found in the chapter 4.3.1.

2.2 Definitions

The following glossary defines general important terms used throughout this document and the whole project.

Access Provider. See network provider.

Administrative Domain. A collection of network elements under the same administrative control and grouped together for administrative purposes. It is usually managed by a single corporate entity. For QoS enforcement purposes, a **network domain** refers to any domain that shares a common QoS policy. It may or may not overlap with other kinds of domains like IP or NT domains.

Admission Control. The process of determining whether a flow can be granted the requested QoS [Ferg98]. Admission Control is processed by the network and can be resource and/or policy based.

Local Admission Control. Admission control based on locally managed resources and/or policies.

Admission Control Agent (ACA). A logical entity of the RCL. The ACA performs policy control and local admission control. There is a 1-1 relation between the logical entity ACA and the physical edge device.

Advanced Reservation Mode. Reservation mode in which the technical oriented and the very detailed QoS reservation form has to be filled. It is for a professional end-user that knows the meaning of each parameter of an AQUILA reservation request.

Application. In terms of AQUILA an **end-user application** that uses a network (i.e. the Internet) for communication-based purposes. Such applications mainly consists of two levels: Firstly, the underlying user program, and secondly, the online service to be made available.

Legacy Application. An end-user application which is not QoS-aware but can *indirectly* benefit from the QoS capabilities of the network.

QoS-aware Application. In terms of AQUILA an application that benefits *directly* from the QoS capabilities of the network by using either an API of a QoS middleware or an appropriate signalling protocol such as RSVP. (Applications that use the API of the EAT middleware are called **EAT-based applications** in the following.)

Application Profile. Unique description of one certain application containing in a hierarchical manner its network, technical, and session characteristics. Application Profiles aim at the storage of two

levels of QoS abstraction: a network oriented, technical view, and a human oriented, more abstract (session) view. In that way, they allow a mapping between both levels.

Autonomous System (AS). A self-connected set of networks that are generally operated within the same administrative domain.

Basic Internet Application. A usual Internet application such as Voice over IP, Video Conferencing, TV on Demand, FTP, Streaming, etc. A Basic Internet Application can be a standalone application or a web plug-in. It is often a legacy application, which is not QoS-aware.

BGRP Agent. A logical entity of the RCL. BGRP agents aggregate and communicate resource reservations between different network domains.

Border Router. See edge device.

Complex Internet Service. A online service offered by e.g. a content provider to a customer group in form of a web platform integrating, binding and presenting Basic Internet Applications to a combined, value-added service. The use of web technologies (HTML, Plug-ins, Java, JSPs, etc.) allow the realisation of such a service, on the one hand, and offer access to QoS APIs as the one provided by the EAT, on the other hand.

Content. The multimedia data offered to users of an online service, e.g. a video in a Video-on-Demand service, dynamic financial information in an online banking service, etc.

Content Provider. Somebody who offers content for online services.

Control Loop. Feedback from the network towards the RCL, in order to *close* the **Open Loop Control**. More precisely, the measured behaviour of the network elements is used to dynamically influence decisions taken by the RCL concerning the admission of flows (Measurement Based Admission Control) and the distribution of resources (re-provisioning).

Core Router. A router that is deployed at the core of an administrative domain.

Customer. An entity that purchases a specific network service. The customer acts either as an intermediate entity between the network provider and the end-user or as the end-user itself. In AQUILA a customer is equivalent to an end-user.

Edge Device. A device such as a router or a gateway that is deployed at the border of an administrative domain. This can be an inter-domain border (then also called **border router**) or the border to the hosts. Two specialisations exist specifying whether the edge device belongs to the core (provider edge, **edge router**) or to the access side of a network (customer edge, access router).

Edge Router. See edge device.

Egress. The point where traffic *leaves* the network or the domain. The receiver is located at this point.

End-user. A person or a group of persons external to the network that utilises the network to work on a task, to offer something, etc., by using so-called end-user applications.

End-user Application Toolkit (EAT). A logical entity of the RCL. The EAT mediates between the applications of the host and the ACA on the network.

End-user Application. See application.

End-user Service. See service.

Flow. In terms of AQUILA a set of packets belonging the same application session.

Globally Well Known Service (GWKS). A certain traffic behaviour (in terms of traffic classes) that describes common, over domain boundaries accepted QoS objectives. More specifically, there are no fixed QoS parameters but optimisation targets such as “low delay”, as well as traffic ranges such as “maximum packet size”. In different domains, a GWKS might be implemented in different ways.

Guarantee. The level of probability that an end-user gets the QoS he/she requested. While **hard guarantee** means the probability of 100%, a **soft guarantee** means a lower probability.

Host. Computer system on a network belonging to an end-user.

Ingress. The point where traffic *enters* the network or the domain. The sender is located at this point.

Link. A network communications channel consisting of a circuit or transmission path and all related equipment between a sender and a receiver. Most often used to refer to a WAN connection. Sometimes referred to as a line or a transmission link. In AQUILA the latter meaning is used, i.e. the connection from one hop to the next.

Measurement Based Admission Control (MBAC). Admission control that dynamically uses at network elements measured parameters to decide whether a flow can be granted the requested QoS or not. This is done in *addition* to the **State Based Admission Control (SBAC)** based on resource pools only.

Middleware. A software that occupies a position between an infrastructure (e.g. an operating system or a network) and applications, particularly in a distributed system.

Network Domain. See administrative domain.

Network Operator. An entity that is responsible for the development, provision and maintenance of network services and for operating the corresponding networks.

Network Provider. An entity that controls a network infrastructure and offers network services. A provider can act as an **access provider** to prepare network access and/or as a **service provider** to offer network services with a specific behaviour, and perhaps to charge and account them.

Network Resource. The capacities of a network infrastructure to be shared between several utilisation. Main resources are bandwidth of links and buffers within routers, for example.

Network Service. A product that a network provider offers to its customers. In detail, it describes how customer's traffic is handled across the network as it is implemented by one or more traffic classes. Usually, there will be a set of pre-defined services but also the possibility to request for special parameters.

Online Service. A product that somebody (e.g. an online provider) offers to another end-user or a group of end-users of any network. It requires user programs. Online services can be online and multimedia documents, client/server programs, or electronic commerce products, for instance, which may benefit from the use of network services.

Per Hop Behaviour (PHB). The forwarding treatment given to a specific class of traffic, based on criteria defined in the DiffServ field. Routers and switches use PHBs to determine priorities for servicing various traffic flows. There are currently two standard PHBs defined by the IETF: **Expedited Forwarding (EF)** [RFC2598] and **Assured Forwarding (AF)** [RFC2597].

Policy. (QoS Policy.) The binding of traffic recognition and registration profiles to specific network behaviours including, though not exclusive to:

- Admittance/denial of identified traffic getting anything better than best-effort QoS.
- Simple prioritisation or specific bandwidth reservation for identified flows or aggregated flows.

Policy Control. The process of determining whether access to a particular resource should be granted.

QMTTool (QoS Management Tool). A software tool for the network operator, providing a GUI for the visual creation and modification of resource pools and network services, as well as the configuration and surveillance of network and RCL elements.

QoS Monitoring. The watching of QoS parameters in order to give a feedback to the customer who pays for the QoS, or to the application to adapt its reservation.

Quality of Service (QoS). An overall measurement of the service quality based on certain key parameters [Black99]. QoS can be seen on two levels: In terms of end-user applications, it is expected to get the data in a sufficient manner with minimal delay or latency, minimal variations of delay (jitter), and error freeness.

In terms of a network, QoS is used to describe a connection, on which data are transmitted in a manner better than best-effort by using the network resources efficiently, and with minimal data loss.

Protocol Gateway. In terms of AQUILA a **Proxy** that resides between the applications and the network in order to get information which might be meaningful to *complete* or *initiate* reservation requests for the applications. In this way, QoS can be provided also for (legacy) applications that either dynamically negotiate data port numbers (e.g. via H.323), or that use special (QoS) protocols (e.g. SIP, RSVP) to signal their QoS needs.

A Proxy can act as **Application Level Gateway** or **Network Level Gateway** depending at what protocol level it works.

Proxy. See protocol gateway.

Receiver. See role.

Regular Reservation Mode. Reservation mode in which a non-professional end-user requests for a reservation by selecting a pre-defined, abstract QoS option. Such an option represents the technical behaviour of a QoS offer in a for usual end-users understandable manner.

Re-provisioning. The redistribution of resources by dynamically reconfiguring the resource pool tree.

Request. (QoS Request). An explicit demand for getting QoS from an infrastructure. Usually, signalling protocols such as RSVP are used for the request. However, requests could be based on APIs and CORBA as well.

Requester. See role.

Reservation. Part of a resource that has been dedicated for the use of a particular traffic type for a period of time through the application of policies. Three modes can occur: sender-initiated (forward reservation), receiver-initiated (backward reservation), as well as third-party-initiated.

Reservation Style. The amount of senders/receivers involved in a reservation. Generally, there are three types: point-to-point (p2p; one sender, one receiver), point-to-anywhere (p2a; one sender, loads of receivers), and anywhere-to-point (a2p; loads of senders, one receiver).

Reservation Adaptation. Behaviour of an application itself, or of a QoS middleware to adapt the reservation requests to the real needs of the application, considering for example, the QoS monitoring results.

Reservation Group. An association of several single reservations that have a common context. Reservation groups can be established for bi-directional reservations (consisting of two unidirectional single reservations), for several reservations that belongs to the same application but different service components, or for multi-conference sessions, for example. Reservation groups are requested by a single message.

Resource. See network resource.

Resource Control Agent (RCA). A logical entity of the RCL. The RCA controls the resources of one administrative domain and distributes them to the ACAs.

Resource Control Layer (RCL). An overlay network layer which monitors and controls the resources of the core DiffServ and access network as well as offers a QoS interface to the applications. The RCL consists of: ACAs, RCAs, and EATs.

RCL Platform. Physical platform, on which one or several ACAs and/or RCAs are running. May be a separate hardware entity or integrated into a router.

Resource Pool. The concept of distributing and sharing resources in a hierarchical tree structure. A Resource Pool consists of components that are either other (sub) Resource Pools or resource pool leaves. A Resource Pool is managed by an RCA, and visually created by using the QMTool.

Resource Pool Leaf. A component of a resource pool at the deepest level. In AQUILA, it is associated to an ACA.

Role. A host that participates on a network session can play the role of the **sender** of QoS traffic, or the role of the **receiver** of the QoS traffic. Additionally, the role of the **requester** which reserves for the QoS traffic can be played either by the sender, the receiver, or a third party.

Sender. See role.

Service. By default service is meant as synonymous to **end-user service**: A set of functions offered to an end-user by an organisation (service provider). From the network point of view, end-user services are products offered to the host.

However, these products are seen on two different abstraction levels from an end-user point of view: The end-user subscribes a set of network services, but chooses more abstract services by using his/her applications. *Due to these different views, it is proposed to avoid this term but to use the terms network services and session characteristics, respectively, instead.*

Service Component. Part of an application profile, describing one element/medium of an application. For example, a multimedia conferencing application may have three service components: “video”, “audio”, and “data”. The purpose is to allow several, different reservations per application due to the different requirements of its media.

Service Level Agreement (SLA). A contract between a network provider and a customer defining provider responsibilities in terms of network services.

Service Level Specification (SLS). A set of parameters (such as flow characteristics, traffic specification) and their values which together define the service offered to a traffic stream by a DS domain. In AQUILA, a SLS is used in combination with a network service (which can also be seen as **predefined SLS**) to exactly specify a request.

Service Provider. See network provider.

Session. Time during which an application uses a network with QoS.

Session Characteristics. An end-user does have the possibility to individualise his/her applications in order to choose their session quality. For example: He/she can either choose between different pre-defined video qualities or directly set the parameters such as frame rate, picture size, etc. These characteristics have to be mapped into network services.

Subscriber. Used within the RCL to identify a customer that has been subscribed a SLA with the network provider.

Traffic Class. In terms of AQUILA the *implementation* of a network service, i.e. the network view on that product. A traffic class contains rules how to handle the traffic belonging to this class such as per-hop behaviour, rules for traffic conditioning as well as for admission control.

Usage Data. A collection of statistic data consisting such parameters as start and end time, duration, volume, etc. for a flow that belongs to a reservation.

User Program. In terms of AQUILA a standard or special software that allows the use and the offer of online services. Typical examples for standard software are Web browsers like Netscape Communicator and Microsoft Internet Explorer as well as Web servers, FTP and E-mail programs as well as multimedia conferencing programs like Microsoft NetMeeting and Mbone tools, etc.

3 Services and Network Architecture

3.1 Services

3.1.1 Network Services

An AQUILA network offers a number of transport options for user IP traffic. These are called network services. Each network service provides a certain QoS, expressed by statistical statements about e.g. delay and packet loss. Also, for each Network service it is exactly defined, which parameters must be passed in a reservation request requesting this network service, and what values are allowed for these parameters. As part of these parameters are used for policing the traffic, the traffic allowed to use a given network service is forced to obey certain characteristics.

The network services and their characteristics are defined by the network operator. ACA and RCA only care about network services, especially the traffic classes mapped to mentioned defined network services, not about specific applications and their demands. The latter is covered by the EAT. The EAT has access to the network service data, and on the other hand knows the application demands. It maps application demands to network services. So, a reservation request from EAT to ACA specifies in particular the requested network service.

3.1.2 Characterisation of a Network Service

The following information characterises a network service:

- name or id identifying the network service
- requirements for requests
 - required reservation style (p2p or p2a) (i.e. is “any” as destination address allowed or not)
 - required traffic description

Procedure: There is a fixed, common, “maximal” traffic descriptor, comprising the `TOKEN_BUCKET_TSPEC` of RFC 2215. For the individual network service, the following is defined:

default values for all parameters

parameters, for which explicit values can be specified

ranges for values of the explicitly specifiable parameters

(This description contains some redundancy, which has been left intentionally)

- QoS: statistical information about

- delay

- jitter

- loss probability / degree of bandwidth guarantee

- packet ordering required / not required

Example: 95% of packets experience no queuing delay, at least 99% of packets are not discarded due to queue overflow, no packet disordering.

- Rules for admission control

3.1.3 Implementation of Network Services

While the RCL knows about network services, routers don't. Routers know DSCPs, and they have scheduling mechanisms treating packets according to their DSCPs. Edge devices also may look at other header fields in order to classify traffic.

The idea is to implement each network service by using one or more DSCPs. The routers are configured in such a way, that their IP forwarding behaviour for such traffic results in the QoS defined for the corresponding network service.

3.1.4 Traffic Classes

The term traffic class shall be used to describe the implementation of a network service.

So the following characterises a traffic class:

- Per-hop behaviour
- Rules for traffic conditioning

3.1.5 Mapping of Network Services to Traffic Classes

The ACA performs the mapping from network service to traffic class. Therefore, the RCL configuration database contains the available

- network services
- traffic classes
- mapping rules, allowing to derive the traffic class from the requested network service plus other parameters of a request

This database is also used to provide the available network services to the EAT.

3.2 Network Architecture

3.2.1 Overview

The AQUILA's view of the transmission infrastructure is divided into access network and core network, where AQUILA only covers the core network infrastructure. The access and core networks can utilise different architectures. For example, in the access network, where scalability issues are not of the main concern, the IntServ network architecture can be used. The core network architecture is solely based on the DiffServ architecture as this solution provides for network scalability in terms of network size and capacity. The AQUILA architecture distinguishes four types of network elements: Hosts, Edge Devices, Border Routers and Core Routers. The overall network scenario assumes that the user terminal (Host) is connected through the access network to the edge router (Edge Device) that provides access to the core network (see Figure 3-1) while Border Routers provide the access to other IP networks.

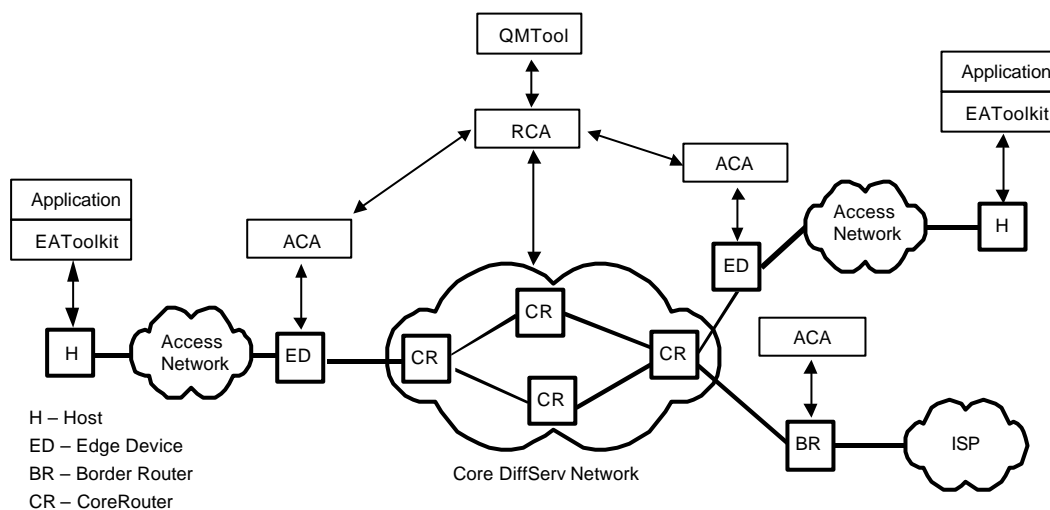


Figure 3-1: General AQUILA Network Architecture

3.2.2 Access Network Architecture

The access network connects user terminals (hosts) with edge devices (ED). It has to provide adequate level of performance not to nullify the quality of AQUILA network services. The solutions that provide point-to-point resource reservation for packet streams between a host and the associated edge device should be preferred.

To provide the required quality of service in an access network the following approaches can be considered:

- The QoS guarantees are provided by layer 3 mechanisms e.g. IntServ, DiffServ, MPLS
- The QoS guarantees are provided by layer 2 mechanisms e.g. ATM, Frame Relay

- The QoS guarantees are provided by over-dimensioning e.g. dedicated Ethernet connections, Fast Ethernet etc.

However note, that AQUILA does not cover QoS in the access network.

3.2.3 Core network architecture

The DiffServ is one among the many propositions to introduce the QoS into the IP networks. It provides scalable service differentiation in IP networks. *The service defines some significant service characteristics of packet transmission in one direction across a set of one or more paths within a network* [RFC2475]. These characteristics can be specified in quantitative (or statistical) or relative terms. The quantitative terms can relate to such parameters as throughput, delay, jitter or packet loss. Relative terms can describe the relative priority in access to network resources of IP packets using different services. However the service definition is not the part of DiffServ network specification. Instead this architecture provides rather the framework for defining network services.

The main requirement for the AQUILA core network is scalability and reliability. Therefore the DiffServ architecture is adopted for AQUILA project. This model assumes that all per flow processing is performed at the network boundary (in AQUILA terms in the Edge Device). The Edge Device is thus the boundary node in the DiffServ terminology and implements such functions like traffic policing, marking, shaping and dropping. The Core Routers (interior nodes in DiffServ) should not be engaged with complex traffic processing (i.e. per flow processing).

The packet handling in the forwarding path of the Edge Device (DiffServ capable boundary router) is depicted on Figure 3-2. The incoming packet stream is classified by the Multi-field Packet Classifier (MF Classifier) into different packet flows. A flow represents a stream of IP packets that receives QoS guarantees and is therefore an analogue to a connection in the connection oriented packet networks, e.g. ATM. The packet classification function in the Edge Device selects IP packets based on the contents of the IP header and assigns them to different packet streams for further processing (conditioning and scheduling). The classifier module of the Edge Device is configured by the ACA, which supplies it with traffic filters constructed on the basis of user request for QoS services. Exactly one traffic filter is present in the Edge Device for each user request (being accepted by the ACA). The ACA constructs filters using the information contained in request messages send by the EAT.

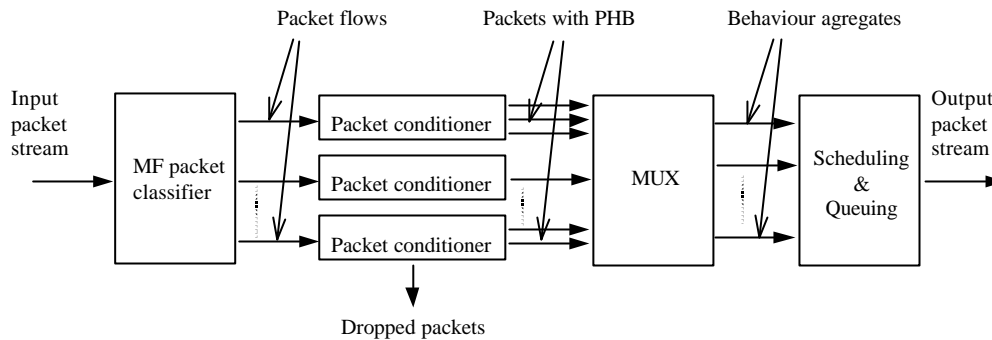


Figure 3-2: Packets forwarding in Edge Device

After passing the classification step, the packet flows are steered to traffic conditioning blocks (TCB). Each TCB block is configured according to the traffic class of a given flow. The TCB ensures that the packet stream entering the AQUILA network conforms to the traffic contract. Traffic conditioners are typically deployed at the DS boundary node (in context of AQUILA architecture this function is deployed in the edge device and the border router). The traffic conditioner may mark/re-mark packets according to the traffic contract and discard or shape packets to change the properties of the packet flow. The traffic conditioner may consist of the following elements: meter, marker, shaper and dropper.

The following TCB blocks are required in the AQUILA network architecture:

- Premium CBR – single token bucket with dropper
- Premium VBR – dual token bucket with dropper
- Premium Multimedia – single token bucket with marker
- Premium Mission Critical – dual token bucket with marker

The conditioner is instantiated and configured by the ACA for each accepted user request.

The traffic stream leaving the conditioner is marked with a specific code point or code points (DSCP). The packets marked with the same code point (independently of the original flow) are merged forming the so-called Behaviour Aggregate. Each Behaviour Aggregate is associated with the specific PHB (per hop behaviour) that determines the forwarding treatment of its packets. Thus packets with the same DSCP will receive the same treatment in the network. The PHB are meaningful in relation to other PHBs (the performance of a single PHB depends only on the load of the link). The PHBs are implemented by means of scheduling and buffer management algorithms.

The packet handling in the forwarding path of the Core Router is depicted on Figure 3-3. The per-flow classification and conditioning functions are not present inside the network. So these mechanisms are not required in Core Routers. The incoming traffic stream is classified by a BA classifier (on the basis of the DSCP field) into flow aggregates that correspond to one of the four AQUILA

traffic classes. The scheduling and queuing block implements buffer management and service disciplines required for each traffic class (this functional block is the same as in the Edge Device).

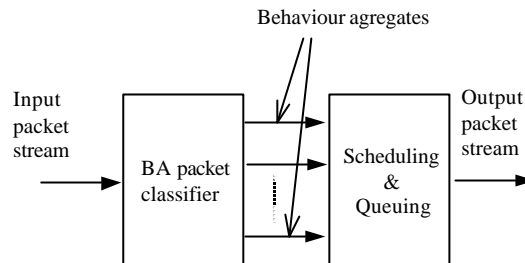


Figure 3-3: Packets forwarding in Core Router

The model of the scheduling and queuing block in the edge and core router is described in [D1302].

3.2.4 Network Topology

3.2.4.1 Access network topology

The access to the AQUILA network should not be limited to any specific solution. Therefore various technologies in the access have to be supported. Such a situation takes place in existing networks, where subscribers are connected by LANs, dedicated ISDN links, Frame Relay etc.

At the lowest level of network hierarchy usually the tree topology is used. The protection against failures at the access using multi-homing is used only in specific cases. The most important reason for that are the significant costs of redundant links. Since the failure of an access link affects only one subscriber, it is his decision whether to protect the access or not and how the protection is realised.

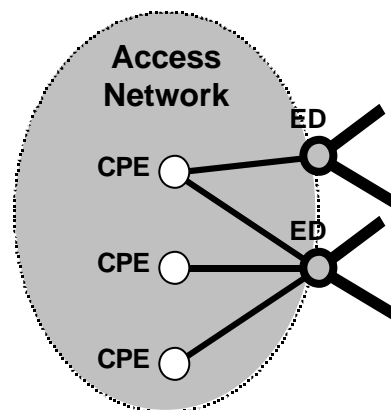


Figure 3-4: Typical access network topology.

3.2.4.2 Core Network Topology

The core part of AQUILA network could be decomposed into some sub-areas following the RCL architecture (see chapter 4.1.2). These sub-areas correspond to the concept of resource pools managed by the RCA. Notice, that each sub-area should also handle its local traffic without using resources allocated to the higher levels.

The highest level is the backbone network, which connects all the regional networks thus offering full connectivity between all network subscribers.

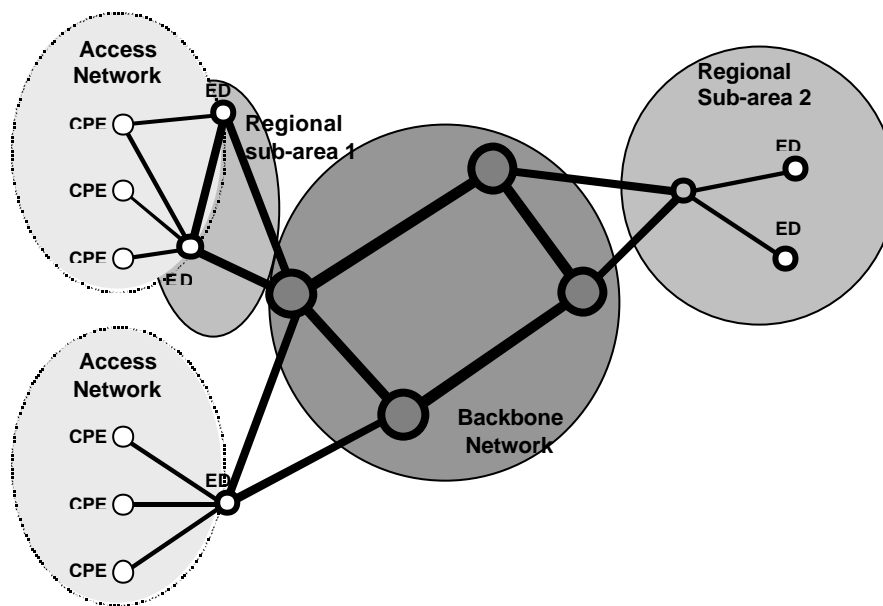


Figure 3-5: Example of network topology.

The core network consists of core routers connecting edge devices. Since the network has to provide full connectivity and reliability, the topologies used at these levels have to be characterised by at least 2-connectivity. The division of core network into sub-areas is related to the existence of resource pools created by the RCL, but also covers other aspects such as geographical location, routing hierarchy etc.

In the case of the core network, tree-like structures are not recommended because they do not provide the necessary reliability. The general candidates for this part of network are ring or mesh structures.

The ring is the simplest topology offering 2-connectivity, i.e. there are always two candidates for forwarding path. In case of a single link or node failure the topology still allows full connectivity. For that reason ring structures are widely used in data-link technologies such as FDDI or SDH. Unfortunately, IP does not directly benefit from the ring configuration – dynamic routing protocols perform the same topology discovery process without any assumption on underlying topology.

The mesh topology is typically used in IP networks. In general, the mesh structures can offer K -connectivity (where $K \geq 2$) and are very flexible in the realisation of network protection.

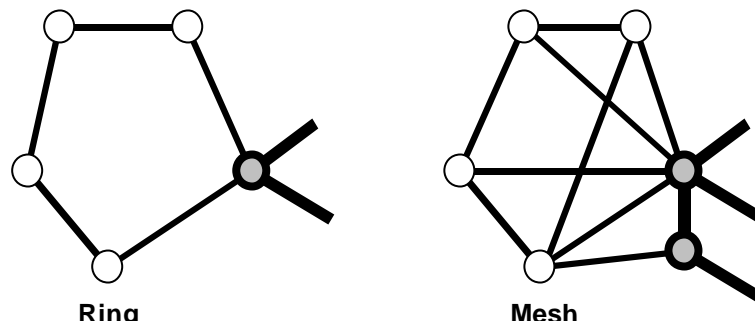


Figure 3-6: Examples of core network structures.

3.2.4.3 Low bandwidth links

At present, access to the IP core network is often provided by low-bandwidth links, using such technologies as ISDN, Frame Relay or synchronous private lines. Therefore it is inevitable, that the AQUILA network architecture should support such technologies. Low-bandwidth links (ranging from 64 to 1024 kbps) are always the problem in providing QoS guarantees, especially in case of interactive traffic, which is susceptible to increased latency. When the traffic generated in the subscriber's network is composed of large packets created by e.g. FTP applications, large queuing time of such packets could strongly influence delay-sensitive traffic generated by applications such as Voice-over-IP, Telnet etc. For that reason two additional mechanisms are used in order to solve delay problems:

- Link fragmentation and interleaving
- Header compression

Link fragmentation and interleaving is a method of splitting, sequencing and recombining large packet across low-bandwidth links. Arriving packets are classified and sorted into queues. Next, the large packets are fragmented to packets of small size and interleaved with time-sensitive traffic by an appropriate queuing discipline. Fragmenting and reassembling of the packets is performed using a special encapsulation based on the Multilink PPP protocol.

Header compression reduces of number of bytes needed to carry IP, UDP/TCP and/or RTP headers by eliminating information that is not changing in each packet belonging to a particular flow and thus reducing the bandwidth needed to carry such packets. In case of RTP traffic, header compression allows to reduce 40 bytes constituting the IP, UDP and RTP headers to only 5 bytes. The decompressor located after the link is then able to reconstruct the original headers without any loss of information.

The existence of low-bandwidth access links has some serious impact on the AQUILA network architecture. First, the admission control should be performed before such links, because there is a high probability, that it will be the bottleneck on the forwarding path of the flow. Recall, that usually we cannot get significant multiplexing gain on low-bandwidth links with adequate QoS guarantees.

For that reason, a two-stage approach is implemented. At the first stage, admission control should be roughly performed at the customer access router (ED) in order not to exceed the bandwidth of the access link (policy and QoS check). The access router should be suitably configured in order to prioritise QoS traffic. The actual admission control decision is taken by the first core router.

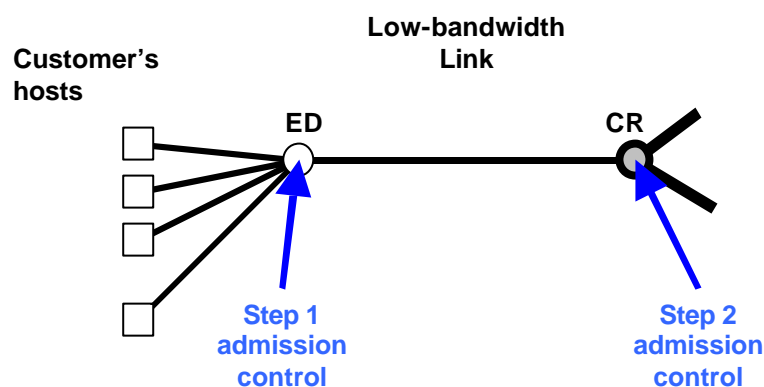


Figure 3-7: Access to the core network using low-bandwidth links

3.2.4.4 Topological Aspects of Hierarchical Resource Distribution

The topology of the network is strongly related to the way, how resource pools are created and configured in the resource control agent (see chapter 4.1.2).

Taking into account hierarchical distribution of resources performed by RCA the most important question is when and where we create resource pools. One can expect that knowledge about network topology and routing should be helpful in solving this problem. First of all, we must assume some mapping between logical resource pool elements and elements of physical network. The following assumptions can be made:

- Resource pool maps into the set of physical links,
- Each link can belong only to one resource pool,
- ACA resource pool is mapped to ED (indirectly also to physical links connecting ED with the core, since there is a relationship between AC limit given and link bandwidth).

The following rules determine the creation of sub-areas:

- They should contain at least two EDs,

- The sub-areas' sets of links must be consistent.
- The sub-area should close its local traffic – traffic directed from one member ED to another member of the same pool should not cross the links that do not belong to the given pool.

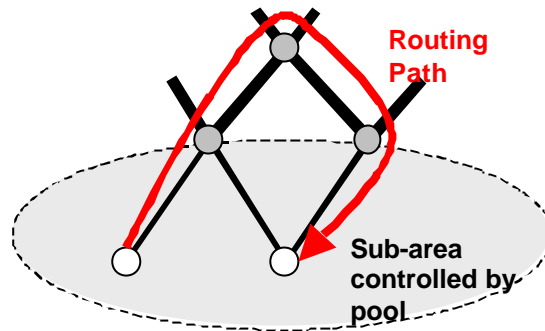


Figure 3-8: Example of disadvantageous situation when local traffic influences other pools

- The sub-areas of the same level of hierarchy should not be directly linked to avoid “stealing” resources one from another (see figure below). Such regions should create a single sub-area.

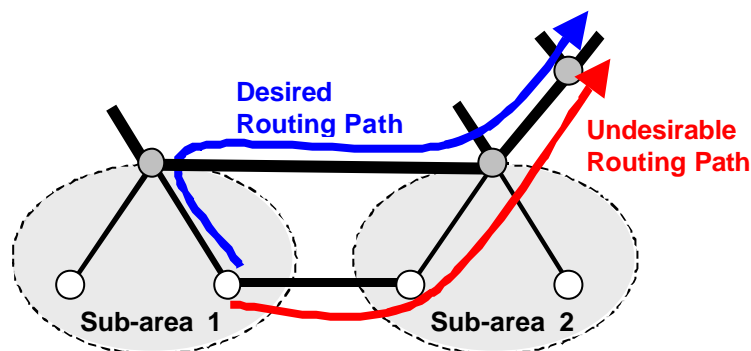


Figure 3-9: Example of undesired routing between two directly linked sub-areas.

Summarising the above discussion, we may conclude that the structure of the sub-areas should be organised in the hierarchical form following the resource pools structure. Each sub-area can be directly connected only to the higher or lower level, and as a consequence this leads to the tree-like structure (see Figure 3-10). EDs and BRs constitute the leafs of the tree. The number of levels in this hierarchy can be chosen as needed, depending on the real network topology.

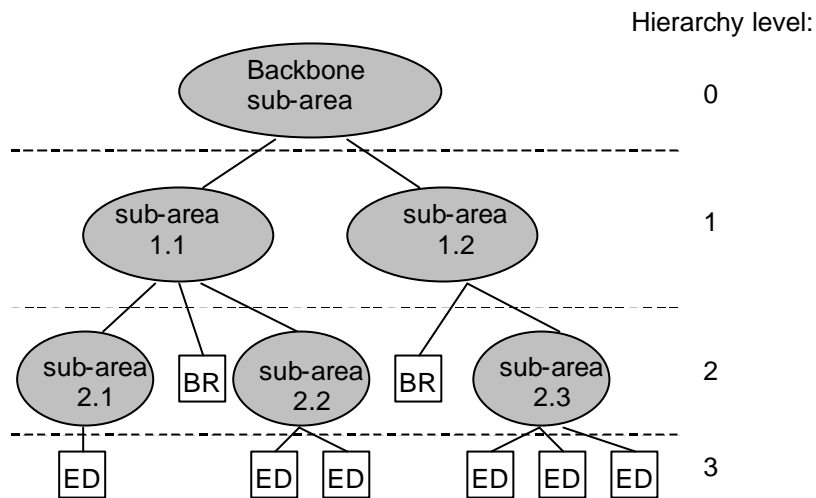


Figure 3-10: Exemplary hierarchical network topology

The ability to structure the network in this manner is unfortunately limited to strict hierarchical configurations. For instance, the full mesh topology, which is not of hierarchical type, does not fit well to such partitioning. Anyway, within a given sub-area the mesh topologies are preferable.

4 Resource Control and Inter-Domain Layer Architecture

4.1 Resource Control Layer

The resource control layer (RCL) is an overlay network on top of the DiffServ core network. The RCL mainly has three tasks, which are assigned to different logical entities:

- To monitor, control and distribute the resources in the network. This task is assigned to the **resource control agent (RCA)**.
- To control access to the network by performing policy control and admission control. This task is assigned to **admission control agents (ACA)**. Each edge router or border router is controlled by an ACA. As each access request necessarily means usage of resources, the RCA may be directly or indirectly involved in handling admission requests.
- To offer an interface of this QoS infrastructure to applications. This task is assigned to the **end-user application toolkit (EAT)**. From the network point of view the EAT acts as a RCL front-end. From the user point of view, the EAT provides a QoS portal.

The entities defined above are associated to network elements within the underlying domain as shown in the following figure:

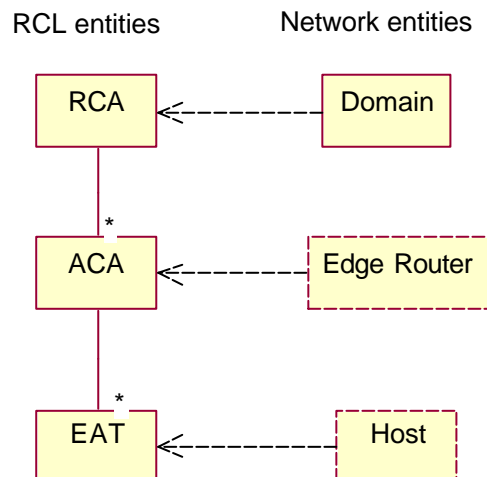


Figure 4-1: Mapping of RCL entities to the underlying network entities

Please note, that an EAT instance can be responsible for a single host as well as for a set of hosts. The latter might be the case, when not a single host, but a whole sub-network is connected to an edge router.

The resource control layer assumes an underlying DiffServ network. The DiffServ code points (DSCP) and the PHBs of this network are assumed to be defined by management. They are installed in the network elements by the RCL. For each traffic class (see chapter 3.1.4), there is a specific amount of bandwidth available in each link of each edge router, border router or core router. So bandwidth is the main resource, which is handled by the RCL.

4.1.1 Admission control

A DiffServ network can only provide quality of service, if it is accompanied by an admission control, which limits the amount of traffic in each DiffServ class. The admission control algorithms implemented in the RCL layer (by ACA agents) accept or reject flows on the basis of information provided by the End-user-Application-Toolkit (EAT) (for legacy applications) or by the application itself (for QoS-aware applications). The ACA agent uses this information to decide whether enough resources for the connection request are available in the network (in hierarchical Resource Pools structure). In AQUILA, the stochastic nature of the packet traffic is described in the form of deterministic parameters that correspond to a token bucket algorithm. Moreover, the admission decision is based on the parameters, which are specified before any real user traffic is sent to the network. The AC methods based on token bucket description are usually rather conservative since they consider the worst-case traffic pattern (in practice, one can observe, that the real traffic is quite far from that assumed for AC). As a result, these methods do not guarantee effective network resource utilisation. The resulting network utilisation can be improved by using feedback from the network.

This chapter specifies the system architecture for implementing measurement procedures for supporting RCL tasks, mainly the admission control. No specific algorithms for these tasks are presented below; instead a general measurement architecture that can support these algorithms is described. The actual algorithms for realising control loops in RCL are proposed by WP1.3 and specified in [D1302].

4.1.1.1 AC approach

The AQUILA architecture uses a **local admission control** located in the ACA, which is associated with the ingress and egress edge router or border router. To enable the ACA to answer the admission control question without interaction with a central instance, the RCA will locate objects representing some share of the network resources nearby the ACA. Resources are assigned to these objects proactively. For the ACA, these objects represent a “consumable ResourceShare”. Admission control can be performed either at the ingress or at the egress or at both, depending on the reservation style.

Two AC approaches are implemented, i.e. Declaration-Based (DBAC) and Measurement-Based Admission Control (MBAC) methods. Network operator has the possibility to choose between the MBAC or DBAC methods. In case of DBAC approach we can further distinguish the Peak Rate Allocation (PRA) and Effective Bandwidth Allocation (EBA) methods.

Considering AQUILA architecture we can distinguish the following areas where the AC algorithms will be used:

- Primary access link
- Core network
- Secondary access link
- Inter-domain link

4.1.1.2 Admission control loop architecture (MBAC)

The concept of control loop for admission control is depicted in Figure 4-2. In order to realise an AC control loop, a supporting measurement architecture was defined.

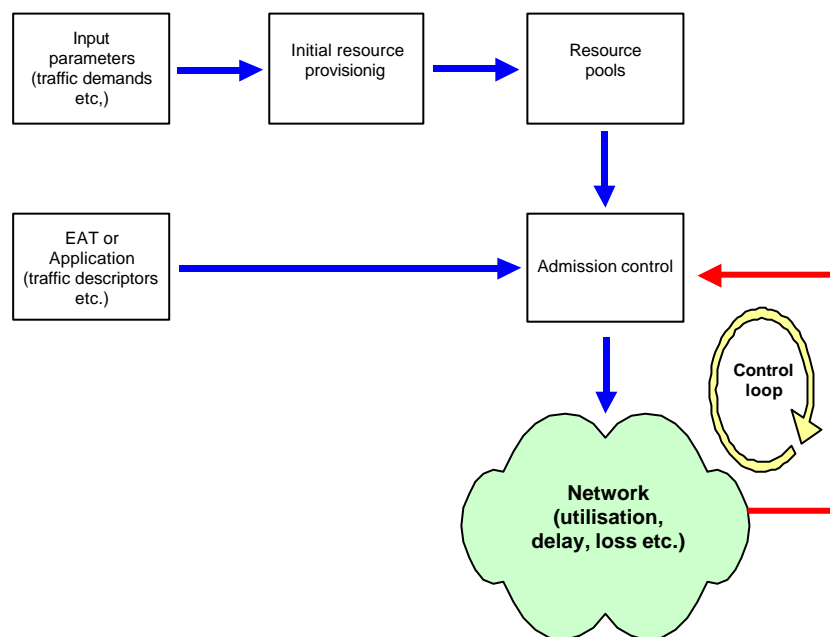


Figure 4-2: Resource control (admission control loop).

Most of the measurement-based admission control algorithms require the measurement of the offered traffic rate. In case of the AQUILA architecture this parameter has to be measured on per traffic class and per ED link basis and per ingress and egress direction (in case of p2p reservations scheme). For this purpose a passive monitoring architecture seems to be the most convenient solution.

The admission control loop architecture is depicted in Figure 4-3. The following design assumes that elements, which implement MBAC functions, are integral parts of the ACA agent. The MBAC consists of one **MbacCollector**, which is responsible for retrieving and processing data from the router

and an **AdmissionControl** entity, which implement the AC algorithms. The MbacCollector regularly polls the router statistics for the offered traffic (number of bytes) on the link between its Edge Device and first Core Routers in each traffic class. On the basis of these parameters it calculates the parameters required by the given admission algorithm i.e. estimation of mean traffic load in each class.

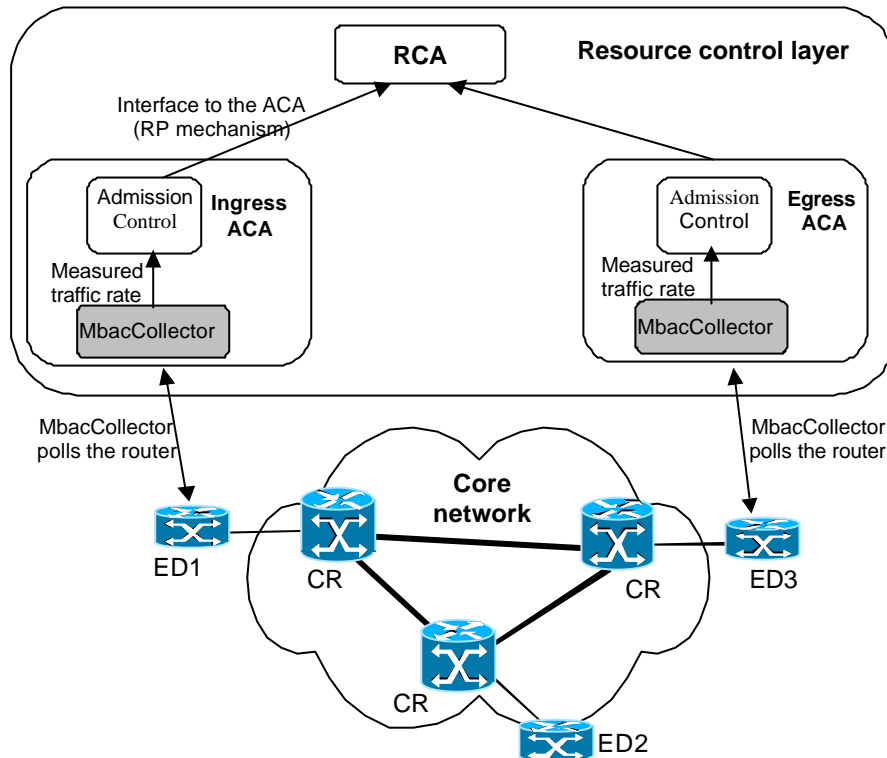


Figure 4-3. Measurement architecture for admission control loops

The role of the MbacCollector is to measure the traffic rate offered to the link controlled by a given ACA. It separately measures traffic for ingress and egress direction. The egress traffic is measured, if the traffic class uses p2p reservation. An example of an ED, which has to be monitored on the ingress and egress side, is presented in Figure 4-4.

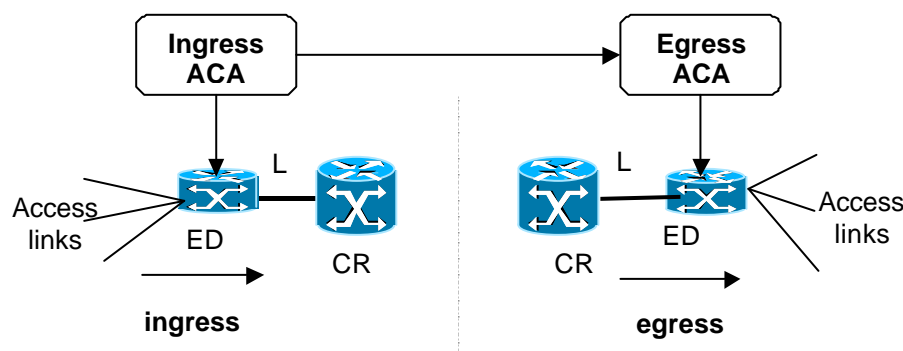


Figure 4-4. Link monitored by the MbacCollector on the ingress and egress side of the ED

At the ingress side, the MbacCollector should monitor the traffic sent from the ED to the CR. Therefore, the MbacCollector reads the statistics from the outbound interface of the link L of the ED. The data has to be obtained from the scheduler statistics on a given interface. In this mode, the transmitted and dropped packets (bytes) should be taken into account. For the egress direction, the MbacCollector should monitor the traffic sent across the link L from the CR to the ED. The statistics are read from the inbound interface of the ED.

Every specified measurement period, the MbacCollector updates the rate samples $X_i(t)$ for each TCL ($i=1\dots4$) in ingress and egress direction (where appropriate). The value of the measurement period duration is a configurable parameter. Based on the collected samples, the MbacCollector calculates the estimated mean rate of offered traffic M_i^{est} , using the window-based mean estimation algorithm.

The AdmissionControl object makes the actual admission decision by implementing a respective admission control algorithm. The operator has the possibility to specify, whether the admission decisions in a given ED and in a given traffic class should be performed using MBAC (measurement-based admission control) or DBAC (declaration-based admission control).

4.1.2 Resource distribution

Resource distribution is performed on a per DiffServ class basis. There is no dynamic reconfiguration of DiffServ classes. So, the resources of each class can be handled separately and independently of each other. This per class distribution however is not appropriate for edge devices, which are connected via small bandwidth links to the core network. In this case, additional mechanisms apply, which are described in chapter 4.1.3.

Resources are handled separately for incoming traffic (ingress) and for outgoing traffic (egress). The following description of resource distribution applies to both.

Resource distribution is performed by the RCA in a hierarchical manner using so called **resource pools**. For this purpose it is assumed, that the DiffServ domain is structured into a backbone network, which interconnects several sub-areas. Each sub-area injects traffic only at a few points into the backbone network. As described later, this structuring may be repeated on several levels of hierarchy.

When considering the resources in the backbone network, all traffic coming from or going to one sub-area can be handled together. So it is reasonable to assign a specific amount of bandwidth (incoming and outgoing separately) to each sub-area.

Depending on the topology of the backbone network, it may be useful to add some degree of dynamic to this distribution. The RCA may provoke a re-distribution of resources, based on real traffic demands. Therefore, it can assign a larger amount of bandwidth to one specific sub-area, when the bandwidth is reduced in other sub-areas.

Moreover, within a sub-area, there may be further subordinated sub-areas, which could be handled similar. Each resource share r_i assigned to a sub-area can be handled again as a resource pool R , which is distributed in a similar way among the sub-areas. Finally, resources can be used by ACAs as “consumable ResourceShare”.

The depth of this hierarchical structure may be chosen as needed. It is also possible to mix several degrees of hierarchy, e.g. to break down the structure near edge routers more deeply than the structure of border routers, which are likely to be directly connected to the backbone.

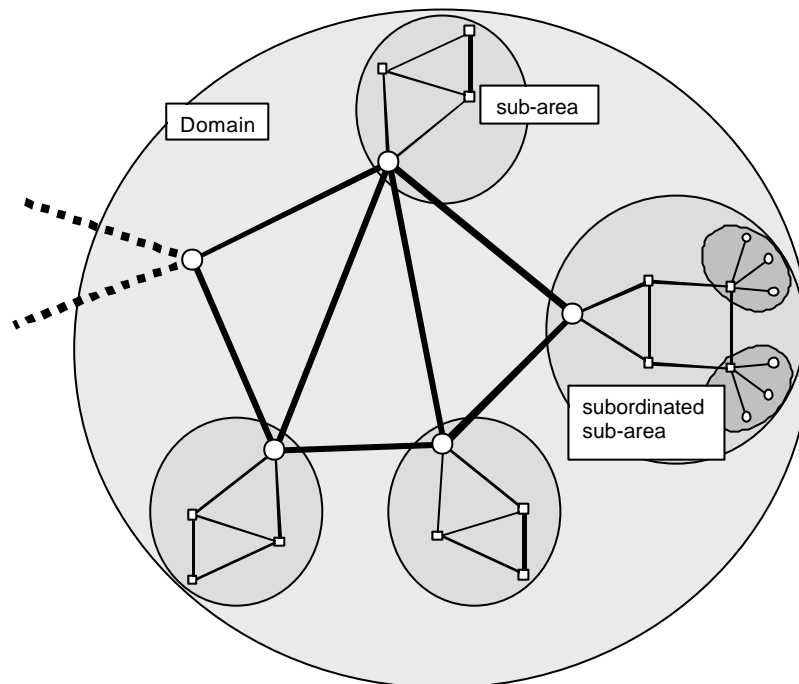


Figure 4-5: Hierarchical resource pools

The figure above illustrates this. It shows an example domain, which contains four sub-areas and one border router. In one of the sub-areas, the further division into subordinated sub-areas is illustrated.

Obviously, the ability to structure a domain like this strongly depends on the topology. In the access area of a network however it is likely, that tree-like structures exist, which enable the definition of such a structure. This structure is called **Resource Pool Tree (RPT)** and reflects the hierarchical concept of the creation of the resource pools.

4.1.2.1 Resource Pool Tree

After creating the RPT, the distribution of resources is composed of two basic phases: initial configuration and re-distribution of resources. Therefore, initially resources are assigned to each resource pool according to traffic loads forecasts.

The term resource, as far as the resource distribution mechanism is concerned, comprises a single parameter, namely bandwidth. Each resource pool is composed of a number of Resource Shares, each one managing the resources of a traffic class (TCL). Therefore, each resource pool has maximum 8 Resource Shares, per traffic class and direction (ingress and egress). The distribution of resources is performed in a top-down approach, where the root of the tree, which corresponds to the resources for the overall network, distributes the resources to its children, and the children distribute the resources to their corresponding children. The resources are actually used by the lower ACAs, which are characterised by the Consumable Resource Shares (CRS). In fact, a resource pool with somehow different functionality is realised also in each ACA, composed of the CRSs.

An alternative approach would be to assign zero initial resources to all the resource pools, apart from the root of the RPT, which would have all the available network resources. The distribution of resources down to the tree hierarchy would be then based on the real traffic load, increasing though the interactions between the resource pools.

Since, the initial distribution of resources is based on some static mechanism, those resources may be re-distributed based on the real traffic loads. The introduction of the resource pool algorithms encounters the dynamic distribution of resources between the resource pools focusing to a better network resources utilisation. A description of those algorithms can be found in deliverable [D1302]. Each RP needs some configuration parameters, including:

- R available resources to be shared by the RPs of the lower level
- R_i : limit for resource assignment to lower RP i
- r_i : current resource assignment to RP i , with $r_i \leq R_i$ and $\sum_i r_i \leq R$
- r_i^{free} : assigned but currently unused BW of RP i ($0 \leq r_i^{free} \leq r_i$)

There is an advantage to use a RP if $\sum_i R_i \geq R$ only. In this sense a RP shares the bottleneck bandwidth of a link.

4.1.2.2 Dynamic Re-Distribution

The dynamic re-distribution of resources between the RPs is based on some general rules and it is invoked when an ACA cannot accommodate a reservation request and its AC limit should be increased. Then, the ACA issues a request to its corresponding RP. The interactions between the RPs entities of the tree are depicted in Figure 4-6. Our purpose is to provide a complete separation between the AC algorithms performed by the ACAs and the resource pools algorithms (which aim at the re-distribution of the assigned resources) realised in the RPT. In fact, each ACA is responsible for performing AC functions and determining whether or not there are available resources for admitting a new flow based on its current assigned AC Limit. In case, a new resource request can not be accommodated based on the current assigned AC Limit, the ACA will request from the RCA additional resources, changing in this way its AC Limit, in order to accept the new request.

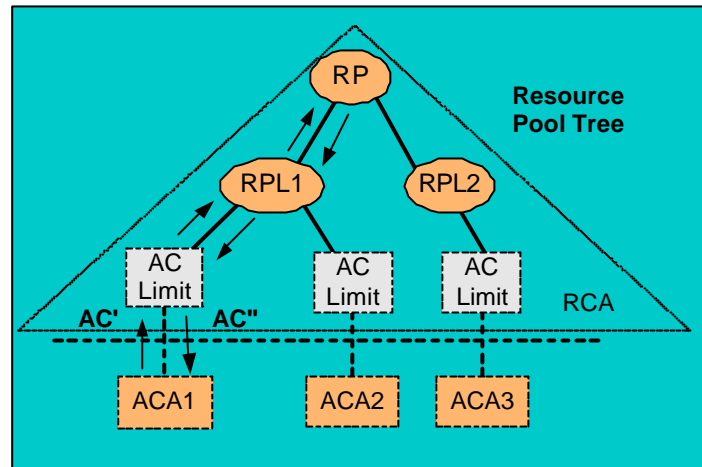


Figure 4-6: Interactions between ACA and RCA

The RCA receives the request for the new AC Limit (AC') and calculates the new value of the AC limit based on its resources and the implemented resource pool algorithm. If a RP does not have enough resources to accommodate the request, then additional resources are requested from the RP of the level above. Each RP runs the same algorithm, which is executed whenever resources should be re-distributed. The actual amount of assigned resources (AC'') from a RP is based on its free resources as well as the implemented algorithm. The request for additional resources may be propagated up to the root of the tree, as depicted in Figure 4-6.

Moreover, in case resources are not used by an ACA, they are released to the upper level RP. The amount of released resources also depends on the implemented resource pool algorithm. In the sequence the RP, it can also release unused resources to the upper level RP, or give them to another ACA, which is in need of resources. In this way, re-distribution of resources is achieved.

4.1.2.3 Inter-Domain Resource Distribution

In case of the interconnection of different domains, each domain will manage its own resources and will distribute them to its RPs. Each domain will also contain a number of Border Routers (BR), which will connect the domain to its neighbourhood domains. The BR will be handled like an ED. For the assignment of resources to the BR, only the connection of the BR to its own domain is taken into account.

In fact, the control is focused on the resources that a BR sends into or gets out of its own domain. As far as the control of the inter-domain link is concerned, this is examined in section 4.2, where inter-domain resource allocation issues are covered.

4.1.3 Small Bandwidth Links

In typical network scenarios there might be edge devices, which are connected to the core network by links with a relatively small bandwidth (e.g. 256 kbit/s). In this case, the resource distribution scheme described above must be slightly modified.

It is not reasonable to split this capacity into different traffic classes, as it is done with larger bandwidth links in the network. Instead, there should be a mechanism, which assigns the bandwidth to those traffic classes, which request them.

Consider the following example network:

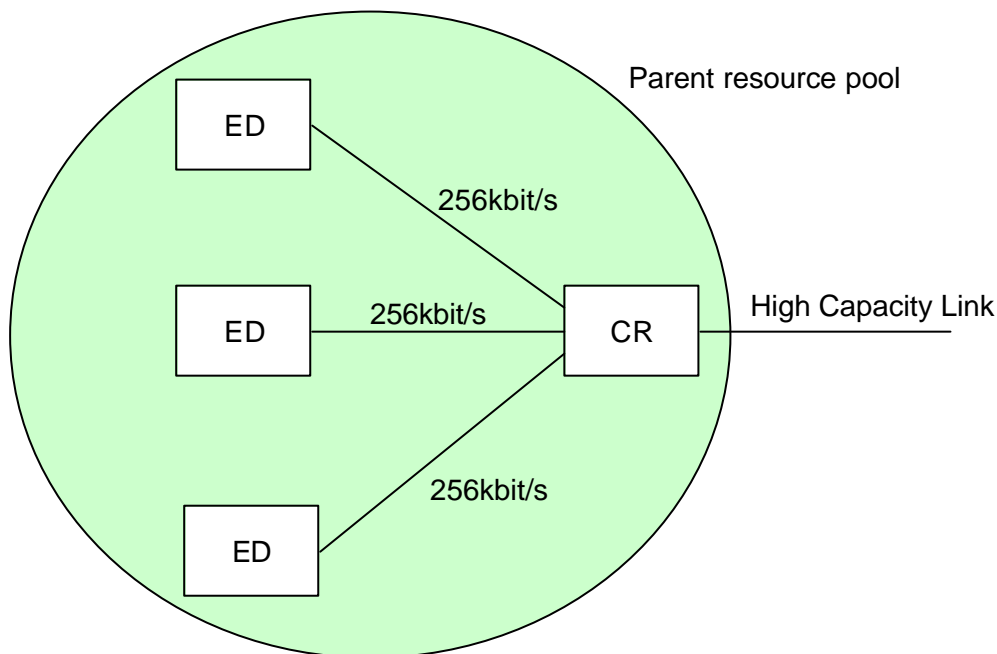


Figure 4-7: Example network with small bandwidth links

The parent resource pool covers the area of the network shown above. The children correspond to the resource shares assigned to the edge devices (respectively the ACAs related to the edge devices).

During initial assignment of resources, the parent will assign empty resource shares to the children, which have an upper bandwidth limit of zero. This is done, because any nonzero resource assignment would split the low bandwidth of the links into even smaller pieces, which may be unusable at all.

When an ACA gets a request, it will try to allocate resources from its associated resource share. According to the assignment described above, this request will not be successful at the first step. Instead, the child will try to increase its resource share to fulfil the request and will in turn request the appropriate amount of resources from its parent.

If the parent is able to fulfil the request, it will give the requested resources to the child, which then can “consume” them and admit the user request.

Likewise the child will return any resources released by the user to the parent. The child’s bandwidth cushion will in fact always be zero.

Up to now, the behaviour described here is nothing new. It is just the general resource pool algorithm. However some parameter values take on specific values (e.g. zero initial resources, zero bandwidth cushion).

To avoid unnecessary resource requests to the parent, the child should control and limit the sum of the requested resources for all traffic classes and reject a request, if it exceeds a certain limit, which depends on the bandwidth of the link to the core router. This is an additional functionality, which can especially support small bandwidth links.

4.1.4 Deployment of logical entities

The previous chapters describe the RCL mainly on a logical level. For an implementation, however, a mapping of these logical entities to physical components has to be defined. Two opposing examples will show, how broad the possible range of mappings can be. For the AQUILA trial, it is likely that none of these extreme approaches is used. A good mixture of both will be the most suited mapping.

4.1.4.1 Mapping to associated physical entities

As defined in the foreword of chapter 4.1, there is a somehow natural mapping of logical to physical entities:

- The EAT is associated with the end-user host
- The ACA is associated with the edge router or border router
- The RCA is a more abstract instance, not associated to a specific network element.

This associations suggest, that

- The EAT is running close to the user’s host.
- The ACA is running on the edge router/border router or on a host closely related to this router.
- The RCA is running on a separate platform.

This is however not the only possible mapping. In a fully different approach, one may also define a single RCL platform, as described below.

4.1.4.2 Single RCL platform

The complete opposite of the previous approach is the single RCL platform approach. Here, all logical entities of the RCL are mapped to a single RCL host. This host runs all the EATs, the ACAs and the RCA.

End-user hosts communicate with their EATs e.g. by using CORBA, RMI, RSVP, HTTP or another protocol. The location of each component is fully transparent to their clients.

The architecture described in this document allows the full range of mappings as shown above.

4.1.5 Roles

In a general QoS scenario, three different roles of actors can be defined:

- **Requester:** The requester sends the QoS request to the network. He/she determines, which service will be requested from the network and who may use it. The requester has to be authenticated to the network, because he/she will be charged for of the reservation. In AQUILA, the role of the requester is always played by the EAT.
- **Sender:** The sender injects the QoS traffic into the network. Admission control has to be performed for the sender. To control the injected QoS traffic, a policer and a marker has to be assigned.
- **Receiver:** The QoS traffic leaves the network at the receiver side. A reservation of network resources is also performed at the receiver side. A policer, however, is not necessary there.

Depending on the scenario, the sender and/or receiver is not always known. The requester role, however, must always be present. So, the following **reservation styles** may be implemented by this approach:

- **p2p, point to point.** The sender and the receiver are known.
- **p2m, point to multipoint.** The sender and a set of receivers are known.
- **p2a, point to anywhere.** Only the sender is known. QoS data may be sent to any destination.
- **a2p, anywhere to point.** Only the receiver is known. Traffic from any sender is prioritised.

Also depending on the scenario, the requester and the sender or the requester and the receiver may be located in the same host. This generates the following **reservation modes**:

- **Sender oriented.** The requester is identical to the sender.
- **Receiver oriented.** The requester is identical to the receiver.

- Third party. The requester is neither the sender nor the receiver, but a third party.

Reservation modes and reservation styles are independent of the requested network service. The admission control may however restrict the possible combinations of network service class and reservation style. E.g. a network service class providing guaranteed services should be restricted to p2p reservations.

4.1.6 Reservation groups

The establishment of some services needs a multiple reservation in the network. A good example is a multi-conference where a collection of reservation requests is necessary. To support this requirement an extended concept of reservation, called reservation groups, is included in the AQUILA structure. The idea of reservation groups is to associate several requests and to send them in a single message with all the reservation information. This feature may also be useful in order to support TCP traffic because the QoS for ACK packages in the backward direction influences the forward data stream.

The request association is done at the EAT where all the information about the individual reservations is known. This reservation group request is sent to the Manager ACA, which extracts the information needed for requesting the individual reservations. When the service finishes the Requester EAT release all the connections corresponding to the reservation group.

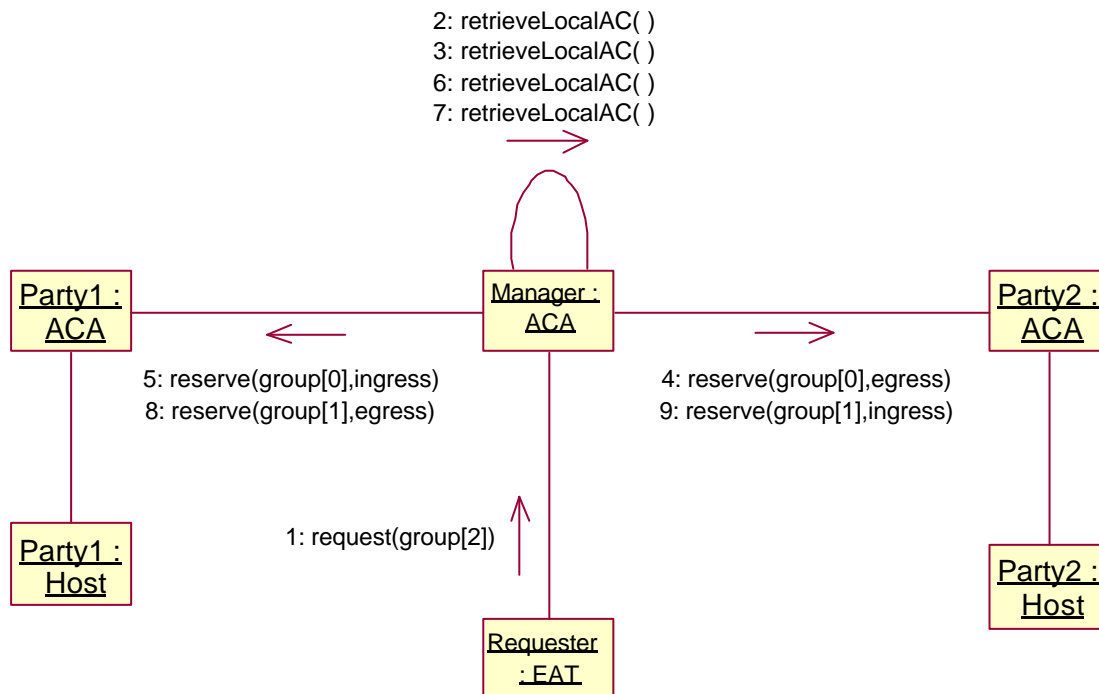


Figure 4-8: Bi-directional reservation request using Reservation Groups

In the above figure an example is shown for a bi-directional reservation. The Requester EAT creates a list of connections (*group*[]). Every element of *group*[] contains the information needed for making a reservation.

For reservation groups, we distinguish between two different species:

- a **reservation bundle** contains reservations, which form an atomic aggregate. An example may be the forward and backward direction of a TCP reservation or a bi-directional voice call. The bundle can only be set-up and released as a whole. If one reservation of a bundle fails, the whole bundle is released.
- a **reservation set** contains several reservation bundles, which are logically related. An example may be the reservations necessary to set-up a conference call. Reservation bundles may be added or removed from a set during lifetime.

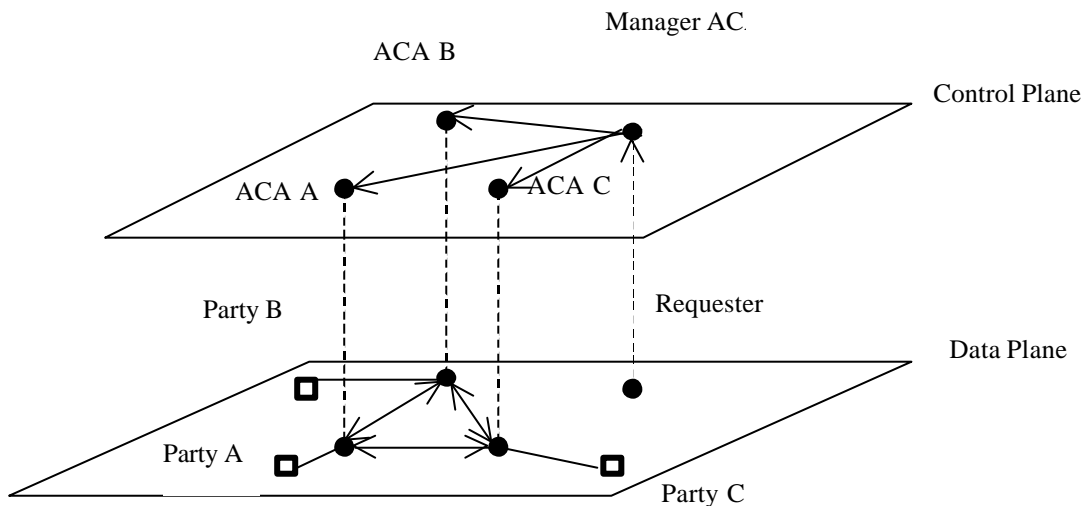


Figure 4-9: Example of reservation groups in a conference scenario

Due to the presented schema a service using reservation groups is thought as a group of individual connections, and is far away from a centralised idea, where all the users are just connected to a distributor. The Requester EAT and the Manager ACA are just administrating the reservations, they are just working within the control plane (RCL). In the same way, when a new reservation has to be added to the reservation set, the Requester EAT is the only one, which can generate the “*join*” message. Returning to the conference example, the figure below shows how a new user (Party D) joins the reservation set. The join message cannot be sent by the new user. The message has to be sent by the Requester EAT, which has all the information about the reservation set and is the one that will be charged by the reservation. The Manager ACA has to send reservation requests to all the ACA implied in the conference in order to connect the bi-directional communication Party D with all other conference participants.

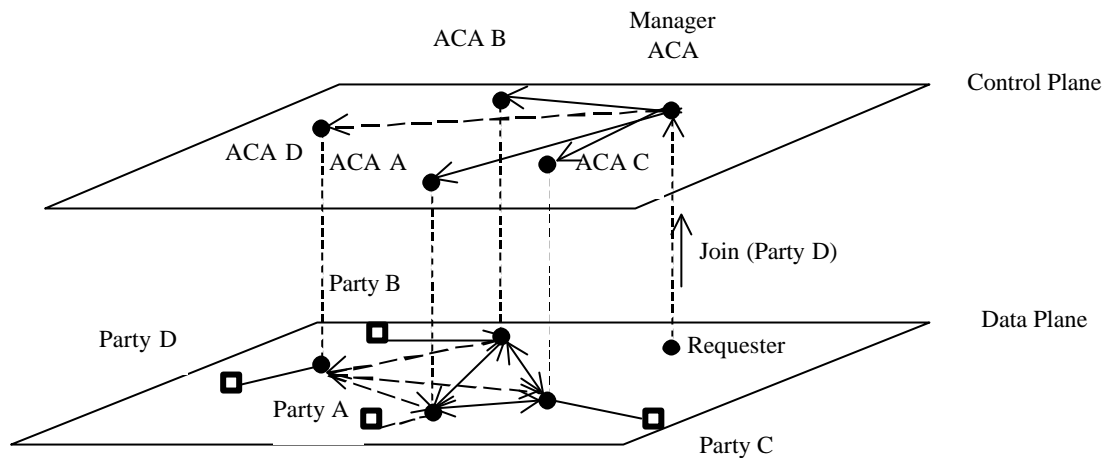


Figure 4-10: Example of join message in a conference scenario.

4.1.7 Prioritised signalling and control traffic

The AQUILA architecture is constructed as an overlay to a DiffServ aware network. The resource control layer -RCL- is a logical layer with connections to the DiffServ net in order to control and to communicate with the QoS environment (DiffServ aware ED/CR/BR). For this inter-working with the DiffServ network, signalling traffic or control messages of the RCL traverses the DiffServ domain (nameable AQUILA domain) up to the controlled entities and vice versa.

It is assumed that this traffic is carried about the same physical links as the data load. In this case the signalling traffic rivals with the prioritised or best effort traffic for the available bandwidth and in an DiffServ based IP network only a prioritisation mechanism has to be developed in order to guarantee that specific traffic flows aren't blocked by the other data flows.

The Figure 4-11 shows the RCL as a logical overlay net with some physical connections to a Diff-Serv aware network. The lines symbolise a link between two entities.

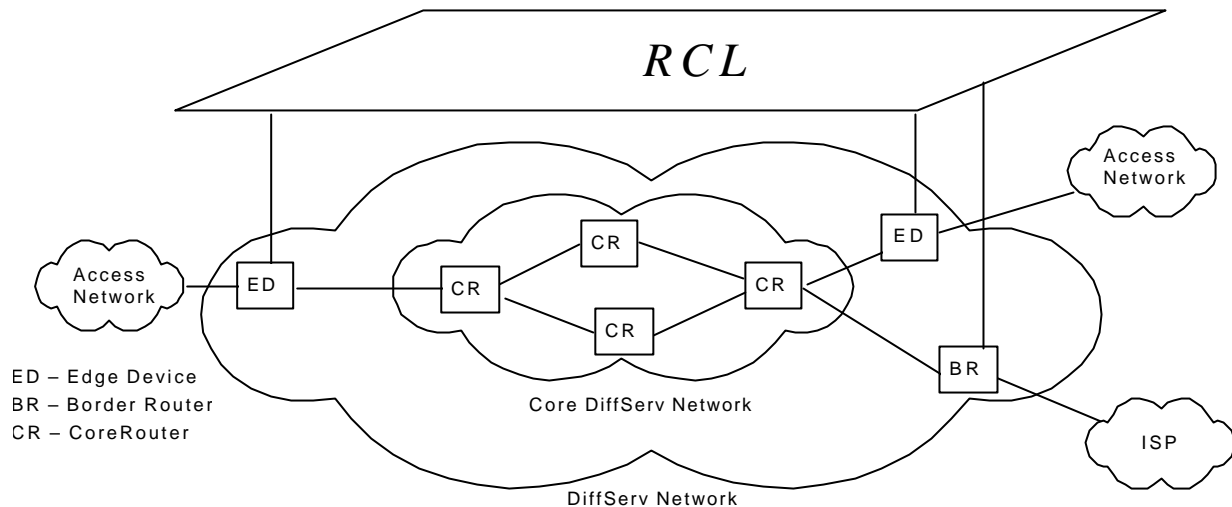


Figure 4-11: AQUILA Resource Control Layer

Due to the AQUILA overlay network, signalling traffic occurs within the RCL, between the user and the RCL, between the RCL and the DiffServ network and between different domains.

Be aware that the RCL is a logical layer realised by software components that are hosted on different machines, which can be located theoretically anywhere. The actual localisation of the components affects some of the following considerations. However we have carefully tried to cover most of the possible scenarios.

4.1.7.1 Marking of the signalling traffic

In IP networks the data load and signalling load use the same links and therefore the signalling traffic has to be prioritised at the ingress point of the AQUILA network or on the host machines on that the control traffic is generated.

One premise is that all the entities within the DiffServ network are trusted and the traffic type generated at these machines has to conform to the requisites of the DiffServ AQUILA network (like valid DSCPs, expected traffic load etc.). Then these actions has to be done

- the signalling traffic inside the RCL has to be marked at each host of a RCL component,
- the signalling traffic from the user to the EAT has to be recognised, policed and marked at the ingress ED including possibly the user's http requests to the QoS portal WEB server
- the signalling traffic between the ED and the ACA must be marked, if the ACA is not directly connected to the ED

- the trusted signalling traffic into the adjacent net has to be marked with a DSCP according to the SLA with the neighbour domain operator

Different solutions for the handling of the marked traffic exist

1. mark the signalling traffic with the same DSCP used for one of the existing traffic classes and mix the signalling traffic with that prioritised type of traffic
2. set-up a new traffic class for the signalling traffic with an own scheduling queue
3. set-up a new DSCP for the signalling traffic but mix this traffic into an other scheduling queue
4. use the same DSCP as the routers use for the router to router signalling protocols
5. or fully mash the RCL components with MPLS paths

The signalling traffic is characterised by the requirements of low bit error rate, low delay, high priority, bursty traffic and possible strong variation of the packet size.

First of all it is necessary to recognise the AQUILA signalling traffic inside the AQUILA domain in order to be able to modify dynamically the policing mechanisms set in the domain regarding this type of traffic. It is also important for the traffic engineering to get information about the AQUILA signalling traffic load and distribution within the domain. The AQUILA measurement tools e.g. have to get the possibility to watch this traffic type. Therefore on the IP layer a typical ToS marker has to be set and therefore a new DSCP should be created.

4.2 Inter-domain resource allocation

Resource control between domains is generally different from resource control within a domain:

- For intra-domain resource control, there may be some entity, which has an overall picture of the domain. In the AQUILA approach, this is the RCA. However for inter-domain resource control, there will be no similar entity, which manages an overview of the whole Internet.
- Intra-domain resource control has to police single flows as they enter the network in order to guarantee QoS for each customer and to block malicious flows. Inter-domain resource control however has to operate on aggregated flows in order to be scalable.
- Within a domain, one can assume a homogeneous network architecture. Intra-domain resource control however has to cope with different kinds of networks in terms of technology, routing, QoS control and scale (ISP or backbone).

These facts require a basically different architecture for inter-domain resource control than it is used within a domain. However, interoperability of resource control at both levels has to be guaranteed.

So the following principles are applied, when selecting an inter-domain resource control architecture for the second phase of AQUILA:

- Inter-domain resource control is performed by an additional layer built on top of the intra-domain resource control. The interface between these two levels of resource control must be standardised and independent of the actual implementation of the intra-domain resource control, so that network operators are free to use any kind of resource control within their domain.
- Inter-domain resource control is performed by a set of independently managed, but co-operating entities, which may be called “bandwidth broker”. A protocol for communication between these entities must be specified and standardised.

The architecture specified in this document is based on these principles. Experiences from the intra-domain resource control are re-used, where applicable.

4.2.1 State of the art

Currently, mainly two approaches are available for inter-domain resource control:

- SIBBS, a protocol specified by the QBone Signalling Design Team in the context of the Internet2 project.
- BGRP, an inter-domain resource control framework developed by P. Pan et al. at the Columbia University, department of computer science.

Both approaches are shortly discussed and rated in the following.

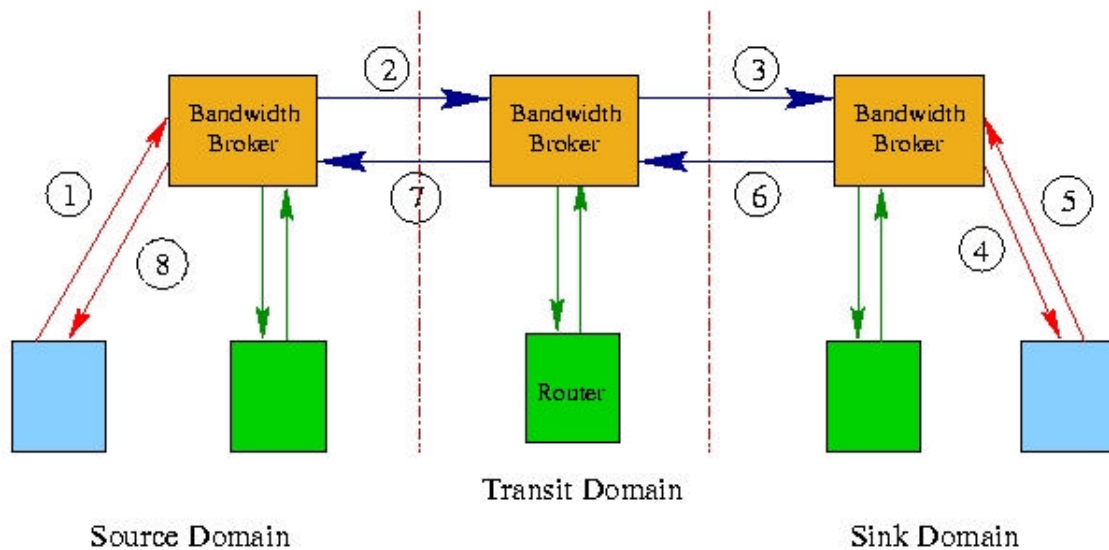
Currently, discussion about QoS signalling is also emerging in a newly created working group in the IETF, called NSIS (Next Steps In Signalling). We also try to consider the AQUILA approach in the light of the currently ongoing – and still rather unstable – discussion there.

4.2.1.1 SIBBS

The QBone bandwidth broker architecture and the SIBBS protocol are specified in [SIBBS]. This is a living document, as the QBone Signalling Design Team is still working on the architecture and the protocol.

The basic idea of SIBBS is to form a single layer of bandwidth brokers, which control the resources within each domain. Resource allocation requests from end-systems are sent to the bandwidth broker of that domain, which performs admission control for that domain and forwards the request to the next hop domain on the way to the sink domain. For communication between the bandwidth brokers, the SIBBS protocol is proposed.

The following figure is taken from the document cited above.



End system request with fully specified destination

Figure 4-12: SIBBS: architecture and message flow

From the viewpoint of the requirements listed above, the SIBBS architecture has several drawbacks:

- Intra-domain and inter-domain resource control are not clearly separated. Instead, the bandwidth broker is responsible both for controlling resources within the domain and for inter-domain resource allocation.
- This makes it much more difficult to achieve a scalable solution, because the bandwidth broker has to cope with each single request.
- In fact, SIBBS basically assumes, that each end-system request is forwarded along the path to the destination domain. To reduce the number of signalling messages, core tunnels are proposed, which provide a pre-reserved pipe for further bandwidth allocations.

4.2.1.2 BGRP

BGRP is a framework for scalable resource control [BGRP]. It assumes, that BGP is used for routing between domains (autonomous systems, AS).

The basic idea of BGRP is the aggregation of reservations along the sink trees formed by the BGP routing protocol. It is a characteristic of the BGP routing protocol to forward all packets for the same destination AS to the same next hop AS. This property guarantees the formation of a sink tree for each destination AS. All traffic destined for the same AS travels along the branches of this tree towards the root.

Similar to the QBone approach, some kind of “bandwidth broker” is established in each domain. However, not just a single entity is responsible for the whole domain. Instead, an BGRP agent will be associated with each border router. Reservations for the same destination AS are aggregated at each BGRP agent. This has the following implications:

- The number of simultaneous active reservations at each domain cannot exceed the number of autonomous systems in the Internet.
- The source and destination addresses cannot be carried in the reservation requests between domains, because of the aggregation mechanism.

However, the aggregation mechanism does not automatically reduce the number of signalling messages. Each request may still travel end-to-end. Additional damping is necessary, e.g. by reserving additional resources in advance or by deferred release of resources.

In summary, the BGRP framework provides a possible approach to a scalable inter-domain architecture. However, the following issues have to be solved:

- Introduction of a damping mechanism as described above. The authors of [BGRP] make some proposals here. However, also the experiences from the resource pools used for the AQUILA intra-domain resource allocation are well suited to address this topic.
- Because BGRP messages not always travel all the way to the destination domain, the problem of QoS signalling within the last domain towards the destination host has to be solved.
- BGRP is still a framework only. The detailed information exchange between BGRP bandwidth brokers as well as the interaction with the intra-domain resource control have to be specified.

4.2.1.3 NSIS

In November 2001, the NSIS working group was formed in the IETF. The aim of this working group is to “develop the requirements, architecture and protocols for the next IETF steps on signalling QoS” (cited from the NSIS charter).

Currently, the WG is working on the requirements for end-to-end QoS signalling. It is recognised, that these requirements vary in different scenarios. However it is still the goal to define a common protocol, which can be used in all scenarios and areas. The possibility to have different flavours of this protocol is left open.

NSIS clearly states, that the definition of services is out of scope for this WG. Mobile scenarios are considered as an important area for future QoS-aware applications, and are thus explicitly included.

4.2.2 Requirements

An architecture for the AQUILA inter-domain resource control has to fulfil the following requirements:

- Scalability

When high quality services will be established in the Internet world-wide, the number of individual resource reservations will grow rapidly. The architecture must be able to cope with that.

- Works with multiple intra-domain resource control mechanisms

Operators should be free to use any resource control mechanism within their domain. The AQUILA intra-domain approach is just one possible example. An interface must be defined and standardised, through which the inter-domain resource control interacts with the domain specific QoS mechanisms.

- Edge-to-edge QoS guarantee

The architecture must be able to support a certain level of QoS guarantee from the ingress edge of the source domain to the egress edge of the destination domain.

- Stepwise deployment

It must be possible to deploy the architecture in the Internet step by step. An architecture, where any modification or enhancement has to be installed in each AS, is not acceptable.

4.2.3 Architecture

In order to fulfil the requirements listed above, an architecture according to the BGRP framework will be chosen. However, a number of extensions and enhancements have to be added to make a running implementation out of the framework.

Also, ideas and mechanisms developed for the intra-domain resource control will also influence the AQUILA inter-domain architecture.

This chapter specifies the general architecture for the AQUILA inter-domain resource control, where the next chapter addresses detailed aspects.

The following picture gives a rough overview of the architecture and depicts the basic interactions between the intra- and inter-domain resource control layer in the source, intermediate and destination domain.

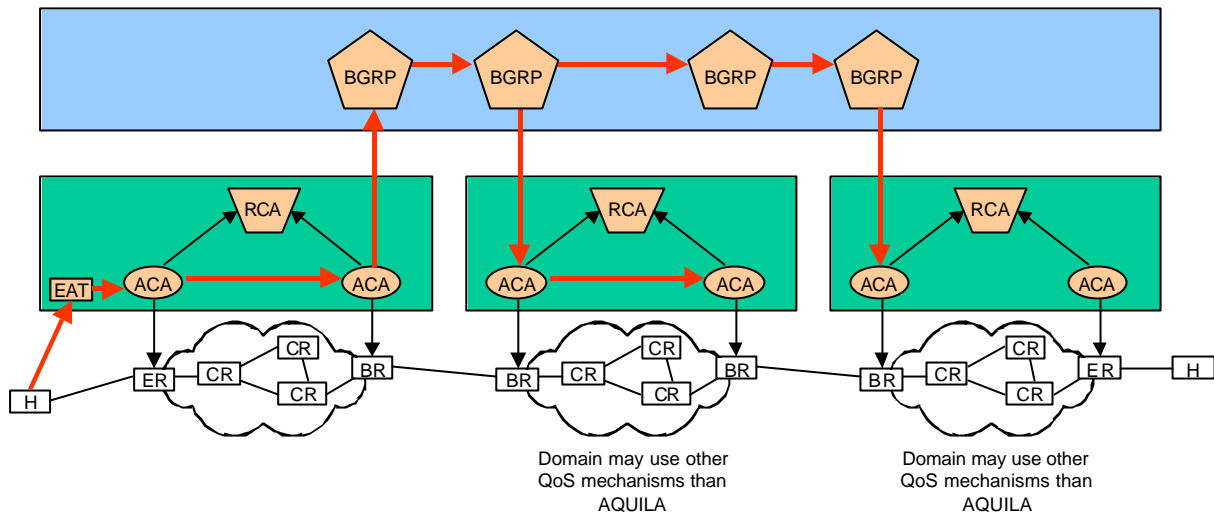


Figure 4-13: General inter-domain architecture and message flow

A so-called BGRP agent is associated with each border router. These agents interact with the AQUILA intra-domain resource control layer in the following way:

- Inter-domain resource requests are initiated by the ACA associated with the egress border router of the initiating domain and sent to the corresponding BGRP agent.
- BGRP agents associated with ingress border routers use the ingress ACA to establish intra-domain resource reservations.

4.2.4 Detailed aspects

4.2.4.1 Offered services

When network services spanning multiple domains administered by different operators are offered, there must be some common understanding of these services between the domains. The offered services also have to be known to the customer. To accomplish this, two opposed approaches are possible:

- Globally well known services (GWKS): There is a set of standardised services, which are implemented by all domains. The customer also knows this set of services he/she can select from.
- Dynamic detection of services: The network may have the ability to detect the possible services from a given source to a given destination which arise from chaining of different services in the different domains on the path from the source to the destination.

The latter induces some serious problems, which largely reduces its usability: When each domain may define its own network services, some kind of mapping has to be performed at each domain

boundary. When multiple mappings are chained, it is very likely that the resulting network service is very unspecific and not suitable for the end-user's application.

Another problem arises, when one of the domains defines fewer network services (e.g. just one premium service) than the previous one and subsequent domains. In this case, several network services have to be mapped to a single service, when that domain is entered. Later on, these packets can no longer be separated in different services in subsequent domains, which possibly again implement a larger number of network services.

So it seems to be necessary, that a fixed number of globally well known set of network services is defined, so that the mapping at the domain boundaries can be eliminated. For each GWKS some parameters of the traffic description are bound to a fixed range, e.g. a maximum packet size.

So the following approach for the definition of network services is proposed for AQUILA:

- Define a set of globally well known services. However, instead of fixing all QoS parameters for these services, leave them open and specify an optimisation target, e.g. no loss, very short delay, etc.
- When an end-user requests a network service, he also specifies an amount of bandwidth for that service.

Note also, that inter-domain services need to specify both an ingress and an egress point (source and destination address). Point-to-any services as the PMC service specified in [D1302] cannot be used in an inter-domain scenario, because it is unknown, which domains the traffic will cross and therefore one is unable to give guarantees for that traffic.

4.2.4.2 Sink-tree-based aggregation

In the Internet, BGP is used as the inter-domain routing protocol. The border routers are the hops, which are considered by BGP. It is a property of the BGP routing protocol to create paths in such a way, that a BGP router forwards all packets to the same destination via a single next-hop router. This guarantees the creation of so-called sink trees. All traffic to a specific destination travels along the branches of such a sink tree towards the root of the tree, which represents an ingress border router of the destination domain.

The AQUILA inter-domain reservation architecture uses this property. It forwards path discovery messages (PROBE) along the BGP routing paths. Reservations are set up in the reverse direction (GRAFT). At each BGP hop (border router), reservations for the same destination (i.e. belonging to the same sink tree) are aggregated.

Note, that this primarily limits the number of simultaneous reservations to be kept at each border router. However, additional mechanisms have to be used to reduce the number of signalling messages travelling end-to-end (see 4.2.4.6, Damping mechanism, page 59).

4.2.4.3 Co-operation with BGP

To determine the path towards the destination domain, the inter-domain resource control layer has to interact with the BGP routing, to get the next hop BGRP agent. The IP address of the next hop border router is determined from the NEXT_HOP attribute of the BGP route.

The only interface between BGRP and BGP will be a possibility for BGRP to query

- a list of neighbours to establish the communication channels between the BGRP agents;
- the BGP NEXT_HOP for a given destination address;
- the NLRI advertised by the destination domain for this path.

4.2.4.4 BGRP agent communication

BGRP agents are set up for each border router in an BGRP enabled domain. Communication is always between “adjacent” BGRP agents. Adjacent BGRP agents can be in the same domain or in different (neighbouring) domains.

Between each pair of adjacent BGRP agents, a reliable and secure communication channel is established.

Reservations are always initiated at the BGRP agent associated with the egress border router of the source domain. This agent represents a leaf in the sink tree towards the destination domain and is therefore called the leaf BGRP agent of the reservation.

The end-point of a BGRP reservation is the BGRP agent associated with the ingress border router of the destination domain. This agent represents the root of the sink and is therefore called the root BGRP agent.

In transit domains, the BGRP agents associated with both the ingress and the egress border router are involved in processing a request. These agents are called ingress and egress BGRP agents.

4.2.4.4.1 Reservation set-up

To set-up a reservation, basically two messages are involved: PROBE and GRAFT.

The leaf agent initiates a reservation by sending a PROBE message to the next hop towards the destination. The PROBE message contains the reservation request, a set of QoS parameters, destination network information and a route record. PROBE messages travel downstream from the leaf agent to the root. When a PROBE message arrives at an ingress BGRP agent, that agent consults its resource database to verify, whether the new reservation fits into bilateral agreement with the previous domain. If the new reservation can be accepted, the BGRP agent inserts its own IP address into the

route record field and forwards the PROBE message to the next hop. Note, that no reservation is carried out and no state is kept in any of the involved BGRP agents during the PROBE phase.

PROBE messages are terminated either due to an error condition at an intermediate BGRP agent or when the message reaches a BGRP agent, which can doubtless identify the tree identification for the destination network (see 4.2.4.7, Quiet grafting, page 60). At the latest, the root agent can terminate the PROBE message.

When the PROBE message successfully arrived at the terminating BGRP agent, a GRAFT message is sent back along the path recorded in the route record field of the PROBE message. The GRAFT message contains a tree ID, which uniquely identifies the sink tree.

At each hop, the reservation is merged with other reservations for the same sink tree. At each ingress hop (except the one in the destination domain), local reservations are carried out to request resources within domains.

GRAFT messages also contain a reference to the intra-domain resource control of the last domain along with the IP address of the ingress border router, which together can be used by the initiating domain to request resources in the last domain along the path from the ingress border router to the end-user host (see 4.2.4.8, Signalling in the last domain, page 60).

4.2.4.4.2 Refresh

Reservations are periodically refreshed between peers. For this purpose, a BGRP agent periodically sends a REFRESH message to all known next-hop and all known previous-hop agents. A REFRESH message is sent for each active reservation for that hop.

REFRESH messages in upstream and downstream direction have different purposes, corresponding to the different purposes of PROBE and GRAFT messages:

- A downstream REFRESH corresponds to a PROBE. This message confirms or reduces the amount of resources requested on that particular link. Downstream REFRESH messages are also used to detect BGP route changes.
- An upstream REFRESH corresponds to a GRAFT. This message confirms the NLRI and other associated information of a particular sink tree.

If an agent does not receive a downstream REFRESH message for a reservation for a specific amount of time, it assumes, that the affected reservations are no longer active and removes them from its internal tables. If a downstream REFRESH message specifies a lower amount of resources than stored in the agent's internal tables, it assumes, that the reservation was reduced.

If an agent receives a downstream REFRESH message for an unknown reservation, This might be the result of a BGP route change. The agent will try to reserve the indicated resources by sending itself a downstream REFRESH message along the sink tree. A "confirm" flag is set in the message to

force the next hop to answer with a REFRESH or ERROR message. If an ERROR message is received or a timeout occurs, the agent will send an ERROR message upstream to all affected previous-hop agents.

4.2.4.4.3 Reservation tear-down

To release a reservation or to reduce the amount of resources allocated to a reservation, the refresh mechanism described before may be used. However to speed up the operation, a TEAR message may be sent. A TEAR message may be seen as a partial REFRESH message. Each agent is free to release the resources as a whole or partially from the next hop agent with a TEAR message, to wait for the REFRESH mechanism, or to keep the resources for later use.

A TEAR message with a zero amount of resources will remove all information for that reservation from the agent's internal tables. When no more previous-hop entries exist for a given reservation, an agent will also send a TEAR message to the next-hop agent.

4.2.4.4.4 Errors during reservation

During the PROBE phase an agent may detect, that the QoS parameters requested for that reservation cannot be handled by the network. An agent associated with an ingress border router may also detect, that the additional request cannot be covered by the bilateral agreement with the previous-hop network operator. In these cases, an ERROR message may be sent back already during PROBE processing.

During the GRAFT phase, an agent may detect, that resources cannot be reserved for that part of the path it is responsible for. In this case, the agent will send an ERROR message upstream to inform the requestor and may additionally send a TEAR message downstream in order to release the already reserved resources. Sending of the TEAR message is optional. The agent may also wait for the UPDATE mechanism to propagate the change or keep the resources for possible future reservations.

4.2.4.5 Interface to intra-domain resource control

An inter-domain reservation is always initiated by some intra-domain resource control entity in the source domain. Inter-domain resource control uses an intra-domain resource control entity to request resources within a domain and on links between domains. Figure 4-12 on page 49 illustrates the communication between these entities.

The interface has to fulfil the following requirements:

- At the initiating domain: Request of resources to a destination domain. The request should contain the network service, traffic descriptor, required QoS parameters, destination. The response should contain a success indicator and the actual QoS parameters.

- At a transit domain: Request of transit resources through a domain. The request should contain the network service, minimum traffic descriptor, preferred traffic descriptor, ingress and egress. The response should contain a success indicator and the actual QoS parameters.
- At the destination domain: Request of destination resources. The request should contain the network service, minimum traffic descriptor, preferred traffic descriptor and the ingress. The response should contain a success indicator and a reference ID, which can be used by the intra-domain resource control of the initiating domain to request resources within the last domain from the ingress to the end-user host.

4.2.4.5.1 AQUILA intra-domain roles

AQUILA defines a three-role-model for intra-domain reservations. The three roles are:

- a requestor, which initiates the request;
- a sender, which injects the traffic into the network,
- a receiver, which receives the QoS enabled traffic.

The inter-domain approach specified in the current document however, follows a sender-initiated model for reservations. This is necessary, because the GRAFT messages for path discovery always have to follow the BGP routing path from the sender to the receiver. This path cannot be discovered in the reverse direction from the receiver to the sender, nor can it be determined by a third party.

To match both approaches, the following mechanism for three-role-reservations is proposed:

1. If the requester is not in the same domain as the sender, the requester will initiate a reservation to the sender with a bandwidth request of zero. The answer to this request will include the reference to the intra-domain resource control of the sender domain.
2. The requester will then request the required resources from the intra-domain resource control of the sender domain. This request will include information, which enables the sender domain to charge the requester for the reservation.
3. Intra-domain resource control of the sender domain will then carry out a “normal” sender-initiated resource request, including the initiation of the inter-domain request to the receiver domain, if necessary.

This approach to the three-role-model adds the following requirement to the interface to intra-domain resource control:

- At the source domain: Request of inter-domain resources. The request should contain the network service, requested QoS parameters, traffic descriptor, source, destination and charging au-

thorisation information. The response should contain a success indicator and the actual QoS parameters.

4.2.4.5.2 *Role of ACA for border routers*

In the intra-domain architecture of AQUILA, admission control agents are responsible for traffic carried over edge devices into and out of the network. In a multi-domain scenario, border routers play a similar role for traffic exchanged between neighbouring domains. There is, however, one mayor difference:

While edge devices are able to police and mark traffic on a per flow base, this is not achievable at border routers. So border routers have to police traffic on a coarser level. There are the following two possibilities:

- Border routers may police and mark traffic “per reservation”. In the BGRP scenario this means per sink tree and per traffic class.
- Border routers may police and mark traffic “per traffic class”, aggregating all reservations for the same traffic class.

Note, that policing at domain boundaries has a very different basis than policing at the ingress edge device:

- Policing at the edge device controls, that the customer does not send more traffic than admitted. Excessive traffic may be degraded or dropped. The network operator does not give any guarantee for that.
- Policing at the border router however controls, that the neighbour domain does not send more traffic than admitted. However, if you choose to drop or degrade the excessive traffic, the end-customer will be affected, not the neighbour domain. So dropping is not the right “punishment” for excessive traffic between domains.

Instead, it might be useful to negotiate service level agreements between domains in such a way, that excessive traffic is charged with a high price. In this case, the neighbour domain will be interested in controlling the QoS traffic it generates or receives and forwards. However, the definition of such models is behind the scope of the project.

In any case the border router has to police incoming traffic. This might be done on different levels of detail:

- Per reservation handling:
 - Border routers must be able to set up a multi-field classifier based on the DSCP field (to identify the traffic class) and a list of IP prefixes (to identify the sink tree). Currently, the AQUILA intra-domain architecture can handle only a single IP prefix.

- When the neighbour domain sends more traffic than admitted, the policer can identify the violating reservation.
- Per traffic class handling:
 - Border routers must be able to set up a classifier based on the DSCP field.
 - The policer cannot detect excessive traffic in a single sink tree, as long as the sum of the traffic in all trees of the same traffic class does not exceed the overall limit.

For the AQUILA project we propose to use per traffic class handling. The policer should be set up in such a way, that excessive traffic is not immediately dropped or degraded, but reported for further administrative actions. Dropping or degrading should occur, when the traffic reaches a level, which would affect the network's ability to carry other (non-offending) QoS traffic.

4.2.4.6 Damping mechanism

Cited from [BGRP]:

The basic BGRP protocol aggregates reservations into trees, thereby reducing the number of reservations. We will quantify this in Sections V-A and V-B. Reducing the number of reservations obviously shrinks the memory needed to store the control state information. It also reduces the overhead associated with REFRESH messages for all these pieces of control state; refresh costs include CPU processing and link bandwidth. These savings take us much of the way toward our goal. However, BGRP's other control messages, PROBE and GRAFT, also consume processing and bandwidth. We would like to control the volume of these control messages as well and thereby add another dimension of scalability to BGRP. This can be done by making the following enhancements to the protocol.

The enhancements proposed by the cited paper include the following:

- Over-reservation, quantisation and hysteresis
- CIDR labelling and quiet grafting
- Reservation damping

The first proposal tries to reduce the number of reservation messages by using mechanisms, which request more resources than actually needed or delay the release of resources, in order to use them for subsequent reservations. Note however, that this does not reduce the number of PROBE messages. This proposal is discussed in more detail later on in this chapter.

The second proposal also addresses PROBE messages. It tries to respond to PROBE messages already at an earlier point in the sink tree, by enabling intermediate BGRP agents to identify the sink

tree, which will be used for that reservation. This topic is discussed in more detail in chapter 4.2.4.7, Quiet grafting, on page 60.

The third proposal addresses the problem of reservation changes due to BGP route changes. As many of the daily BGP route changes are pathologically and do not reflect real network topological changes, mechanisms could be used to reduce the number of unnecessary reservation moves. In spite of the fact, that this is a undeniable problem in real networks, the project will not further investigate it.

4.2.4.7 Quiet grafting

So far, the source domain cannot uniquely identify the destination domain, and thus the sink tree for a new reservation, without sending the PROBE message all the way to the destination domain. BGP does not provide this information in its AS_PATH attribute, because paths leading to a set of autonomous systems may be aggregated and can no longer be distinguished from the source domain's point of view.

Quiet grafting tries to enable other nodes before the final one to determine the sink tree, to shorten the average distance the PROBE and GRAFT messages have to travel. This will also reduce the signalling load.

To enable quiet grafting, the GRAFT messages will additionally carry a (possibly incomplete) list of announced IP address prefixes of the destination domain. This list contains at least the IP address prefix covering the destination address sent in the PROBE message, but may also include other address prefixes announced by the destination domain. However, this list may not contain any IP address prefix space, which is possibly covered by another more specific route. Each BGRP agent stores this information along with the reservation. The restriction formulated in the previous sentence guarantees, that there are no overlapping IP prefixes stored in this table. If – due to any failure or disregard of that restriction – overlapping IP prefixes exist in that table, the most specific prefix should be used for any lookup in this table.

When a PROBE message arrives at a BGRP agent, the current list of known IP prefixes is consulted to possibly find an already existing reservation towards that destination domain. BGP is consulted to verify that the reservation found matches the IP prefix for that destination. If an existing matching reservation is found, pre-allocated resources may be used and the PROBE message may already be answered with a GRAFT. In addition, the offered QoS parameters from the actual location up to the destination may be compared to the requested parameters and the reservation may be rejected, if the requested QoS parameters cannot be met.

4.2.4.8 Signalling in the last domain

The damping mechanisms and quiet grafting described above inhibit the forwarding of signalling messages to the last domain. So that domain may be unaware of a new reservation. Specifically, no resources are reserved on the path from the ingress border router to the destination host.

The AQUILA inter-domain architecture addresses this problem by returning a reference to the intra-domain resource control and the address of the ingress border router within the GRAFT message to the initiating domain. The initiating domain then explicitly requests the resources in the destination domain, if necessary.

In order to authorise this request, it has to contain a reference to the bilateral agreement of the last domain's network operator with the last but one, so that the request could be interpreted as follows: "You (the last domain) have already promised to the last but one domain to receive my traffic. With reference to this promise I request to carry my traffic to the final destination within your domain."

4.2.5 Scalability

Scalability in resource reservation has several – partly correlated – dimensions:

- **Number of reservations:** How many reservations are simultaneously active at each network element? A high number of reservations needs a high amount of memory to store the reservation states. Additionally, this will also mean a high number of signalling messages, because the states most probably need to be refreshed regularly.
- **Number of hops a reservation request has to cross:** Can a reservation request be handled "near" to the requester or does it have to travel all the way to the destination? A large number of hops means a high number of signalling messages in the network and thus a high amount of CPU processing power, especially for backbone domains.

The proposed architecture inherently solves the first issue. The number of simultaneous reservations cannot exceed the number of autonomous systems in the Internet multiplied by the number of network services offered. So the architecture exposes a linear effort growth regarding the number of autonomous systems in the Internet.

However, without the quiet grafting and damping mechanisms, each signalling message still has to travel all the way from the source to the destination domain. As shown in [BGRP], this may lead to scalability problems, especially for short-lived small bandwidth flows, such as VoIP. Transit domains are mainly affected by this behaviour.

4.2.5.1 A first approach for quiet grafting

The use of BGRP as the inter-domain protocol in the AQUILA architecture provides an end-to-end resource allocation solution. However, the current proposal of the BGRP does not contain a detailed analysis of the protocol and in addition the protocol is not yet implemented in a real network environment. That implies that the deployment of the BGRP protocol to the AQUILA architecture should be analysed detailed described and finally implemented. So, many open issues should be encountered and a first solution should be given.

This contribution handles with the scalability problems arising from the deployment of the BGRP, and especially with issues such as quiet grafting and damping mechanism. A first approach is given as far concern the quiet grafting mechanism, which provides a solution to scalability problems.

4.2.5.2 PROBE & GRAFT messages

The main scalability problem arises from the fact that a reservation request has to cross a significantly large number of hops in order to reach the destination host. The latter implies that a large number of PROBE and GRAFT messages should be exchanged and processed by the intermediate Border Routers. That has as result a lot of bandwidth and processing power to be consumed. Therefore, a major task is the control of number of PROBE and GRAFT messages, limiting their number and adding in this way scalability to the BGRP protocol.

Quiet Grafting is a basic solution for limiting the signalling messages. In [BGRP], it is not described the mechanism, but only a first approach is given.

4.2.5.3 Classless Inter-Domain Routing (CIDR) Labelling

Classless Inter Domain Routing (CIDR) is a new addressing scheme for the Internet, which allows for more efficient allocation of IP addresses than the old Class A, B, and C address scheme. CIDR is a replacement for the old process of assigning Class A, B and C addresses with a generalised network "prefix". Instead of being limited to network identifiers (or "prefixes") of 8, 16 or 24 bits, CIDR currently uses prefixes anywhere from 13 to 27 bits. Thus, blocks of addresses can be assigned to networks as small as 32 hosts or to those with over 500,000 hosts.

A CIDR address includes the standard 32-bit IP address and also information on how many bits are used for the network prefix. Therefore, routing destinations are represented by network and mask pairs and moreover routing is done on a longest-match basis (i.e. for a given destination, which matches multiple network-mask pairs, the match with the longest mask, is used). A typical CIDR address for a network could be of the form 147.102.32.0/19 where the IP address is 147.102.32.0 and the mask is 255.255.224.0 (it corresponds to 19 bits). In this way, routers do not just keep the IP address of the networks for making routing decisions but also maintain the information that is introduced by the mask. It is obvious that an IP address of the old addressing scheme, for example 147.102.240.0 is different from the same IP address of the new scheme where a mask is additionally defined, i.e. 147.102.240.0/20. As a result, CIDR brings flexibility and adaptability to assigning address spaces that fit an organisation's specific needs.

It is often the case that the border routers are not aware of the exact set of CIDR addresses that are reachable to a specific border router. This is due to the fact that the CIDR addressing scheme enables "route aggregation" in which a single high-level route entry can represent many lower-level routes in the global routing tables. Therefore, a border router may keep the list of its reachable CIDR address destinations but will not advertise the same list to the adjacent border routers. It will perform an aggregation of its routes if possible (it will not aggregate those routes that cannot be treated as

part of a single unit due to multi-homing, policy or other constraints) [RFC1519] and will advertise the aggregation of routes that will always be a superset of the explicit routes.

It is envisaged that CIDR will be used in BGRP for the labelling of the sink tree. Therefore, an analysis is needed for the identification of the possible problems arisen and the enhancements introduced to the scalability of BGRP.

Since CIDR labelling will be used for the identification of a reservation tree, it is important that a CIDR address suffices to guarantee the uniqueness of a sink tree passing from a border router. In other sense, it is required that a router can tell, based only on the CIDR label, that a new reservation belongs to a sink tree and there is no way that there is another sink tree with the same CIDR label passing from the same router. For example, it is possible that two border routers of an AS reach the same sub-domain and therefore advertise it as one of their destinations. Both routers can form sink trees for this destination domain (they are roots of the sink trees) making in this way the CIDR labelling not sufficient for identifying a sink tree. It is obvious that the IP address of the router is also needed for that purpose.

Nevertheless, taking into consideration that the BGP route selection is based on the shortest path, it is certain that there is only one route passing from a border router to a specific destination. Therefore, the above assumption is not valid and the uniqueness of a CIDR labelled sink tree is guaranteed within the list of sink trees that a border router belongs to. The following figure (Figure 4-14) depicts more clearly the above concept.

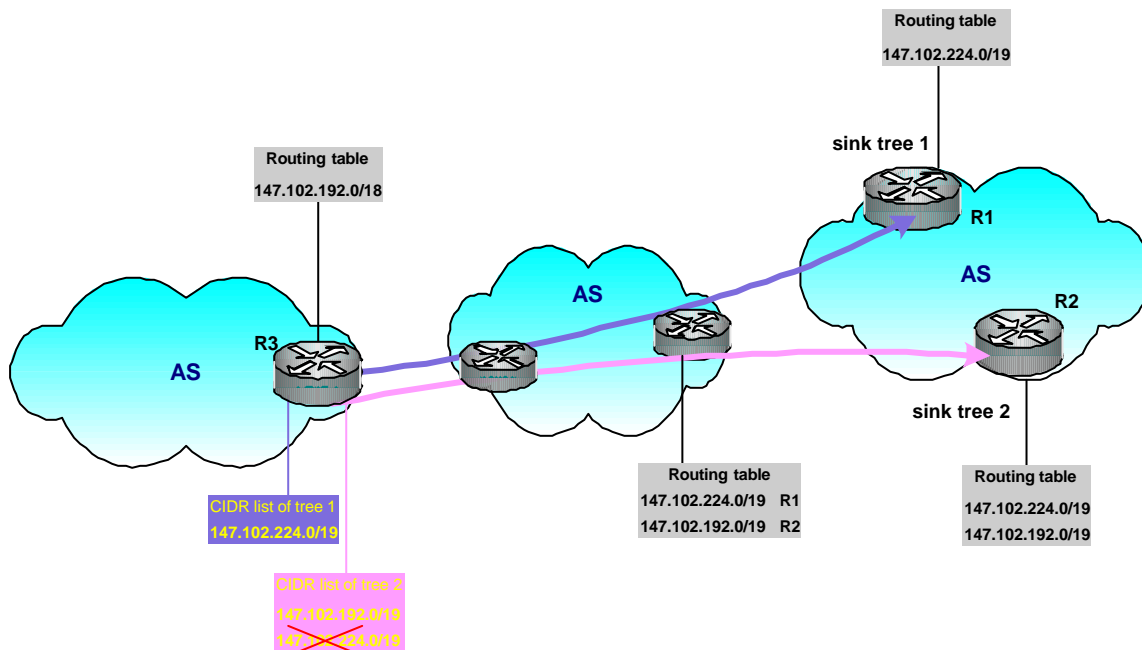


Figure 4-14-Uniqueness of a CIDR labelled sink tree

Packets passing from R3 to destination 147.102.224.0/19 will always be forwarded to router R1 based on the shortest path algorithm that is used by BGP (even if this destination is also reachable by R2). Therefore, they will belong to tree 1 and as a result the CIDR label of each tree will be unique. In case that R2 also advertises the same destination to the R3 BGRP agent, the agent has to understand that this label is not valid since the packets to that destination will not follow the tree 2 route based on the BGP routing decisions.

As mentioned before, CIDR addressing scheme performs route aggregation. Therefore, a border router may not be aware of the explicit routes of an aggregation but will most often know the summarisation of these routes. Supposing, that two border routers (R1 and R2) of an AS have access to the CIDR addresses 147.102.224.0/19 and 147.102.192.0/19 respectively. A distant router may not be aware of that explicit information but might instead be aware of a route to 147.102.192.0/18 CIDR address (aggregation of the two addresses). Supposing that one of these routers (R1) is root to a sink tree and that the distant router (R3) belongs to that tree. It is therefore essential that the latter has a more specific view of the domains that can be reached by the sink tree it belongs to enabling in this way the quiet grafting of a future reservation request to the tree. Therefore, it is required that the root of a sink tree provides the other border routers with the list of its reachable sub-domains (within the GRAFT message). This list should at least contain the destination CIDR address of the existing resource reservations, which implies that it will be continuously updated while reservations for new destinations are incorporated into the sink tree. Apart from the CIDR destination addresses of the existing reservations, the root can also include in the GRAFT message some more reachable CIDR addresses in order to enable quiet grafting for future reservations to new destinations. It is FFS whether or not these destinations are included at the first GRAFT message that forms the sink tree or they are gradually distributed to the routers of the tree.

Moreover, a certain problem arises from the advertisement of the whole CIDR list from the root of a sink tree. Supposing that the border router R2 is also root to a sink tree to which R3 belongs as well. One more assumption is that R2 can also reach the sub-domain 147.102.224.0/19 and that it has sent this information to R3 within the GRAFT message. Therefore, router R3 maintains two lists of CIDR addresses for the two different sink trees. The list that has been created from the sink tree of R2 contains two CIDR addresses, 147.102.224.0/19 and 147.102.192.0/19 whereas the other list contains the CIDR address 147.102.224.0/19. At this point, when a new reservation request is made for the domain 147.102.224.0/19, the BGRP agent that resides at the router R3 will not know the sink tree to which it should graft the current request. One possible solution is that it keeps some further information in order to distinguish between the CIDR addresses of a list (whether there exist reservations for destination CIDR addresses). In this way the BGRP agent will know that the sink tree of the router R1 provides the only route to that destination (if a reservation has already been made to that tree for that destination). The above problem is illustrated in the following figure (Figure 4-15).

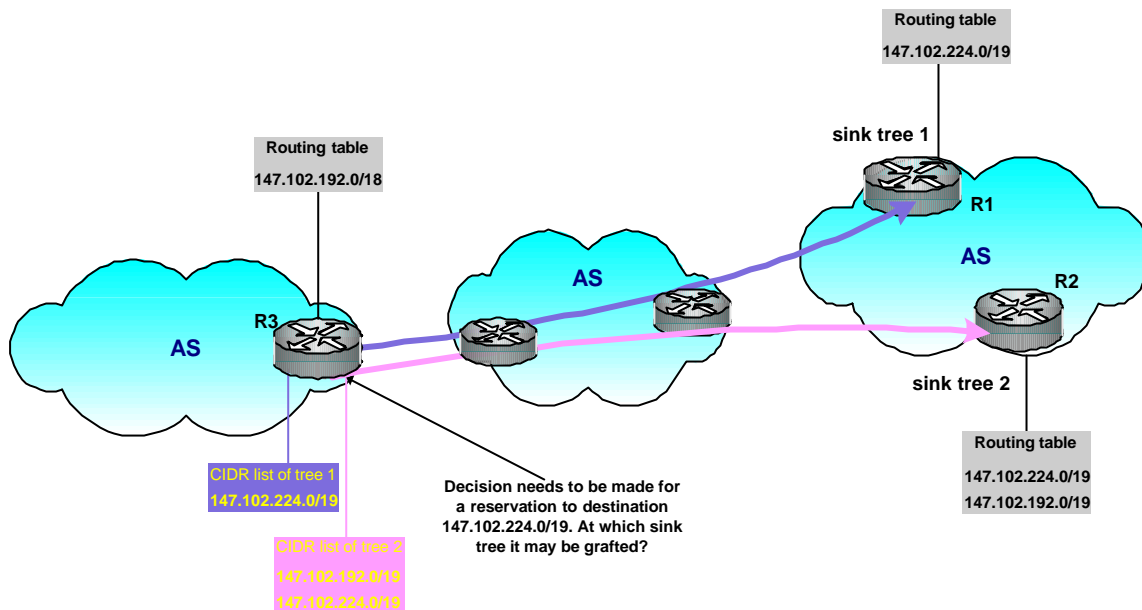


Figure 4-15-Distribution of destination CIDR addresses

It is important to be noted that CIDR is supported by BGPv4 and implemented in most routers.

To conclude, the CIDR labelling guarantees the identification of the sink trees that a border router participates in and therefore justifies its use for the tree label. Furthermore, the way that a root of a sink tree distributes its destinations as well as which destinations are finally distributed has to be decided.

4.2.5.4 Mechanism for Pre-Allocating Bandwidth

As aforementioned, a certain amount of bandwidth should be pre-allocated in order to serve future requests, and in this way limiting the number of PROBE and GRAFT messages.

Resource Pools algorithm for dynamic bandwidth distribution might be a first solution, but the BGRP is a particular case with different requirements and characteristics. The main factors that influence the algorithm are:

- The arrival rate of reservation requests
- The duration of each reservation
- The average bandwidth of reservations requests
- The sink trees created
- The popularity of a certain sink tree (destination host)

- The target network resources utilisation
- The target limitation of messages overhead

All those factors should be considered when determining the algorithm for pre-reservations.

The BGRP agent will not be regarded as a part of the RPool hierarchy, but it has partially a common functionality with the RPool.

4.2.5.4.1 Mechanism for Releasing BW: A crucial Point

The concept of quiet grafting is based on a resource pre-allocation mechanism in order to serve future requests. By following that concept the risk of reserving resources that aren't actually needed is emerging.

Thus, an additional mechanism for the release of any "unwanted" pre-allocated resources should be defined. The term "unwanted" is used for resources that have been pre-allocated but based on some criteria are not needed any more. These criteria might be defined following many different approaches (i.e. time-based, event-based or measurement-based).

In a time-based approach the pre-allocated resources will be released if they haven't been used after a certain period of time. That approach requires the use of timers in order to keep track of the time period, which generally might add complexity in the system. However, this is not our case because timers will be included for the support of the UPDATE messages. The event-based approach is actually the one that is currently used in AQUILA. Each time a reservation is released, the RPL is checking whether unused resources should be given back or not. Each time an event occurs (e.g. release of a reservation) an algorithm, which can be similar to those that are already discussed in AQUILA, will determine if there are any "unused" resources and how many of them should be released. Finally, the measurement-based approach could use measurements concerning the utilisation in order to take the decision for the release of resources.

Nevertheless, the approach that will be chosen should be consistent with the logic of the pre-allocation mechanism. In other words, the criteria used for the pre-allocation of resources should be conformant to those for the release otherwise the mechanism will become inefficient. For instance, if the criterion for pre-allocating resources is the rate of the reservation requests (which might reflect the popularity of the given sink tree) then more resources will be pre-allocated to that node. If, on the other hand, the release algorithm is invoked every time a reservation request is received, that node will very frequently check for "unwanted" resources while probably all the assigned resources are "wanted".

The actual specification of the algorithms used in AQUILA is contained in deliverable D1302.

4.3 End-user application toolkit

The End-user Application Toolkit (EAT) is an application that aims to provide access to end-user applications to QoS features. The EAT is a middleware between the end-user applications (Basic Internet Applications and Complex Internet Services) and the AQUILA network infrastructure.

The tasks of the EAT are: to allow legacy applications (QoS-aware and non-QoS-aware) to benefit from QoS features, and to allow, by the way of an API, the implementation of QoS-aware EAT-based applications.

This toolkit provides reusable and generic components for both client and server sides of applications. The toolkit should be used by end-users as well as by application developers.

The EAT should support a variety of operating systems, network and reservation protocols, by providing a logical abstraction that hides the low level details of particular systems. The EAT is structured in such a way that it will allow easy update. For this reason the EAT consist of several, distributed components with different tasks and to support different protocols, for example QoS protocols. Moreover, the EAT includes some platform-specific components (e.g. some application proxies) while the whole toolkit is platform independent.

In the following, the overall architecture is introduced at first. After that, the different application interfaces are described in more detail. And finally, a short deployment scenario is given.

4.3.1 Overall architecture

As depicted in Figure 4-16, the refined EAT will consist of the several basic building blocks. The main components are:

- The EAT Manager: is the main part of the EAT and controls the whole process. It is responsible for managing users and reservations. Moreover it acts as mediator between the other EAT components and towards the ACA. It is also important to mention that the EAT Manager directly implements the internal EAT API interfaces. In other words, the API is the interface of the EAT Manager towards applications and GUIs.
- The Converter: The main task of the EAT corresponds to the mapping/converting of the (by the end-user subscribed) network services and application profiles into session characteristics (QoS options) corresponding to the application in use (see chapter 4.3.4).
- The GUIs: They are mainly to support non-QoS-aware applications. The three main GUI dialogs are: one for manual reservation requests on the advanced level, one for requests on the regular level, and one to display, group, and release the active reservations (see chapters 4.3.2 and 4.3.4).

- The (internal) EAT API¹: enables the development of new applications that will directly use the AQUILA QoS architecture (see chapter 4.3.3).
- The Proxies* are for applications which are not based on the EAT's API. They enable the selective processing of the control plane information by forwarding QoS-relevant data to the EAT Manager (see chapter 4.3.5).

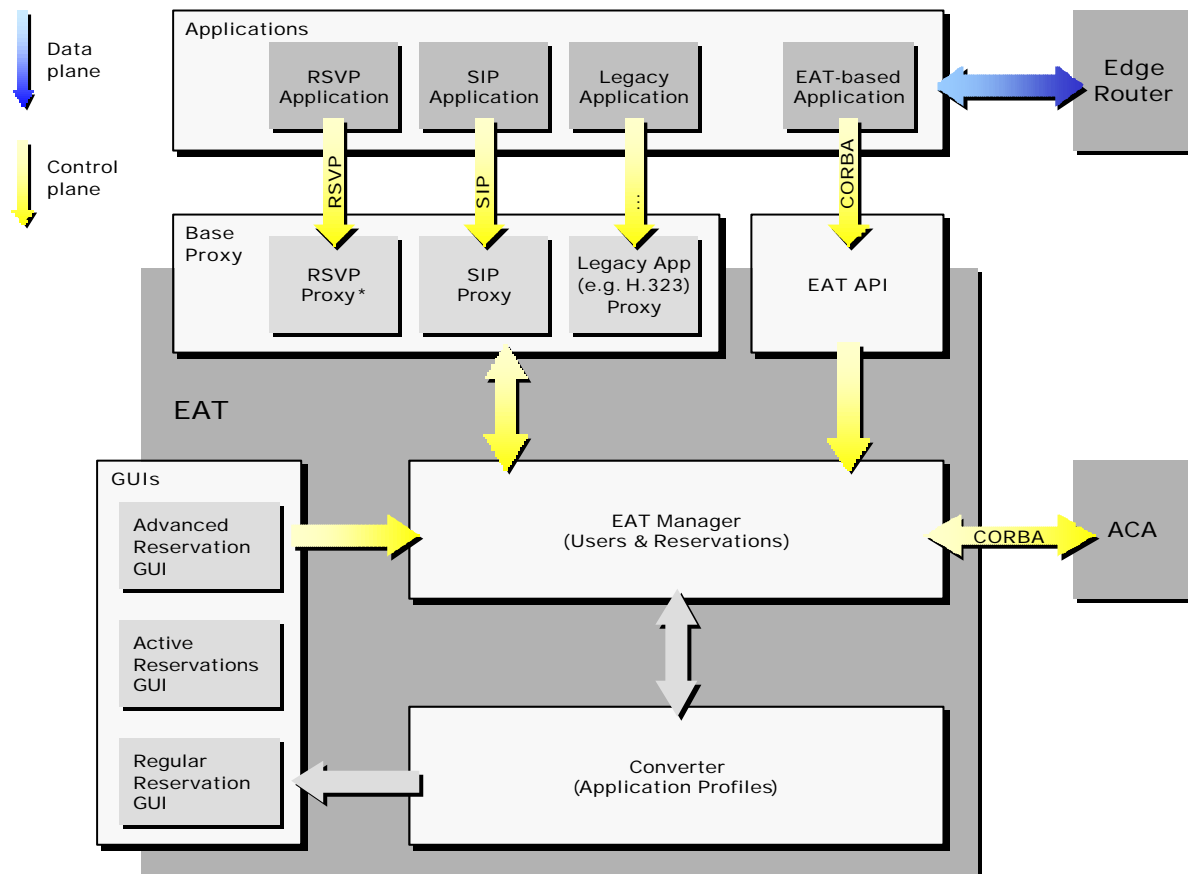


Figure 4-16: Block diagram of the overall EAT architecture

Additionally, there are three packages/modules for further tasks:

- Script: It contains a class for batch processing of reservation requests and releases.
- Application Profile: It manages the application profiles which are stored in a directory server (see chapter 4.3.2).

¹ The implementation of a general QoS API is not foreseen for the second trial.

* The RSVP Proxy will also not be implemented for the second trial.

- Persistence Layer: It provides an own persistence layer mechanism to store the reservation history (incl. accounting data) in a database.

4.3.2 Non QoS-aware application support

One of the aims of the AQUILA project is to support legacy applications without changing them. The AQUILA project proposes an approach based on application profiles and proxies. The application profile approach can be used if precise information about an application is known, the proxies are in a first time used when an application uses for its data flow dynamically negotiated ports. The application profile is used by a converter that translates end-user decisions at application level into reservation requests.

One aim of the second trial is to develop a complex Internet service that will use the capabilities of the AQUILA solution.

4.3.2.1 State of the art, references, ideas

There are few QoS projects taking (legacy) applications and end-users into account. The following chapter gives an introduction to the different approaches relevant for the AQUILA approach.

- The project SABA2 [SABA2] supports multimedia non QoS-aware applications with static (VIC and VAT) and variable (ISABEL [ISABEL]) QoS requirements. It proposes a RSVP [COMA] agent based on a generic QoS API [QOSWG] that monitors the traffic generated by the application. The RSVP agent generates RESV messages following the transmission rate of the application.
- In the works [BISSEL], [BOGEN] interactive real-time applications are analysed in a comparable way as for the AQUILA application profiles. This work could be a further information source.
- The INDEX project [ALTM] is indirectly for relevance for the AQUILA application support approach. This project gives arguments for an user friendly approach for providing QoS.

“The INDEX project, which stands for INternet Demand EXperiment project, is a field experiment for investigating people’s willingness to pay for a certain service quality. The subjects of the INDEX project, who got a installation of a dedicated ISDN line between their homes and the Internet for free, have the possibility to choose between different options (i.e. as connection speeds) at different prices depending on the current experiment running.”

„Since the user is concerned about how much money he is going to spend on a service, this item has to be taken into account when designing a QoS architecture.“ [BRUS]

- Here a short description of the Eclipse [BLAN] operating system. This is for information only. The methodology used is in our opinion not relevant for the AQUILA approach. Nevertheless a reviewer of a submitted paper to the IWQoS2001 mentioned it as missing reference.

The Eclipse operating system is a test bed for Quality of Service (QoS) based on FreeBSD version 3.4. Eclipse provides flexible and fine-grained Quality of Service (QoS) support for applications. Eclipse is being used to guarantee QoS to server applications, in particular, to differentiate the performance of different web sites hosted on the same platform. Have been implemented:

- i) hierarchical proportional-share CPU, disk and link schedulers,
- ii) the /reserv file system providing an API to manipulate "reservations" and
- iii) a tagging mechanism for the association of reservations with schedulable operations.

“The use of default lists and garbage collection makes it possible to provision resources for legacy applications. Reservations can be assigned to the default list of an application without its knowledge and it will transparently obtain the QoS support provided by these reservations. When the application finishes, these reservations can be transparently garbage collected.”

4.3.2.2 Application profiles and Basic Internet Applications

The application profiles support the QoS offer towards end-user for legacy (non QoS-aware) Basic Internet Applications. A Basic Internet Application is a usual Internet component like: voice over IP application, video conferencing application, TV on demand, ftp, streaming application, audio applications... The aim of the profile is to enable and keep a mapping activity between the end-user, the application and the network transparent with the goal to be generic and implementation independent.

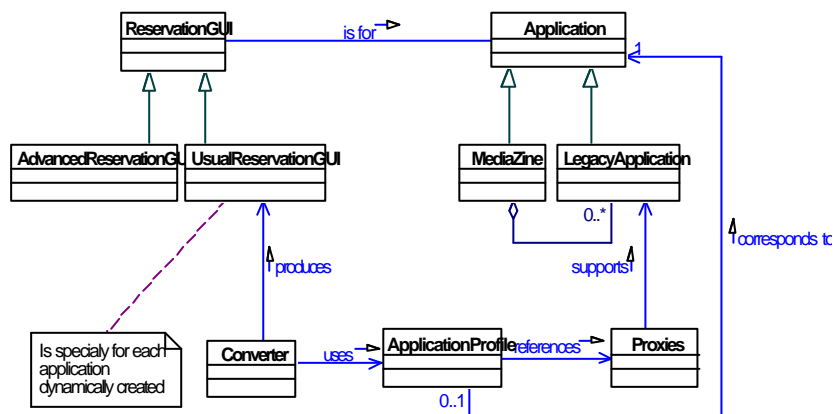


Figure 4-17: Relation between the reservation GUI and the application profile

The application profile describes the information needed for the mapping between application level and network level. One part of the profile is very close to the AQUILA architecture and approach and not very generic whereas other parts of the profile are very generic and AQUILA independent.

The application profile is the basis for the transformation of the end-user request in an AQUILA (or other QoS enabled infrastructure) request. A converter is responsible for the mapping between application, end-user and network level.

To create an application profile for a concrete application it is necessary to in a first step analyse it, in order to determine the values of the different parameters.

The application profile syntax is ruled by a DTD and the concrete information concerning an application is saved in an XML file.

4.3.2.3 Application profiles

With the `ApplicationProfile` syntax it is possible to:

- Describe the protocol used by an application by applying the `protocol` syntax
- Describe the different service components composing an application by applying the `ServiceComponent` syntax
- Describe the transport protocol of each service component by applying the `TransportProtocol` syntax
- Describe each service component by applying the `ServiceComponentProfile` syntax explained below.

With the `ServiceComponentProfile` syntax it is possible to:

- Describe the different possible QoS options of a service component by applying the `Option` syntax
- Describe the QoS expectations of the service components by applying the `QoSRequirement` syntax
- Describe the traffic produced by a service component by applying the `TrafficSpecification` syntax
- Describe application characteristics at end-user level in a user-friendly manner by applying the `SessionCharacteristic` syntax

4.3.2.4 Application profiles and Complex Internet Services

4.3.2.4.1 QoS-aware Complex Internet Service

Complex Internet Services are services offered by e.g. a content provider to a customer group in form of a web platform integrating, binding and presenting Basic Internet Applications

The AQUILA Complex Internet Service is Mediazine - a platform for music fans. As the current web technologies do not enable at application level a direct use of QoS technologies, there exist three possibilities to develop the Mediazine that are depicted and evaluated in the Table 4-1. The Mediazine will use the AQUILA QoS API the so called EAT API in order to produce the QoS requests. It will integrate the user-friendly QoS selection/request (using the profiles of the integrated basic applications) in its overall GUI (see Figure 4-18).

Solution A	Solution B	Solution C
Bind basic Internet applications together	Implement from null a service supporting QoS	Implement a service based on a QoS API
To implement:		
<ul style="list-style-type: none"> Platform integrating the applications using proxies or API 	<ul style="list-style-type: none"> Basic applications Platform Signalling protocols 	<ul style="list-style-type: none"> Basic applications Platform using API
Remarks		
Most efficient	Not efficient	Basic application to rewrite
AQUILA solution		prepared for future applications

Table 4-1: Possible solutions to develop a QoS aware Complex Internet Service

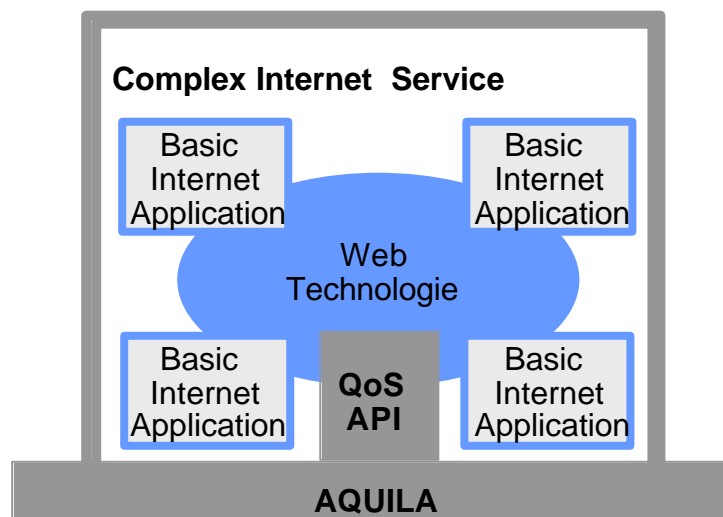


Figure 4-18: Complex Internet Service and AQUILA

4.3.2.4.2 CIS interacting with the EAT

The EAT API offers the access to end-user oriented application information by the way of the converter. The profile approach is used by the EAT API in order to provide to the “new” QoS aware complex Internet services an user friendly QoS interface towards the end-user.

The diagram below (Figure 4-19) gives an example of the utilisation of the application profile by the Complex Internet Service “Mediazine” (for more information please refer to [D2203]).

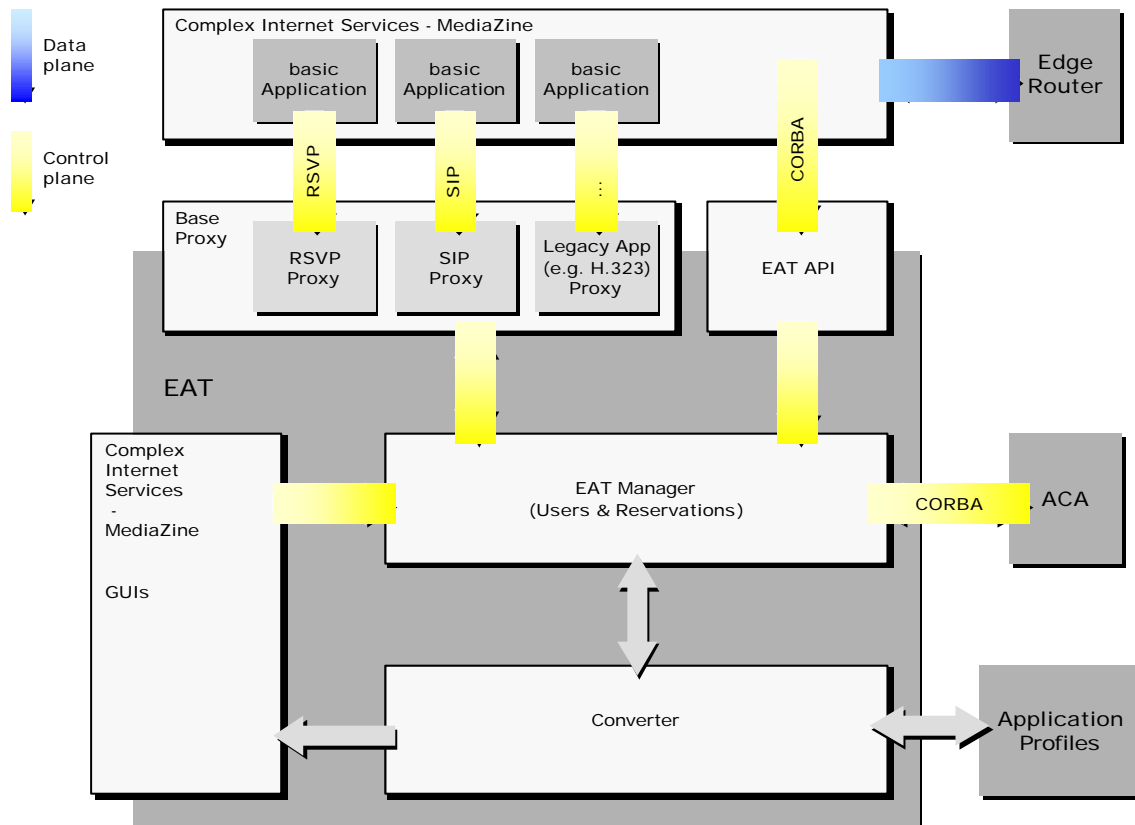


Figure 4-19: Block diagram representing the EAT and the CIS

4.3.2.5 QoS monitoring

An important feature concerns the QoS monitoring. The end-user is paying for a “quality”. This quality is as mentioned in [ABOBA] very objective and depends on the kind of applications, their usage, the art of the agreement concluded with the ISP etc. Nevertheless a kind of monitoring can be important and can convince the end-user by for example monitoring the delivered bandwidth.

This feature can be supported by a measurement architecture that continuously measures the delivered bandwidth and could give feedback to the EAT that could present it in a graphical form.

4.3.2.6 Reservation adaptation

Many legacy applications, to cope with the best effort QoS of the actual Internet adapt their resource needs to the resource level and implement QoS adaptation at application level. This means

that the produced traffic is not constant and predictable. A solution associating packet sniffers and a measurement architecture could enhance the AQUILA architecture.

4.3.2.7 Service presentation to the customer

Imagining the case of a SLA negotiation it is conceivable to base the negotiation on the predefined SLS. Two scenarios are thinkable: one presenting the SLS information in an advanced GUI for “professional user”, the other one presenting the SLS information in an user-friendly way. The latter assumes a mapping mechanism in order to translate the technical parameters into a user friendly “language”. An approach similar to the application profiles might be useful.

4.3.3 End-user Application API

The Application Programming Interface (API) of the End-user Application Toolkit provides **application services/interfaces** for QoS. Via this API, application and service developers can directly access – i.e. by using a programming language – the QoS features of the AQUILA’s Resource Control Layer.

In detail, the EAT API provides services for:

- the end-user’s authentication against the network,
- the retrieval of by the RCL offered QoS network services (predefined SLS’),
- the retrieval of Service Level Agreements (SLAs),
- the retrieval of pre-defined application profiles and their available QoS options,
- the retrieval of installed protocol gateways to be used for special applications,
- the request and the release of QoS reservations,
- as well as the retrieval of status information concerning QoS reservations (accounting results, ...).

In contrast to signalling protocols, which serve applications at network level, the API serves applications at application level. The advantages of this approach are:

- The API is a single component to be used by all applications. In the case of the proxy dealing with signalling protocols, a separate proxy must exist for each supported signalling protocol. This causes a very high effort.
- Predefined SLSs (network services) and SLAs can be presented towards applications and end-users.
- With the API it is basically possible for an application to get feedback from QoS measurements and accounting information.

- The API can easily allow several requests per application session.
- End-users can directly be involved in choosing the QoS level.

With signalling protocols and proxies it is difficult if not impossible to realise the above mentioned features.

However there is one major disadvantage of the API approach:

- An application must be modified in order to access the API. This is not possible for many existing applications.

Application development is not the focus of the AQUILA project. Instead, we use a Complex Internet Service (e.g. Mediazine). Such a service encapsulates unmodified Basic Internet Applications and does the reservation requests for them. The realisation of such a service is much more efficient than to develop/modify Basic Internet Applications.

Furthermore, there are two more or less contrary requirements for the API:

1. The API must provide the full functionality of the AQUILA approach in order to allow appropriate reservations. I.e. it must offer an interface that is strongly related to the reservation request interface of the ACA. It is therefore not generic but tailored for AQUILA purposes.
2. The API should also be as generic as possible in order to be widely accepted. It can be based on and/or can contribute to standardisation activities of the Internet community.

Therefore, we propose a twofold way: The EAT API, placed on the top of the EAT, is divided into two levels: the CORBA-based “internal” API and the Java-based “general” API (Figure 4-20).

The **Internal EAT API** summarises in a separate package all the interfaces of the EAT Manager towards applications and services which wants to directly access the EAT on an AQUILA-specific way. The graphical user interfaces of the EAT which are related to QoS reservations as well as the EAT’s scripting interface are already using it. The internal API is a proprietary interface designed for the specific purposes of AQUILA.

The **General QoS API** is a more generic one in order to provide QoS support not only based on the AQUILA’s RCL approach. The services that this API provides are more abstract and non-AQUILA-specific. Via the general API it should be possible to access different lower-level APIs such as the AQUILA’s internal EAT API, the RSVP API (RAPI), and the WinSock 2 API. For that, different “wrappers” are needed in order to map a general QoS request into a proprietary one.

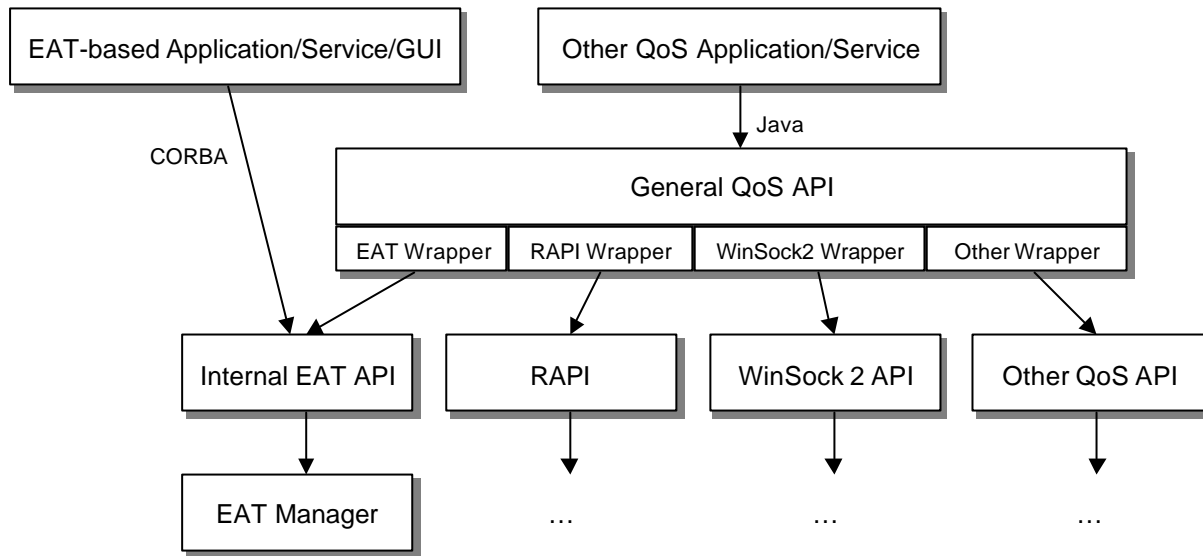


Figure 4-20: QoS APIs at different levels

The following sub-chapter presents the Internal API in detail. (The General QoS API will not be implemented for the second trial.)

4.3.3.1 Internal EAT API

The Internal EAT API represents those CORBA interfaces of the EAT Manager – the central component of the EAT – which are related to end-user authentication, network services, QoS reservations, etc. The aim is that all EAT-based external applications, services, tools, and graphical user interfaces use this API to have access to the EAT Manager’s features.

These features are based on the RCL’s, more specifically the ACA’s interface for user reservations. Consequently, the API provides similar interfaces for login, reservation request, reservation release, and logout. In contrast to the ACA, the reservation request, however, can be made in two modes:

- **Advanced reservation mode for specialists:** The request is made at the same level of abstraction as at the ACA, i.e. the full list of QoS, traffic, and flow parameters has to be specified. This mode is mainly for test and experimental purposes and for end-users/applications that are aware of the technical meaning of the request.
- **Regular reservation mode for non-specialists:** The request is made at a much higher level of abstraction. By using the pre-defined application profiles, the API provides abstract, well understandable QoS options to be selected by “normal” end-users. This mode should therefore be the *regular* way to request for QoS via the EAT.

The internal API, moreover, allows reservations for applications that dynamically negotiate data port numbers (e.g. via H.323) and/or use special protocols to establish a communication session (e.g.

SIP). In this case, the EAT Manager either asks the Proxy to complete the reservation request “form”, or the Proxy itself requests for a QoS reservation.

The model of the internal API considers the requirements above (Figure 4-21). The scenario for the usage is as follows (for more details refer to [D2203]):

1. Login is to allow end-user authentication. It also acts as a “factory” for a new `QoSSessionRequest` object when an end-user successfully logs in.
2. `QoSSessionRequest` is the basic handler for an end-user. It is to logout, to get information about the established SLAs (i.e. the by an end-user subscribed network services), and to allow reservation requests for groups and units of QoS sessions; both can be requested in the advanced as well as in the regular mode. Also the `QoSSessionRequest` interface acts as a factory for a new `QoSSession` object when a reservation request succeeds.
3. Depending on the kind of the request, a new `QoSSessionGroup` or `QoSSessionUnit` is created. Both specialise the abstract `QoSSession`, which is in general the representation of a reservation. It allows an abstract reservation release, for example.
 - 3.1 A `QoSSessionGroup` consists of other (sub-) groups or (at the deepest level) of session units. An existing `QoSSession` can join or leave this group. As shown in the figure below, the content of a `QoSSessionGroup` may not only be single `QoSSessionUnits` but could also be other groups. By using the composite pattern, the EAT API allows to build hierarchical group structures for multidimensional reservation groups. Such a group may consists, for example, of two dimensions like in a conferencing scenario: one for the calling partners, and one for the different service components such as “video” and “audio”.
 - 3.2 A `QoSSessionUnit` represents a unit of one or more inseparable reservation elements, for example one element for a unidirectional reservation, and two for a bi-directional one. Accounting data can be retrieved via this interface.
4. The aim of `EventObserver` is to have a reference to the QoS requesting application (the “requester”). It might be implemented by the client application, in order to be informed when something happens with the requested reservation (the `RequestEvent`). This mechanism is of importance mainly for so-called provisional reservations for which the Proxy detects the necessary flow data *after* requesting the reservation *and* then starting the application.

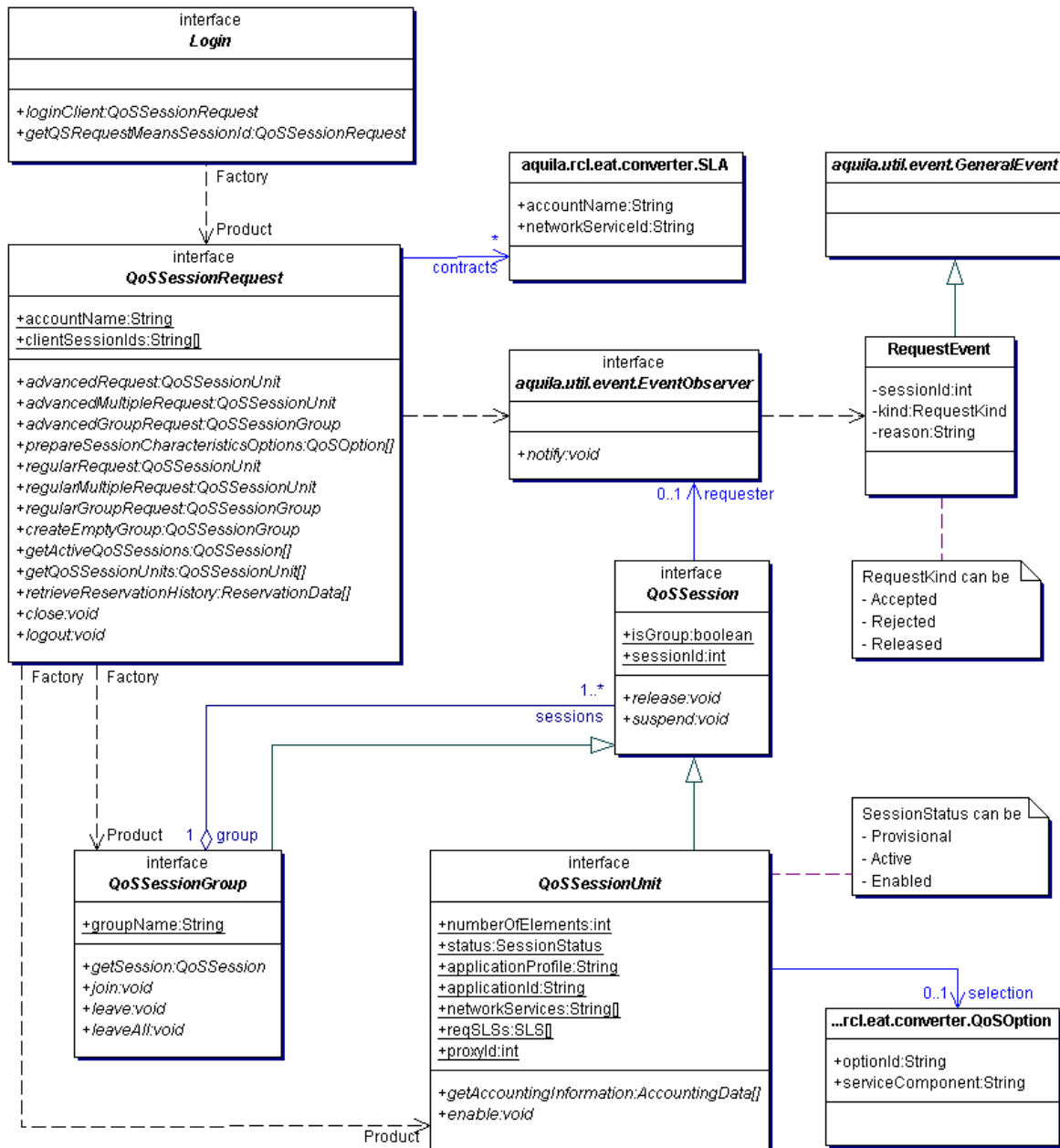


Figure 4-21: The Internal EAT API, part 1

Besides the above mentioned interfaces, two additional exist (Figure 4-22):

- ServiceDistributor shows the by the RCL provided network services, may be useful to negotiate a new SLA for an end-user.
- ApplicationManager shows the by the EAT provided application profiles (for regular requests), and the installed proxies (if needed by an application).

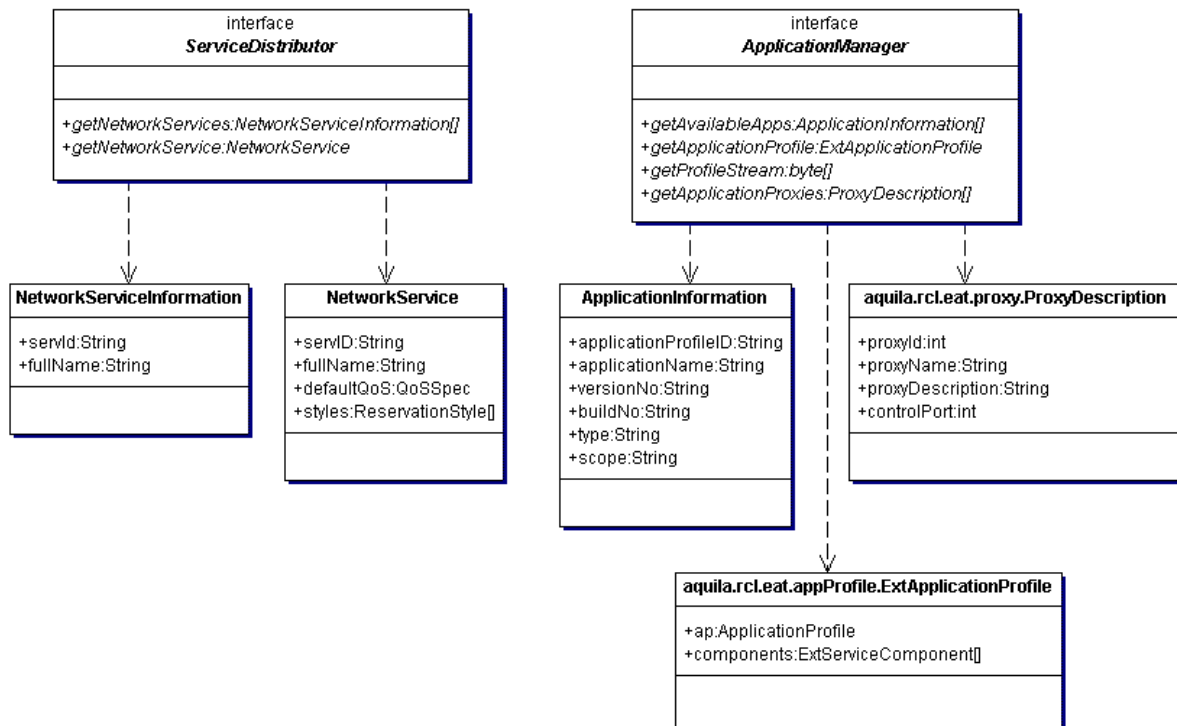


Figure 4-22: The Internal EAT API, part 2

4.3.4 GUIs, Converter, and Application Profiles

The legacy application support as well as the support of newly created complex Internet services are based on the inter-working of the GUIs, the Converter, and the application profiles: The GUIs correspond to the interface between the system and the end-user. The GUIs for normal end-users in the regular reservation mode are based on the application profiles and are dynamically produced using the session characteristic part of the application profile. The Converter takes care of the mapping between the session characteristics and the technical characteristics of the application profile in order to map between the user-friendly, abstract GUI QoS level, and the technical level of the AQUILA RCL.

In the following diagram, the scenario for a regular request is depicted, where all components (also the EAT Manager and the Proxy) are involved:

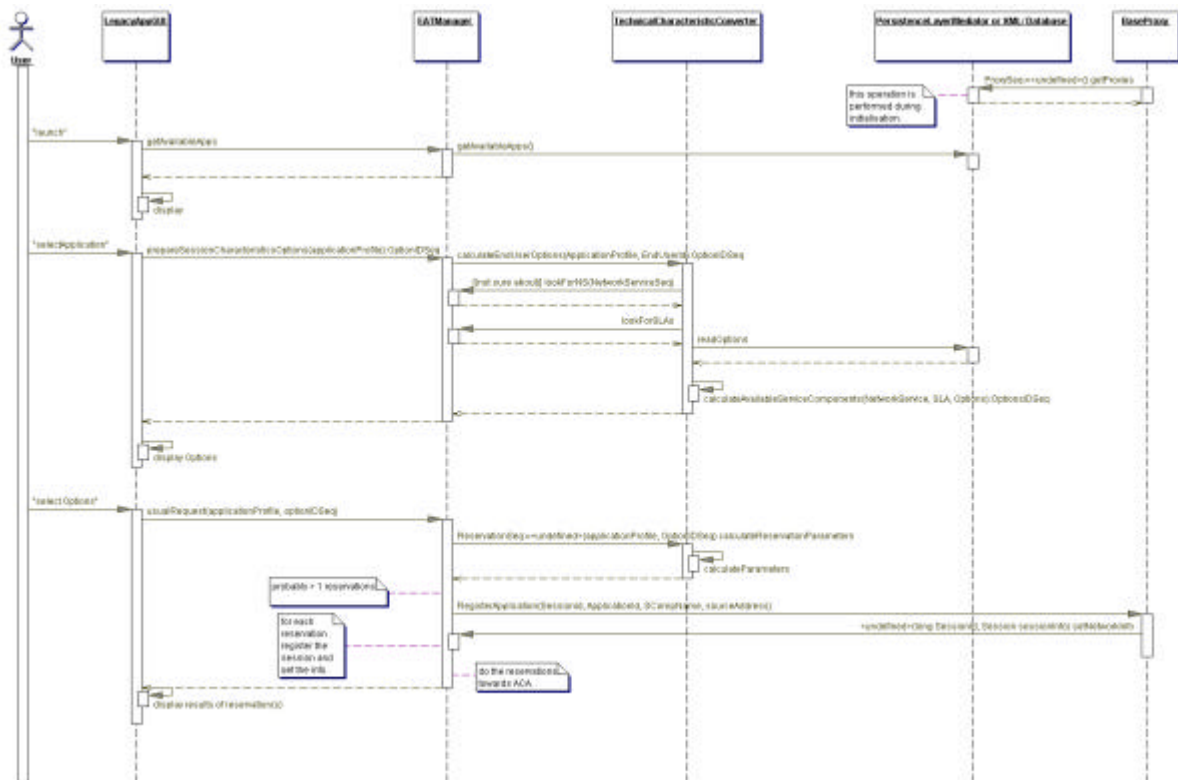


Figure 4-23: Sequence diagram for usual request

Firstly, for the calculation of QoS options for the end-user and its application, the Converter has to look for the existence of appropriate SLAs and network services and to read the possible options from the application profile (as shown above). Secondly, when the end-user selects an option, the Converter calculates the reservation parameters by using the profile.

4.3.5 Protocol gateways

4.3.5.1 Introduction

One of the important objectives of the AQUILA architecture is to transparently support legacy applications. In general, legacy applications cannot be modified in order to use a particular QoS API. Moreover, the parameters needed for a QoS request for such an application (e.g. TCP/UDP port numbers, traffic profile) cannot be examined or even influenced. Therefore, the EAT should use a variety of mechanisms in order to detect or measure those parameters.

In the AQUILA architecture, a reservation request from the EAT to the ACA comprises various parameters: source and destination addresses and ports of the IP flow, its traffic profile in the form of a TSpec as well as its QoS requirements (selected by the user only in the second trial). In the case of legacy applications, not all parameters are known to the user, so that a manual reservation can be made. To this end, the EAT provides several mechanisms that provide us with those parameters.

Application Profiles have been defined, that capture the QoS needs of a particular application, by providing the necessary set of parameters for a QoS request. Protocol Gateways or Proxies are used for the detection of TCP/UDP port numbers of application flows.

Many legacy applications open data connections using well-known port numbers, or ports that can be pre-configured to a known value. However, other applications communicate with dynamically negotiated port numbers. Such applications are mainly IP telephony or videoconference services that use connection establishment protocols such as SIP or H.323. Nevertheless, the same concept applies also to other applications that communicate through standard connection protocols (e.g. streaming applications that use RTSP). In order to make a resource reservation for such an application, the port numbers of each of the application flows should be known at the time of the QoS request. However, those parameters are negotiated between the communication entities during the connection set-up. Moreover, there is often no way for the user to manually select a particular port number.

What is needed, is the interception of connection set-up messages and the detection of those parameters. Then they are passed over to the EAT for the completion of a QoS request. The EAT uses the entities called Protocol Gateways or Proxies in order to get those parameters.

4.3.5.2 Protocol Gateways in general

4.3.5.2.1 Functionality

The protocol gateways used in AQUILA act as Application-level Proxies. The Application-level Proxy resides between the application and the network. In general, such a proxy interferes with the connections between two hosts. The client host, instead of connecting directly to the server, contacts the proxy. The proxy, in turn, connects with the server and relays packets coming from the client, to the server and vice versa. In this way, the port numbers of the Proxy – server connection are known to the Proxy, who is able then to make a QoS reservation for the IP packets of this connection.

In AQUILA, Protocol Gateways do not just participate in the signalling message exchange between client and server, but also they interpret the exchanged packets. By intercepting and interpreting the exchanged messages, the protocol gateways can find all the important information pertaining to a reservation. In the general case, this information is the source and destination addresses and ports of the flows that carry user data. In special cases, as we will see next, this information may be broader, covering also information about the traffic profile of those flows.

For a Proxy to operate properly, the communicating entities must have been configured to use one, so that all connection set-up passes through it. If there is no such option, the signalling messages should be intercepted by a filter and forwarded to the appropriate proxy.

4.3.5.2.2 Internal architecture

The Proxy package mainly consists of a central entity called ProxyManager and a number of application-level proxies. All proxies implement the same interface towards the Proxy Manager module. The ProxyManager controls the other classes and communicates with the EAT Manager. The application-level proxies communicate only with the ProxyManager. The internal architecture of the Proxy as well as its interaction with the EAT Manager module is presented in the following diagram.

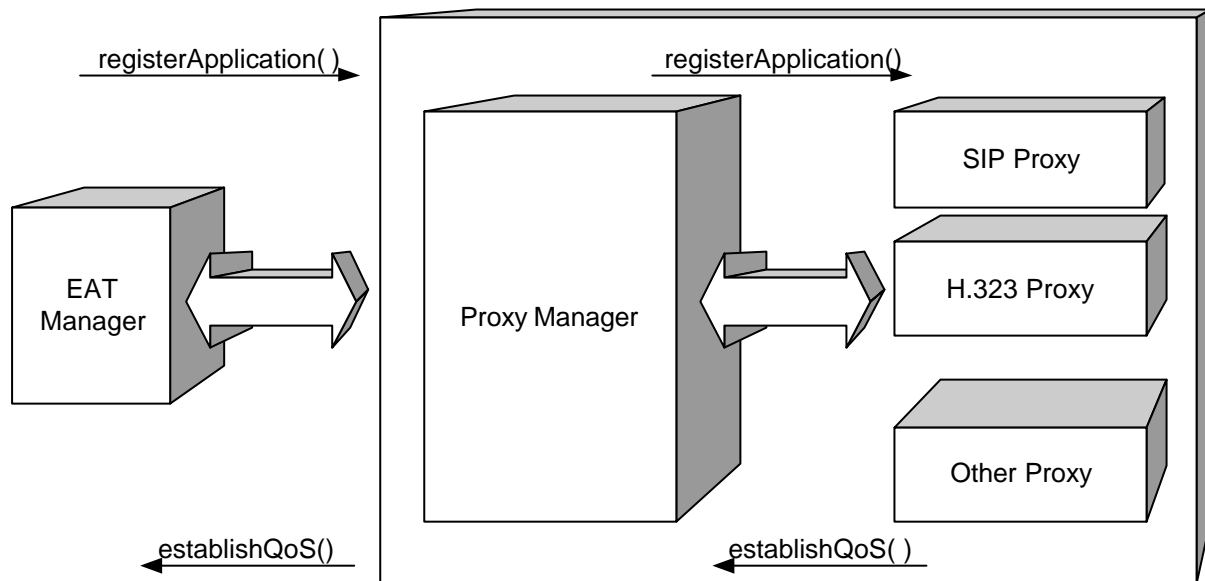


Figure 4-24: Proxy Architecture

This mechanism allows the installation of “plug-able”, remote application proxies for an EAT: A new application proxy registers itself at the ProxyManager when it starts during the runtime of the EAT.

4.3.5.2.3 Operation

In the general scenario of operation followed during the first trial, the EAT Manager asks the Proxy to be alert for the launching of a new application. This is performed by instructing the Proxy to expect a new flow of a specific service component (audio, video, or data) from a specific host. The Proxy Manager forwards this request to the appropriate Protocol Gateway, according to the instructions of the EAT Manager. When the application initiates a new connection, it will go through the Proxy. After the particular proxy has translated the connection establishment messages and detected the necessary addresses and port numbers, it feeds them to the Proxy Manager, who in turn, forwards this information to the EAT Manager.

The Protocol Gateways should also be able to operate on their own. When they detect the establishment of a new connection they should automatically notify the EAT. The EAT then, will notify the end-user and ask for permission to make a reservation for this session.

However, this process may be prolonged and the connection will have started for quite a long time until the user is notified for his approval. Moreover, due to the web-based architecture of the EAT GUIs, it may be difficult even to find a way to asynchronously notify the user. Therefore, in order to simplify this process, the EAT stores some user preferences that allow it to set automatically reservations for QoS-needy traffic, especially IP telephony, with pre-arranged traffic parameters.

4.3.5.3 Case study: SIP Proxy

4.3.5.3.1 Overview of SIP

The Session Initiation Protocol (SIP) is a signalling protocol for Internet conferencing, telephony, presence, event notification and instant messaging. The protocol initiates call set-up, routing, authentication and other feature messages to endpoints within an IP domain. SIP is independent of the packet layer and only requires an unreliable datagram service, as it provides its own reliability mechanism. While SIP typically is used over UDP or TCP, it could, without technical changes, be run over IPX, frame relay, ATM AAL5 or X.25. SIP is a text-based protocol, resembling the hypertext transfer protocol (HTTP) and simple mail transfer protocol (SMTP). SIP uses Session Description Protocol (SDP) for media description.

SIP follows the client/server architecture. The main entities in SIP are the User Agent, the SIP Proxy Server and the SIP Redirect Server. The User Agents, or SIP endpoints, function as clients (UACs) when initiating requests and as servers (UASs) when responding to requests. User Agents communicate with other User Agents directly or via an intermediate server. The User Agent also stores and manages call states.

SIP intermediate servers operate either as proxy or as redirect servers. SIP Proxy Servers forward call signalling from the User Agent to the next SIP server or User Agent within the network. SIP Redirect Servers respond to client requests and inform them of the requested server's address. In this way, User Agents may connect directly to the desired User Agent. To maintain scalability, the SIP servers can either maintain state information or forward requests in a stateless fashion.

4.3.5.3.2 Requirements for a QoS SIP Proxy

For the purposes of the AQUILA project, a SIP Protocol Gateway is used that implements a subset of the functionality of a SIP Proxy server. The main tasks of a SIP Proxy (or Redirect) Server are:

- to find the location of a user (to find the specific workstation/PC that the called user is currently logged on)
- to pass the resulting connection through a firewall.

More than one proxy may lie in the signalling path from the caller to the called party. A general scenario is depicted in the following figure:

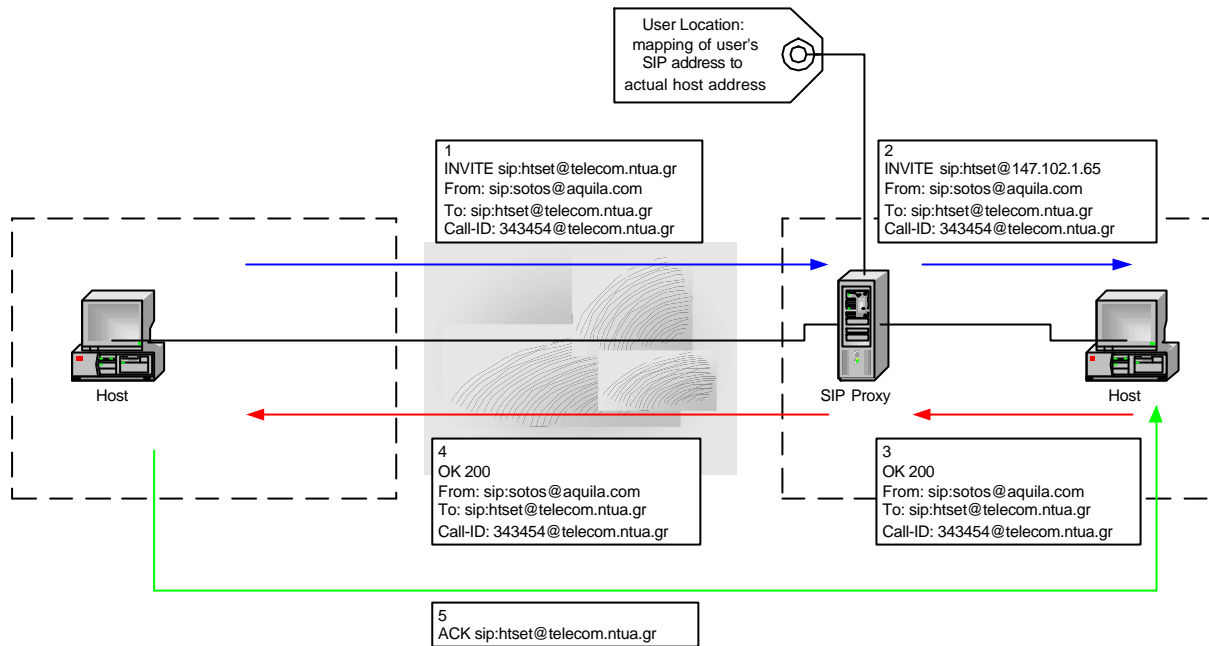


Figure 4-25: SIP scenario

The required information for a QoS request includes the port numbers of the audio/video connection(s). Those parameters are negotiated between client and server User Agents during the set-up of the connection. An example exchange of SIP messages for a simple audio call is presented in the following (taken from [RFC2543]):

```
C->S: INVITE sip:watson@boston.bell-tel.com SIP/2.0
Via: SIP/2.0/UDP kton.bell-tel.com
From: A. Bell <sip:a.g.bell@bell-tel.com>
To: T. Watson <sip:watson@bell-tel.com>
Call-ID: 3298420296@kton.bell-tel.com
CSeq: 1 INVITE
Subject: Mr. Watson, come here.
Content-Type: application/sdp
Content-Length: ...

v=0
o=bell 53655765 2353687637 IN IP4 128.3.4.5
s=Mr. Watson, come here.
c=IN IP4 kton.bell-tel.com
m=audio 3456 RTP/AVP 0
```

... skipped some messages ...

```
S->C: SIP/2.0 200 OK
Via: SIP/2.0/UDP kton.bell-tel.com
From: A. Bell <sip:a.g.bell@bell-tel.com>
To: <sip:watson@bell-tel.com> ;tag=37462311
Call-ID: 3298420296@kton.bell-tel.com
CSeq: 1 INVITE
Contact: sip:watson@boston.bell-tel.com
Content-Type: application/sdp
```

```

Content-Length: ...

v=0
o=watson 4858949 4858949 IN IP4 192.1.1.2.3
s=I'm on my way
c=IN IP4 boston.bell-tel.com
m=audio 5004 RTP/AVP 0
  
```

In this example, the calling party requests an audio connection (INVITE message), stating that he will receive on port 3456 and proposes to use the PCMU (ITU-T μ -law) codec during the call. In its final message (200 OK), the called party does not pose any objection to the port number, and says that it will use port 5004 to receive its own data.

A SIP Proxy is used, located in the caller's premises, that intercepts the messages and finds those port numbers. The Proxy also informs the EAT about the codecs used during the session, so that a correct reservation is requested. The above scenario is depicted in the following figure:

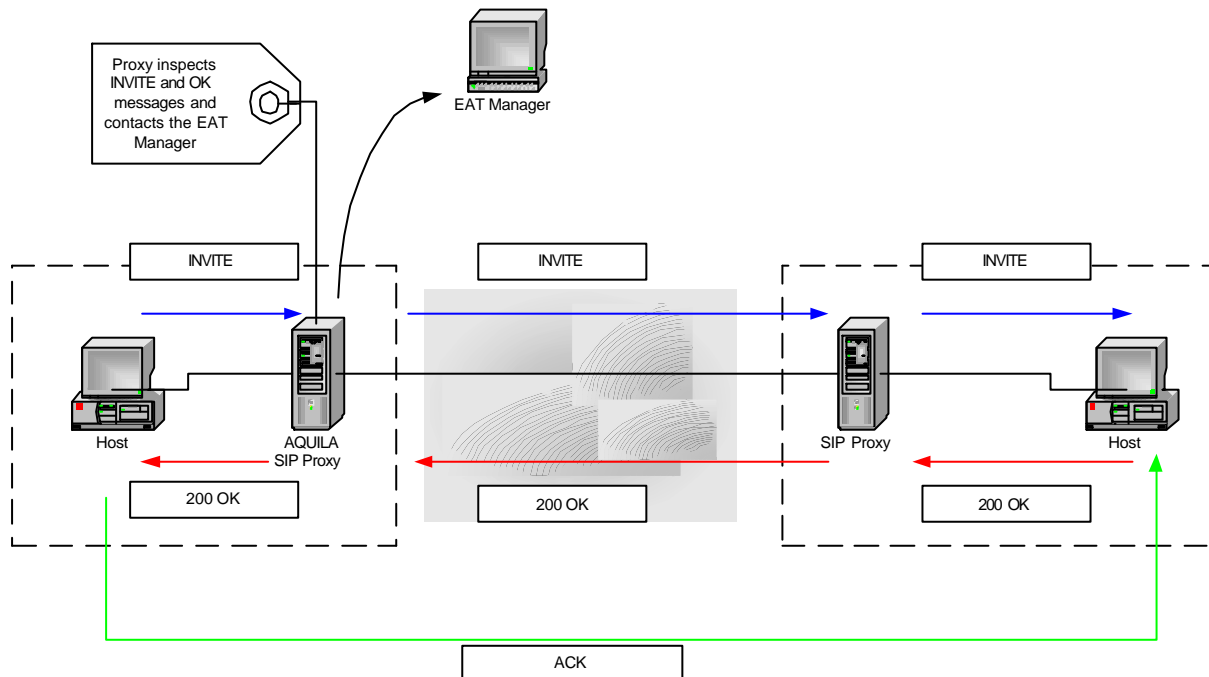


Figure 4-26: QoS SIP scenario

For the specification of the QoS SIP Proxy full compatibility with the current SIP Proxies used in the market today is pursued. As a matter of fact, the QoS SIP Proxy is specified as an add-on to current SIP Proxies and the intricacies of real proxies have been kept in mind.

However, for reasons of simplicity, some assumptions will be made. The Proxy does not have the full functionality of a SIP Proxy, especially as far as User Location services or firewall traversal are concerned (out of the focus of AQUILA). Finally, the SIP Proxy does not interfere in the SIP message exchange, unless it is needed. For example, it does not “fork” an invitation, but it alters the message headers (e.g. the “via” header) so that the backward flow of messages passes through it.

4.3.5.3.3 SIP Proxy Operation

4.3.5.3.3.1 RFC 2543

The objective of the integration of SIP and resource reservation is to *ensure* that an IP call session will receive the requested resources, before the call is started, that is, before the called party answers the call. In other words, the resource reservation procedures must have completed successfully (or not), before a 200 OK method is sent back from the called to the calling party. There are two kinds of sessions: **QoS-Assured** and **QoS-Enabled**. An QoS-Assured session will complete only if the resources are available and allocated. If not, the network will not complete the call. In a QoS-Enabled session, the resources may or may not be allocated. In any way, the session will be completed and the call will start.

The current specification of SIP in RFC 2543 does not foresee provisional responses, in a way that the call initiation could be partially completed, awaiting the reservation procedure to respond. Moreover, the SDP information from the called party is included only in the 200 OK responses. This method is sent immediately *after* the called party answers the phone. As a result, the session will start without the knowledge if the reservation was honoured or not. The exchange of messages in this case was shown in the previous figure.

A QoS SIP Proxy would, in this case, provide only QoS-Enabled sessions. This is acceptable in the context of the AQUILA architecture, since there is no official extension up to now that has reached RFC status. This Proxy will operate and provide the port number information in two ways:

- on demand from the EAT Manager
- upon detection of a new SIP connection, according to installed preferences

4.3.5.3.3.2 Extensions to SIP for QoS support

In order to explicitly support QoS for IP telephony, a number of Internet Drafts have emerged lately. Most of those drafts have been replaced by draft-ietf-sip-manyfolks-resource-01 [SIPRES]. In this draft, it is suggested that connection establishment and resource reservation should in general be decoupled. That is, SIP should not be used also for resource reservation, but a single QoS mechanism should be used for all applications. However, a total de-coupling is not feasible, since vital information that is used for resource reservation (addresses, ports, codecs) is not known in advance, but is dynamically allocated.

This draft specifies some extensions to SIP and SDP. A new message is added (COMET method), used to confirm the successful reservation of resources and allows the completion of a connection. Also, two new SDP attributes are defined: “qos” and “security”. The qos attribute indicates whether end-to-end resource reservation is optional or mandatory and in which direction (send, rcv, sendrcv). An example SDP description, carried in an INVITE message is the following. In bold fonts are the proposed additions:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
m=audio 49170 RTP/AVP 0
a=qos:mandatory recv confirm
m=video 51372 RTP/AVP 31
a=secure:mandatory sendrecv
m=application 32416 udp wb
a=orient:portrait
a=qos:optional sendrecv
a=secure:optional sendrecv
```

In the draft, it is proposed that after both User Agents have agreed on the parameters of the connection (ports and codecs) they engage in the reservation procedure. In contrast to the SIP specification, the called party responds to the INVITE method with a 183 method that also contains an SDP description of the audio stream the called party is about to send. At this point, a provisional acknowledgement is exchanged and reservation is about to start.

In draft-ietf-sip-manyfolks-resource-01 [SIPRES], the chosen reservation mechanism is the RSVP protocol. After the exchange of provisional acknowledgement is completed, the hosts start exchanging RSVP messages. Only after resources have been definitely reserved, may the SIP connection be completed: the called party's phone will ring and upon his/her answer, the voice packet exchange will start. In the following figure an example exchange of packets is illustrated.

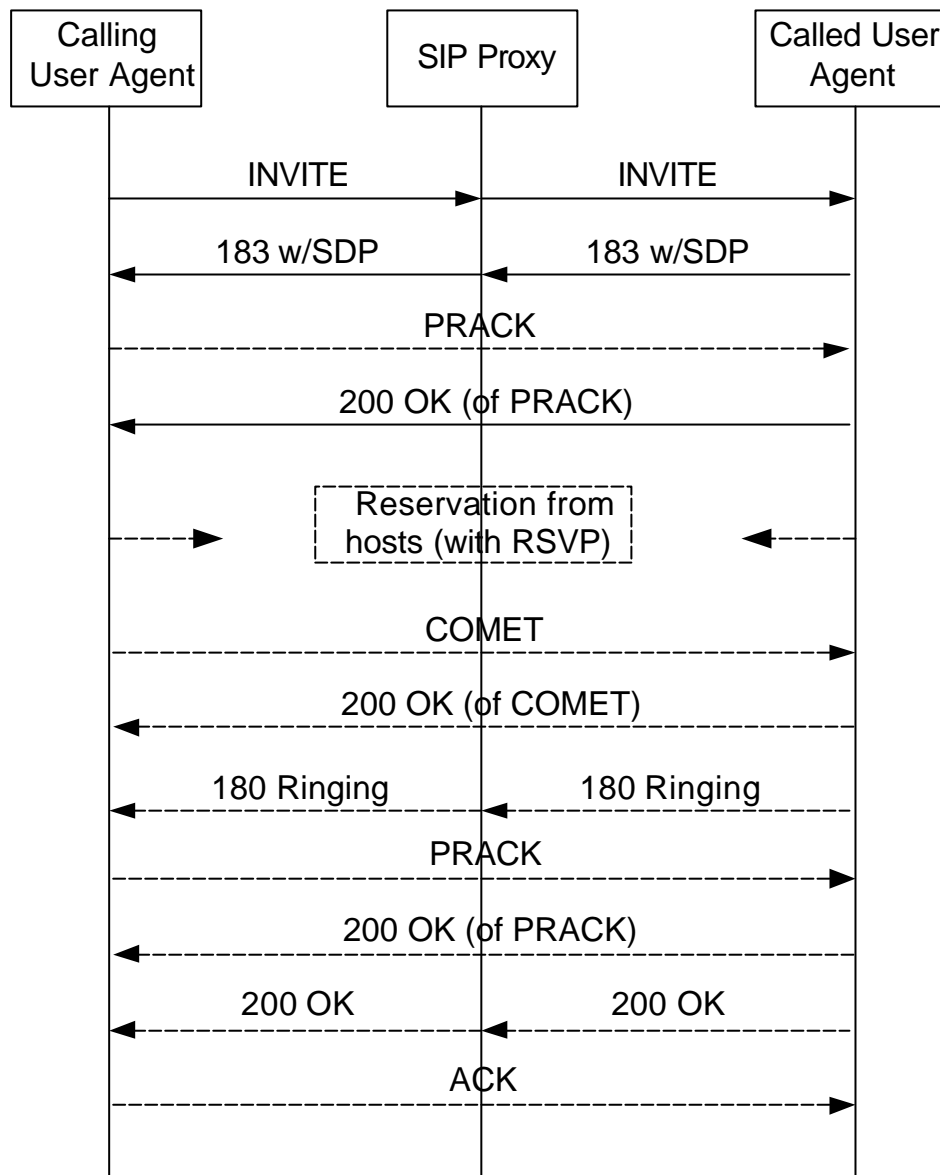


Figure 4-27: Sample message flow

The proposed AQUILA Proxy solution seems to be aligned with the approach envisioned for the support of resource management for SIP-initiated calls. In AQUILA, the Proxy takes part in this message exchange and invokes the resource reservation mechanisms of AQUILA on behalf of the hosts. The definition of the new SIP messages and SDP attributes proposed in draft-ietf-sip-manyfolks-resource-01 [SIPRES] will enable the Proxy to provide QoS-Assured sessions, in addition to the QoS-Enabled ones.

The approach depicted in draft-ietf-sip-manyfolks-resource-01 has some limitations, in comparison to the proxy approach. First of all, the hosts should support the RSVP protocol (or another QoS signalling protocol) if they are to make QoS requests. It is questionable whether simple SIP phones will be capable of running also an RSVP daemon. Moreover, non-RSVP aware (and non-QoS

aware in general) hosts will not be able to leverage the network's QoS functionality. Finally, this solution ties SIP closely with RSVP, even though it is not certain that RSVP will be the actual QoS mechanism in the future networks, given its scalability limitations.

On the contrary, in the proxy approach, QoS functionality is present only in the network and is not required by the end-hosts. Moreover, it can be used with a variety of reservation mechanisms (AQUILA, Bandwidth Brokers in general). However, if this approach is accepted within the IETF as an RFC, the effect of this standard on the EAT architecture should be investigated.

4.3.6 Deployment Scenario

The EAT will reside in the access network, near to the host, and at the edge of the by the RCL controlled core network (Figure 4-28). It is not necessary to install the EAT on the end-user's host, because he/she can access the EAT via a web browser. For that, the EAT interacts with a Web server near to it which supports Java Server Pages (JSP) and Java servlets for the dynamically created GUIs/web pages of the EAT.

These GUI servlets access the EAT via the internal (CORBA) API. Also QoS-aware, EAT-based applications may use this API to get QoS from the RCL. (In the case that the general QoS API will be used, this API will be located at the end-user's host and access the internal EAT API via CORBA, too.)

The Proxy resides also in the access network between the host and the edge router in order to act as a protocol gateway getting the for QoS requests relevant data from application's flow, particularly the control flow (SIP, H.323, RSVP). Between the EAT (Manager) and the Proxy CORBA is used, too, so that they can run on different machines.

The EAT, particularly the Converter, use the application profiles for the mapping of user-friendly QoS terms into RCL-compatible requests. The profiles are accessible via LDAP and can therefore be installed in a central directory server.

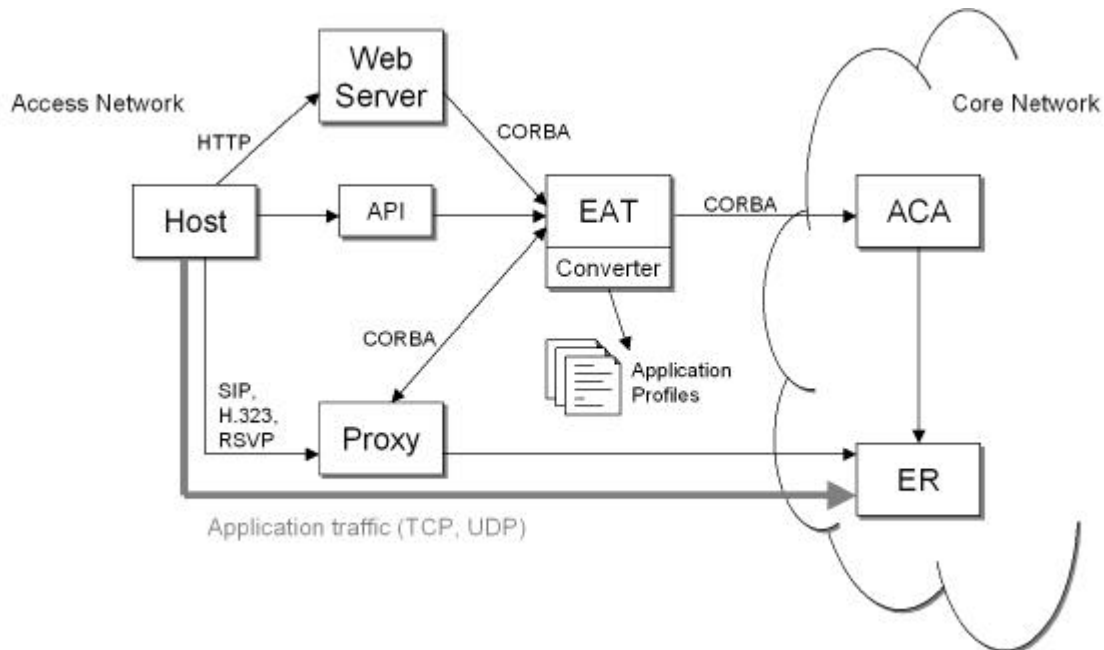


Figure 4-28: Deployment scenario of the EAT components

Generally, there is not need to install an EAT in every access network, since each EAT can act as requester for any host.

For complex Internet services (CIS) that provide services with QoS supported by AQUILA, it is reasonable to have an EAT near to this web site (Figure 4-29). Such a CIS, for instance the Medi-azine application, acts, on the one hand, as customer/end-user regarding QoS requests towards the RCL, and provides, on the other hand, its media services including QoS features to its human end-users².

² Concerning the charging & accounting: First of all, the CIS is in charge to pay for the QoS from the RCL. However, it can forward costs to its human customers.

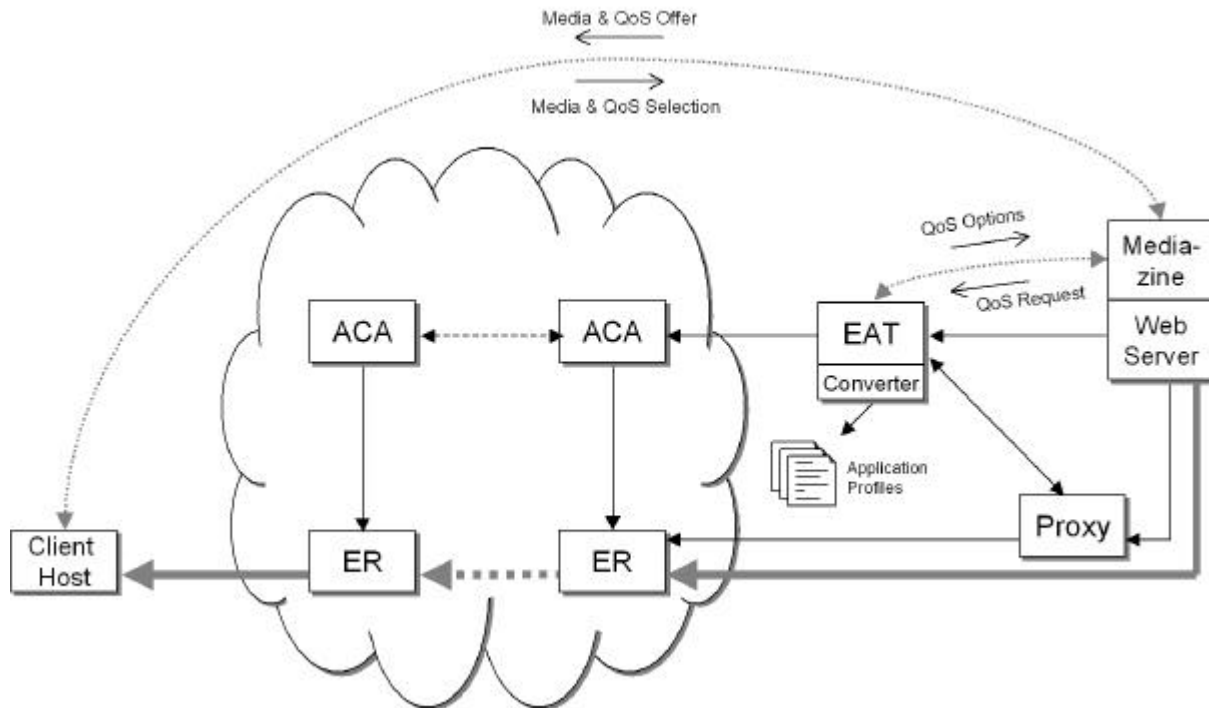


Figure 4-29: Deployment scenario for the Mediazine example

4.4 Management

QMTool comprises the management tool, which will mainly serve the configuration of all components belonging to both the RCL and IDOM layers. In particular, it provides the means to the administrator of the AQUILA network to design the various components of the network, to determine their relations and to configure them.

As far as the design functionality is concerned, QMTool offers a graphical user interface where it permits the graphical representation of the various network elements and moreover the determination of their relations. In particular, the elements that can be graphically depicted are the Resource Pools (RCA), the Admission Control Agents (ACA), the End-user Application Toolkits (EAT), the Proxies and the BGRP Agents. It is assured that the QMTool controls this task in the way that it will not allow for the creation of two components that are not actually related. Therefore, this graphical representation will definitely reflect one possible design of the network.

Apart from the design, which will obviously provide a clearer view of the network, it is its configuration, which will bring it into effect. In other words, the design of the network needs to be accompanied by the configuration of its components. This implies that QMTool provides the means to set the necessary parameters for each aforementioned component. This setting is performed with the use of XML files, which represent the configuration information of each element. These files are finally stored at the LDAP database at a prior defined location.

In order to give a more practical description of this functionality, let's imagine the configuration of a resource pool. The user is able to define or edit the DTD file on which the XML configuration file of the resource pool will be based. Next, by right-clicking on the resource pool component and by selecting the "configuration" option, the user has the ability to specify an existing XML file for the resource pool's configuration, to create a new one or to view the already performed configuration. In all three cases, an XML editor (XMLOperator is used) appears that loads either the existing file, a new one according to the given DTD or the file stored at the LDAP for this component respectively. At this point, the user is able to edit the configuration information. The final step is the storage (re-storage if already stored) of the file to the LDAP server and the configuration of the component is considered complete. It is worth noting that QMTool will perform some basic checks concerning the configuration in order to verify that there is no inconsistency between the configuration and the graphical representation of the network components (specifically in the determination of their relations).

Apart from the configuration of the visualised components, QMTool enables the configuration of some further elements that are necessary for the functioning of the AQUILA network. Those are, for example, the Network Services, the Traffic Classes, the Application and Service Profiles, the Globally well known services etc.

As it can be concluded from the above, the basic modules of QMTool are the GUI and the XML editor, which serve the visualisation and configuration of the network. Apart from this functionality, QMTool, since comprising a management tool, will provide failure detection and monitoring capabilities. As far as failure detection is concerned, the components of both the RCL and IDOM layers will be continuously contacted in order that the administrator is aware of the possible failure of a network component. The notification of the administrator will be performed for example by rendering the component's colour to 'red'. In the same way, as soon as the component has started running again, the colour indicating its normal functioning will be re-established. Failure detection for a component will only start when the user triggers this functionality.

Concerning the monitoring functionality, QMTool will enable the monitoring of the resources of Resource Pools and BGRP Agents. This resources' information is provided to QMTool in the form of XML files. QMTool will use the JAXB technology in order to transform those files to Java Objects and retrieve more easily the monitored data. The XML files will contain data that require both graphical and textual representation and as a consequence, an indication of the format (attribute) will be always given to QMTool. As is the case with failure detection, monitoring will only be performed when it is triggered by the user. Moreover, it will be polling-based where the user will further have the chance to set the polling intervals. In the same way, the user may easily stop the monitoring of a component.

Obviously, for failure detection and monitoring, some interfaces are needed from the corresponding components in order that QMTool communicates with them. Those are the "AlivePeer" interface, which is already used for the deployment of the keep-alive mechanism among the AQUILA network elements and the "Monitoring" interface for the retrieve of the XML file's data. The paths where the monitored objects (those implementing the Monitoring interface) are registered to the CORBA

Name Server are known and therefore QMTool can easily retrieve the reference of them and continue with their monitoring. However, this is not the case with the AlivePeer CORBA objects and for that reason these references should be communicated to QMTool. Consequently, the “Monitoring” interface will further provide this functionality, i.e. the communication of the AlivePeer objects to QMTool.

The description of the interfaces between QMTool and the RCL/IDOM components is depicted in the following figure.

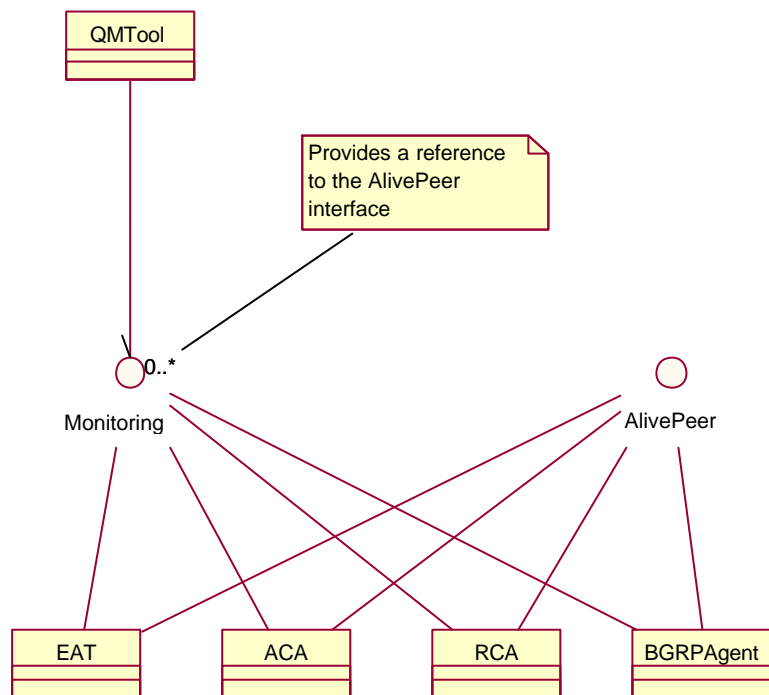


Figure 4-30: QMTool interfaces

QMTool, which is the main class, has a reference to the Monitoring interface implemented by the RCL and IDOM components. From this interface, it can retrieve the data that will be monitored and this applies in particular to the RCA and BGRP Agent. Moreover, through the same interface, QMTool will acquire the references to the AlivePeer interfaces implemented by the above-depicted elements. In this way, it will be able to perform their failure detection.

4.5 Supporting entities

4.5.1 Router control

Several entities in the AQUILA architecture need access to the network elements (edge devices, core routers, border routers). The following list gives just some examples:

- The ACA has to install and remove classifiers and policers in the edge devices.
- The BGRP agent needs access to the BGP routing table to query the next hop for a given destination.
- The distributed measurement architecture needs access to statistical data in the core routers.

For all these components, a common software package for router access is used. The task of this router interface is to set-up and maintain the connection to the router, to send messages to the router and evaluate the responses, to provide an abstract view of the router internals (router model), and to provide specific application interfaces for various parts of the AQUILA architecture. According to these assignments, the router interface is modelled in a four-layered approach:

- **Application**
This layer implements the interfaces to the other AQUILA components. It is fully independent of the underlying router architecture and type and the used router software.
- **Router Model**
This layer builds an abstract view of the router architecture. It contains a software image of the relevant parts of the router configuration and synchronises this software image with the actual router settings by issuing commands and evaluating the responses.
- **Message**
This layer builds the commands of a specific router software and parses the responses. Generally it depends on the software version in the router (Cisco IOS version).
- **Transport**
This layer implements the reliable transport of messages using a transmission protocol like telnet or tftp.

This approach provides the flexibility to access different router types using a variety of software versions and transmission of protocols. All layers are configurable and can easily be adapted to different needs.

4.5.2 Pluggable algorithms

There are two places in the AQUILA architecture, where a set of algorithms is proposed:

- For resource control, several algorithms may be used to calculate the amount of additional bandwidth to be requested or the size of the resource cushion to be kept while releasing resources. These are called “RcRules”.
- For admission control, AC algorithms and the way, how parameters are retrieved, may vary. Especially, a declaration based admission control (DBAC) and a measurement based variant (MBAC) may be used.

Both are candidates for pluggable algorithms, which can be configured through management. Basically, the configuration involves a parameter, which names the actual instance of an algorithm. The JAVA based implementation makes it easy to instantiate the named classes from the configured names during start-up. As an example, the structure of pluggable resource control algorithms is detailed below.

4.5.2.1 Pluggable resource control rules

The general structure of the resource pool and the hierarchical order is shown in Figure 4-31. The following describes the resource handling inside the ACA Resource Component (RC) and the usage of the CORBA interface up to higher resource pool layer covered by the Resource Control Agent (RCA)

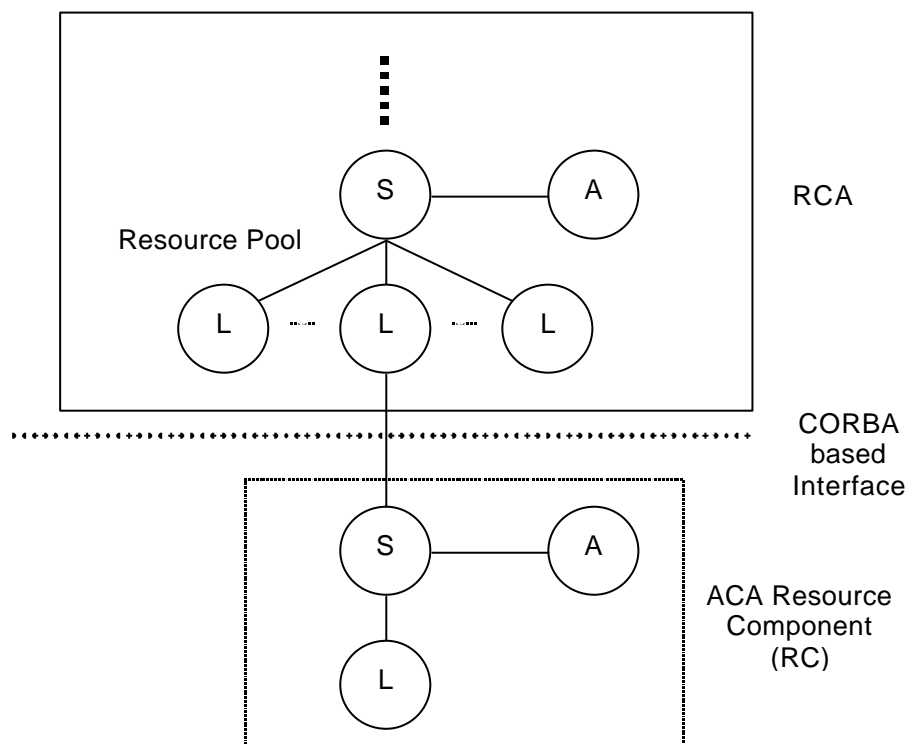


Figure 4-31: resource pool hierarchical structure

Each resource pool consists of three components:

- Share (S)
- Algorithm (A)
- Limit (L)

The **Limit** stores the status information Last Requested Limit, Last Requested Wish and Last Assigned Limit and offers an interface in order to modify a current stored bandwidth limit. Especially at

the ACA's RC the required bandwidth, calculated by the admission control component, has to be requested at this **Limit**. The Limit also provides an interface to retrieve the stored status information

The **Share** offers an interface to receive the modification request from the **Limit**. The **Share** itself calls the **Algorithm**'s interface in order to trigger a re-calculation of the bandwidth limits. The **Share** also offers an interface for the **Algorithm** in order to permit a modification of the bandwidth limits and forwards this modification request to the higher level of the resource pool hierarchy especially to the RCA Limit component bundled with this RC component.

The **Algorithms** solve three different questions:

- Whether new bandwidth has to be requested or released from/to the higher resource pool level?
- How much bandwidth has to be released?
- How much bandwidth can be given to the requester share?

The lowest layer in this overview is the resource component of the ACA and has a little bit different meaning as the layers above. It is not a resource pool in the original meaning because no lower Resource pool consumes bandwidth. Therefore, that resource pool does not contain any lower level Limits. At this lowest level, the result of the admission control calculation has to be requested as new bandwidth amount.

Nevertheless, this lowest component of the resource pool layer is designed like the layers above. The inter working between the Limit, the Share and the Algorithms at this lowest layer is perhaps only a subset of the functionality that is supported in the higher levels but should not differ from that.

To each of the above questions different algorithms can be used. In our case, not all the possible algorithms will be implemented. Concerning the amount of bandwidth that an upper level component will assign to its child, the algorithm (adaptive low watermark) already realised in the first trial and described in [D3201], will be used. The amount that is going to be released to the upper resource pool is calculated based on the Leaky Share algorithms. The last set of algorithms answers the question "whether" new resources should be requested/released from/to the upper level. The request process will be initiated each time the currently assigned resources are not capable of accommodating the new request. The release process will be initiated using two different approaches; one timer-based and one counter-based. In the former a release process is initiated each release period T and if a specific amount of resources are not used, and in the latter the process is initiated after a specific number x of release requests.

5 Abbreviations

A

AC	Admission Control
ACA	Admission Control Agent
API	Application Programming Interface
AS	Autonomous System

B

BGP	Border Gateway Protocol
BGRP	Border Gateway Reservation Protocol
BR	Border Router

C

CBR	Constant Bit Rate
CDN	Content Delivery Network, Content Distribution Network
CIDR	Classless Inter Domain Routing
CLI	Command Line Interface
CORBA	Common Object Request Broker Architecture
CPE	Customer Premises Equipment
CR	Core Router

D

DB	Database
DBAC	Declaration Based Admission Control
DiffServ, DS	Differentiated Services
DSCP	DiffServ Code Point

E

EAT	End-user Application Toolkit
EBA	Effective Bandwidth Allocation
ED	Edge Device
ER	Edge Router

I

IETF	Internet Engineering Task Force
IPDV	Instantaneous Packet Delay Variation
IPPM	IP Performance Metrics
ISP	Internet Service Provider

L

LER	Label Edge Router
LSP	Label Switched Path
LSR	Label Switching Router

M

MBAC	Measurement Based Admission Control
MF	Multi-Field classifier
MPLS	Multi Protocol Label Switching

O

OWD	One Way Delay
-----	---------------

P

PHB	Per Hop Behaviour
PL	Packet Loss
PCBR	Premium Constant Bit Rate
PMC	Premium Mission Critical
PMM	Premium Multimedia
PRA	Peek Rate Allocation
PVBR	Premium Variable Bit Rate

Q

QoS	Quality of Service
-----	--------------------

R

RCA	Resource Control Agent
RCL	Resource Control Layer
RP	Resource Pool
RPL	Resource Pool Leaf

RPT	Resource Pool Tree
RTD	Round Trip Delay
S	
SIBBS	Simple Inter Bandwidth Broker Signalling
SIP	Session Initiation Protocol
SLA	Service Level Agreements
STD	Standard
T	
TCL	Traffic Class
TCP	Transmission Control Protocol
V	
VBR	Variable Bit Rate
VPN	Virtual Private Network
W	
WFQ	Weighted Fair Queuing

6 References

- [ABOBA] Aboba, B. and D. Lidyard, "The Accounting Data Interchange Format (ADIF)", Work in Progress.
- [ALTM] J. Altmann, and P. Varaiya, "INDEX Project: User Support for Buying QoS with regard to User's Preferences," IWQOS'98, Sixth IEEE/IFIP International Workshop on Quality of Service, pp. 101-104, Napa, USA, May 1998. http://www-networking.eecs.berkeley.edu/~altmann/frames/publications_frame.html
- [BGRP] BGRP: Sink-Tree-Based Aggregation for Inter-Domain Reservations Ping P. Pan, Ellen L. Hahne, and Henning G. Schulzrinne, KICS 2000
- [BISEL] Torsten Bissel, Manfred Bogen, Christian Bonkowski, and Dieter Strecker. QoS Assessment and Measurement for End-to-End Services. Proceedings of the First COST 263 International Workshop on Quality of Future Internet Services (QofIS 2000). Jon Crowcroft, James Roberts, Mikhail I. Smirnov (Eds.), Lecture Notes in Computer Science 1922, Springer-Verlag, ISBN 3-540-41076-7, pg. 194-207, 2000.
- [Black99] Black, Darryl P.: Building Switched Networks – Multilayer Switching, QoS, IP Multicast, Network Policy, and Service Level Agreements, Addison Wesley, 1999
- [BLAN] J. Blanquer, J. Bruno, E. Gabber, M. Mcshea, B. Özden, and A. Silberschatz, "Resource Management for QoS in Eclipse/BSD", Proceedings of the FreeBSD 1999 Conference, Berkeley, California, October 1999. <http://www.bell-labs.com/project/eclipse/pub.html>
- [BOGEN] Manfred A. Bogen. A Framework for Quality of Service Evaluation in Distributed Environments. Ph.D. Dissertation, University of Nottingham, UK, September 2000, GMD Re-search Series, ISBN 3-88457-384-5, 2001.
- [BRUS] Jos'e Brustoloni, Eran Gabber, Abraham Silberschatz and Amit Singh, "Quality of Service Support for Legacy Applications", in Proceedings of the 9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99), Basking Ridge, NJ, June 1999, pp. 3-11. <http://www.bell-labs.com/user/jcb/> and <http://www.bell-labs.com/project/eclipse/>
- [COMA] Esteve Majoral-Coma, Xavier Martínez-Álvarez, Angel Luna-Lambies, Jordi Domingo-Pascual, "Design, implementation and test of a RSVP Agent based on a generic QoS API", Universitat Politècnica de Catalunya, Advanced Broadband Communication Center, <http://www.fokus.gmd.de/events/qofis2000/html/abstracts.html> -

[COM0900:Design](#)<http://www.fokus.gmd.de/events/qofis2000/html/programme.html>

- [D1201] IST-1999-10077-WP1.2-SAG-1201-PU-O/b0, System architecture and specification for the first trial
- [D1302] IST-1999-10077-WP1.3-COR-1302-RE-O/b1, Specification of traffic handling for the second trial
- [D2203] IST-1999-10077-WP2.2-TUD-2203-PU-R/b0, User Guide for End-user Application Toolkit
- [D3201] IST-1999-10077-WP3.2-TPS-3201-PU-R/b0, First Trial Report
- [Ferg98] Ferguson, Paul; Huston, Geoff: Quality of Service – Delivering QoS on the Internet and in Cooperate Networks, Wiley Computer Publications, 1998
- [ISABEL] ISABEL <http://isabel.dit.upm.es/>
- [QOSWG] Quality of Service Internet2 Working Group <http://www.internet2.edu/qos/wg/>
- [RFC1519] IETF RFC 1519:Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy
- [RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, RFC 2475 – An Architecture for Differentiated Services
- [RFC2543] M. Hadley et al., RFC 2543 – Session Initiation Protocol
- [RFC2597] J. Heinanen et al., RFC 2597: Assured Forwarding PHB Group
- [RFC2598] V. Jacobson et al., RFC 2598: An Expedited Forwarding PHB
- [RFC2638] K. Nichols, V. Jacobson, L. Zhang, RFC 2638: A Two-bit Differentiated Services Architecture for the Internet
- [SABA2] SABA2 http://www.ccaba.upc.es/projects/saba2/E_saba2local.html
- [SIBBS] QBone Bandwidth Broker Architecture, Work in Progress, <http://qbone.internet2.edu/bb/bboutline2.html>
- [SIPRES] W. Marshall et al., Internet draft: draft-ietf-sip-manyfolks-resource-06