Wireless Internet QoS

Vlora Rexhepi

Examination Committee:

Dr. Ir. Geert Heijenk

Dr. Phil Chimento

Prof. Dr. Ir. I. G. Niemegeers

Universiteit Twente

Telematics Systems and Services (TSS)

Department of Computer Science

P.O.Box 217, 7500 AE Enschede, The Netherlands

ERICSSON

Wireless Multimedia Research (WMR) Department

ERICSSON BUSINESS MOBILE NETWORKS BV

P.O.Box 645, 7500 AE Enschede, The Netherlands

Cover Design: Arian Rexhepi

Printed By: PrintPartners Ipskamp, Enschede, The Netherlands

Wireless Internet QoS

Thesis

towards
Master in Technological Design (MTD) diploma at the University of Twente, Department of Computer Science and Electrical Engineering

Vlora Rexhepi

September 28, 2000

Preface

*This thesis is part of the Designer Course **Tele-Informatics and Open Systems** curriculum given at University of Twente, Telematics Systems and Services (TSS) group. The Designer Course **Tele-Informatics and Open Systems** has the duration of two years leading to a certified diploma as Master in Technological Design (MTD). A prime aspect of this education is the ability to bridge the distance between research and production.*

The research work on this thesis was carried out at Ericsson Business Mobile Networks (EMN), Wireless Multimedia Research (WMR) department, during period 1st of October 1999 until 1st of October 2000. For its duration it has been part of the Quality of Service in the Wireless Internet Next Generation (Q-WING) project, which is a collaborative research project with the Centre for Telematics and Information Technology (CTIT) at the University of Twente, the Netherlands. Further, it is part of the Internet Next Generation (Internet NG) project, via which it is associated with the International QBone initiative, and the Dutch Gigaport program.

Abstract

The current trends in the development of real-time Internet applications and the rapid growth of mobile systems, indicate that the future Internet architecture will have to support various applications with different Quality of Service (QoS) requirements, regardless of whether they are running on a fixed or mobile terminals.

Enabling end-to-end QoS over the Internet introduces complexity in several areas starting from applications, network architectures, but also in network management and business models. It becomes even more complex when one is introducing QoS in an environment of mobile hosts, wireless networks and different access technologies, due to scarce resources. Consequently, QoS deployment in the Internet represents one of the most challenging research topics of computer networks community today.

The efforts to enable end-to-end QoS over the Internet have led to the development of two architectures, the Integrated Services architecture and more recently, the Differentiated Services architecture. Although fundamentally different, both architectures are designed for QoS support on the Internet. However, these architectures do not take into account QoS mobility issues, despite their importance, which may result in solutions that will not be able to cope with the requirements of future Internet services.

This thesis focuses on the interoperability between the Integrated and Differentiated Services architectures as well as on the QoS mobility issues. Its objective is end-to-end wired and wireless Internet QoS support.

The research work has resulted in a general Integrated Services / Differentiated Services architecture design with specific requirements and accordingly in a detail design of the boundary router. The role of this boundary router is to handle the Integrated and Differentiated Services interoperability, in a wired and wireless Internet environment. In order to prove the feasibility of the boundary router design a basic prototype implementation has been developed. Finally this thesis addresses the QoS and mobility issues and describes the applicability of the defined architecture in an environment of mobile hosts and wireless access networks.

Acknowledgements

I am very pleased that I have the opportunity to thank all the people that have guided and supported me not only during the last year spent on the work on this thesis but also during the past 26 months of my work as a Trainee Research Assistant (TWAIO) at University of Twente, which have been one of the best and most challenging months of my life and for some time the most difficult ones.

After earning my engineering degree at Faculty of Electrical Engineering, University of Prishtina I started my postgraduate studies there and was ending my first year when I got accepted at University of Twente. I dropped everything and moved to Netherlands. It was the best decision I had made so far.

My first contact with University of Twente was Kees Wiebering, whom I met while participating in a Pax Christi's youth project in Prizren, Kosova. Through him I applied for the TWAIO position at the University of Twente. Thank you Kees for the quality time you spent either in inquiring about the status of my documents or in trying to get through to Prishtina for a phone call.

I would like to thank Victor Nicola for his guidance in creating a perfect TWAIO curriculum for me, selecting a list of great courses (although the list was a bit to long) and for arranging this perfect project for my second year. Next to doing such a great job with my curriculum, he has been extremely supportive and all the time concerned with my well being especially during the months March-November 1999.

Special Thank You goes to my supervisors Geert Heijenk and Phil Chimento. They have shed a different light on research work than what I had thought it to be and made the work on this thesis a real pleasure. I was given a lot of freedom to choose my own way of working and at the same time I always had their support and guidance. Geert Heijenk with his expertise and kind personality was motivating all the way through. The feedback I got from him on our meetings on weekly basis, his patience to listening all my ideas (sometimes "silly") and his supervision were very stimulating. Phil Chimento with his enormous knowledge on the topic was there to clarify and give me the deep insight on my own ideas. Furthermore, even though he left University he agreed to stay my supervisor until the end, for what I am particularly thankful. Thank you both. I could have not wished for better supervisors.

I want to thank Prof. Dr. Ir. Ignas Niemegeers as the other member of my examination committee.

This thesis would have not been completed in time without the help of my colleagues from the Q-WING project. A very great thank you goes to Georgios Karagiannis, with whom we spent enormous hours in discussing all the problems related to the work on this thesis. These discussions resulted in the work presented in the last chapter. Georgios, thank you for being such a great colleague and friend. Without Simon Oosthoek's Linux expertise, with the limited time I had I would have not managed to have the necessary lab setup for testing and experiments nor to have a working prototype. Thank you Simon for being so helpful at all times. Also, a word of thank you goes to Martin van der Zee, for his valuable comments on my work.

I would like to thank my colleagues at the faculty of Computer Science, University of Twente and colleagues of the Wireless Multimedia Research (WMR) department at Ericsson Business Mobile Networks (EMN) for accepting me so well and making me feel at home even though being far away from it. In particular I would like to thank Mehmet Aksit, who is one of the greatest lecturers I have ever known and one of the most sympathetic people. His attention and care during the months March –November 1999 and his initiative to start up the project "Love unites, hate divides", made it easier for me to cope with the pain and despair I was feeling on what was happening in my country Kosova at that time. I am so very grateful for that.

Moving to Netherlands I was expecting to get acquainted with the Dutch culture and the Dutch people. I never expected that I will, apart from that get acquainted with cultures from all over the world Brazil, China, France, Portugal, Ecuador, Argentina, Albania, some of the places from where my newly gained friends are coming from. I thank you all for the fun times we shared together.

With my roommates at Dommelstraat 34 Mariken, Arjen and friends Karen and Kristel, I really experienced and enjoyed (although some times I was taken by surprise) the Dutch culture. It was a lot of fun. Thanks for all the happy times.

I would also like to thank my family and friends from Kosova and England for their tele – support.

A special and a very great thank you goes to Mariken, for sharing so sincerely the rough times I had during March-November 1999. Mariken, thank you, for your friendship and for being there for me. I feel the deepest respect for Ton and Mieke Ederveen and I want to thank you both for your kindness.

To my brother Alban and to Aida, thank you for your love and care.

A big hug to my brother Arian for his unlimited love and support.

Everything I am I owe it to them, the dearest people in the world, my parents. I can not express with words the gratitude I feel for you. Mami and Babi thank you for always believing in me...

"When the heavens were a little blue arch, stuck with stars, we thought the universe was too straight and close; I was almost stifled for want of air; but now it is enlarged in height and breadth, and a thousands vortices taken in. I begin to breathe with more freedom, and I think the universe to be incomparably more magnificent than it was before"

Bernard Le Bovier [Sieur de] Fontenelle (1657-1717)

To My Parents

Contents

[1] INTRODUCTION

1.1 Introduction

When the physicist Tim Berners – Lee invented the World Wide Web (WWW) probably he could not have foreseen that his application would change the Internet status from an academic and governmental network to a worldwide people's NET. The WWW lead to explosion of Internet's traffic such that in some parts of the communications networks it surmounts the telephony traffic. It offered possibilities for developing new applications, apart from applications such as e-mail, file transfer, etc. This was the beginning of Internet commercialization.

Today's Internet commercialization is still partially driven from new emerging applications, which are usually high demanding real time applications like audio and video streaming, IP telephony or multimedia conferencing. There are always users, either individuals or companies, who are willing to pay more for Internet access in order to be able to use these applications. This is especially desired by those companies that rely on Internet for managing their global enterprises. On the other hand there are a lot of users who are interested in getting the lowest price for access to Internet's traditional applications such as e-mail, web surfing, etc. User's (either individuals or companies) expectations from the Internet vary depending on the applications they are using and their costs. But, this variety of the Internet applications influences not only the expectations of users, but also of the content providers or software developers, which expect from Internet infrastructure to satisfy the diverse service needs of the applications and accordingly the end users. In such a highly competitive environment as the Internet Service Providers (ISPs) world, satisfying customer needs, regardless of whether they are other ISPs or end users is key to their survival. Therefore the ISPs zeal to provide value-added services to their customers comes straightforwardly.

The current Internet architecture, however, provides only simple services like IP addressing, routing, fragmentation and reassembly of IP datagrams and it relies on higher level transport protocols for sequential and assured data delivery and it provides no guarantees on the timely data delivery and throughput of traffic. These services provided by the current Internet are widely known as best-effort service. The best effort service is adequate for traditional Internet applications like e-mail, web browsing or file transfers, what certainly can not be said for the new emerging applications like IP telephony, multimedia conferencing or audio and video streaming, which require high bandwidth capacity and are sensitive to delay and delay variation. Consequently the need to equip the Internet infrastructure with mechanisms to enlarge the level of provided services, i.e. to provide the means for Quality of Service (QoS) on the Internet is natural.

The rapid growth of mobile systems, on the other hand indicates that the future Internet architecture will have to deal with an environment of mobile hosts, wireless networks, and

different access technologies as well. It is expected that mobile terminals will support the same variety of applications as fixed ones. In this environment QoS mechanisms are especially desirable, due to the scarce resources, variable error rate and unpredictable available bandwidth of the wireless link.

The efforts to enable end-to-end QoS over IP networks have led to the development of two different architectures, the Integrated Services architecture and more recently, the Differentiated Services architecture, which although different, support services that go beyond the best effort service.

However, these architectures do not take into account QoS mobility issues despite their importance, which may result in solutions that will not be able to cope with the future Internet services requirements.

Following the trend of end-to-end QoS deployment on the Internet, this thesis focuses on the interoperability of Integrated Services architecture and Differentiated service architecture and on QoS mobility issues. The objective is end-to-end QoS support in wired and wireless Internet.

1.2 QoS concept

Quality of Service (QoS) is definitely one of the most popular and challenging research topics in Internet computer networking today. It is an multidisciplinary topic involving several areas, starting from applications, different networking layers and network architectures but also network management and business models and finally the main target, the users - customers.

The definition of the concept of Quality of Service in itself is somewhat confusing, even though in the networking community it is assumed to be a common knowledge. Being such a multidisciplinary topic the QoS concept has a different meaning to different people depending on whether they are customers, service providers, hardware or software developers. Thus it is difficult to grasp within a single definition, the quality of service (QoS) concept suitable to its wide range of applicability. According to ITU (International Telecommunication Union) Recommendation E.800 Quality of Service is: "The collective effect of service performances which determine the degree of satisfaction of a user of the service". And so far this definition seems to be the most appropriate one, although it gives to QoS a bit of business oriented note.

The characteristics that qualify QoS are low delay and delay variation, loss and predictable consistent throughput capacity.

Ongoing research on QoS has proven that its deployment on the Internet introduces complexity in its overall functionality. Further, it affects the network management, the business patterns of networking companies and it also changes the customer behaviour on perception of the services it receives from the Internet. Therefore, finding an efficient solution for end-to-end QoS over Internet (i.e. IP networks) is a tough undertaking. It becomes even tougher when one is introducing QoS in an environment of mobile hosts and wireless networks.

1.3 QoS Applications and Users

Basically, all of the current Internet applications can be divided into two major groups: the real time applications and non-real time applications, depending on their performance related to QoS characteristics such as e.g. delay and delay variation. The real-time applications are those applications, which are time critical, while non-real applications do not have a time factor, and

accordingly there is no time value by which the data would be irrelevant. In [RFC1633] these applications are named as elastic. Real-time applications depending on whether they are tolerant or intolerant to variations of delay can be divided into intolerant and tolerant real-time applications [RFC1633]. Tolerant applications are those applications that can perform reasonably well in face of the nominal induced delay variation (jitter). Examples of such applications are various audio and video streaming applications. For intolerant applications the induced delay and delay variation would result in unrecoverable distortions. The Voice over IP (VoIP) and multimedia conferencing are typical intolerant real-time applications.

In this thesis intolerant real-time applications will be referred as to non-adaptive applications with strict QoS requirements, while tolerant real-time applications as adaptive real-time applications. And elastic applications are just traditional Internet applications, such as e-mail, newsgroups, etc.

Naturally, the applications QoS requirements are directly related to users, since not only that they will be paying for a better quality, but also they will be the ones assessing the quality they received. Needless to say but this assessment is subjective and differs for different users. Same as with applications, it is expected that users will adapt to QoS they receive and the levels of adaptability vary in sort of same way as for the application. Some users are willing to adapt, while some can not accept a lower quality than what was expected. User's adaptability and tolerance depends on the application and the QoS parameters. Studies focusing on user's perception of QoS discover factors that bridge the relationship between the subjective users QoS and QoS parameters [BoSa00]. These studies are very valuable in defining the application's QoS requirements, since than they are defined based on user's degree of satisfaction. For example, during a VoIP session users would tolerate the delay for the sake of "smoothness" of the voice, resulting in intolerance in jitter. In the afore mention studies related to users and interactive multimedia applications, it was found that the acceptable values of the delay and jitter should be less than 200ms.

1.4 What is this thesis about?

In general the need to enable QoS on the best-effort Internet is not disputable any more, although there is still debate going on whether the QoS mechanisms should be deployed only at the access networks leaving out the core network, or whether it should be deployed end-to-end. Moreover, following the trends in mobile systems, it is expected that the current best-effort Internet, in future will extend not only to QoS enabled Internet, but also to QoS enabled wireless Internet.

Therefore, in this thesis, end-to-end QoS on wired and wireless Internet is assumed to be a necessity.

The research efforts in enabling QoS over IP led to development of Integrated Services architecture which provides the guarantees, but due to per flow management of traffic introduces severe scalability in the core network element, i.e. router where the number of flows reaches up to millions. It has proven to be easily deployable only in access networks where the number of flows is rather moderate in terms of scalability issues.

Learning from the first experiences with Intserv researchers developed a new architecture, i.e. Differentiated Services architecture (Diffserv), which is intended to avoid the scalability problems and complexity of Integrated Services Architecture. Diffserv provides quality differentiation on

aggregates without strict guarantees on individual (micro-) flows, where QoS is attained by marking packets at the boundaries.

Even though it seems that the Differentiated Service architecture has lots of obvious advantages towards Integrated Services as being relatively simple and more scalable, Integrated Services has also advantages applicable to specific environments. Intserv provides end-to-end per-flow guarantees on the applications requirements and consequently achieves high utilisation of the network resources, while Diffserv is not intended to provide end-to-end per application guarantees, rather it provides service differentiation on aggregates. These Intserv characteristics are especially applicable to an environment where the network resources are scarce, the available bandwidth is difficult to predict and where the guaranteed service for flows and high utilisation is essential for the overall operation, i.e. at the environment of wireless networks.

The [BeYa00] proposes a framework for Integrated Services Operation over Diffserv networks and gives several specific realisations, in which end-to-end QoS can be supported. The proposed framework in [BeYa00] addresses in detail only one single scenario, where Intserv architecture is in the access and the Diffserv is in the core, while the rest of possible scenarios are addressed only vaguely. This leaves a lot of unanswered questions on one hand, but on the other hand a lot of freedom to work on specific design of Intserv over Diffserv architecture. As such, it represents a reasonable base for further work on this area and the framework definitely provides means for end-to-end QoS.

As already said, the IP QoS architectures do not take into account the QoS mobility issues. This may result in inefficient solutions when mobile hosts and wireless network are introduced. Furthermore the necessity for QoS support in this environment is even greater than in the wired Internet, because of specifics of the wireless network environment. For example, in terms of available bandwidth in a wired environment, some argue that there is no need for QoS, since the new technologies like Wavelength Division Multiplexing (WDM) will offer more than enough bandwidth for all real-time applications traffic. This kind of argument will not apply to wireless link, because of the limited capacity of the link and also unpredictability of the available bandwidth.

This thesis is a result of one year research work on the particular topic of bringing together the Integrated and Differentiated Services. It has resulted on a general architecture design with specific requirements and goals. Conforming to this architecture a boundary router for handling both types of architectures was designed. In order to prove the feasibility of the design, a basic prototype implementation of the boundary router specific mechanisms was developed. The purpose of having this kind of router is not only to be able to couple Intserv and Diffserv in wired Internet environment but also for wireless networks, i.e. a wireless access network and wired core network. By This combination is especially important for future Internet services, because by using Intserv in wireless access network the user gets per-flow guarantees for its application and the wireless network is highly utilised and Diffserv in the core offers scalability. Therefore this thesis addresses also the issues related to QoS and mobility and describes how the Integrated / Differentiated services architecture is to be applied in an environment of mobile hosts and wireless access technologies.

Based on the above, a statement of this thesis is derived: Solutions for enabling QoS over IP should take into account mobility issues also, in order to be able to fulfil these upcoming requirements of future Internet users.

1.5 Activities towards the Objectives

The work area of this project can be defined as QoS in the Internet, more specifically combination of QoS mechanisms for micro-flows and macro-flows, with possible extensions for mobility.

The main objective is developing an interoperable framework that will support the Integrated Services and Differentiated Services architecture and that will be applicable to both wired and wireless QoS Internet architecture. As such this thesis is a contribution to wired and wireless Internet QoS.

Activities towards reaching the thesis objective will cover studies, research, design, implementation and testing. The activities identified are:

- Literature Study on Internet QoS, Intserv, RSVP, Diffserv
- Continuous monitoring of Intserv/Diffserv integration proposals
- Designing a general Intserv/Diffserv architecture (without reference to network configuration and implementation environment)
- Designing in detail the border router architecture residing at the Diffserv network to support RSVP/Intserv and Diffserv interoperability
- Studying and creating the implementation environment (select existing Diffserv, Intserv, RSVP implementation, plan and install a network configuration (in conjunction with the existing testbed in Ericsson)
- Implementation of the border router prototype based on the detailed design. Possibly using the already existing RSVP/Intserv and Diffserv implementations
- Testing the current behaviour of the prototype implementation in a planned testing environment
- Studying QoS mobility issues and possibilities for applying the general Intserv/Diffserv architecture in an environment of mobile hosts and wireless access networks.

1.6 Outline of the thesis

In this thesis is organised in a top-down manner in line with the activities identified above. Firstly a description of the already existing Internet QoS architectures is given, continuing with a global design of the Intserv/Diffserv architectural framework. Conforming to the global design, the border router architecture for supporting interoperability between the two architectures is designed. To prove the feasibility of the design the prototype implementation of specific mechanisms is developed, the documentation of which is also part of the thesis. Finally it concludes with the results of the last activity defined above, the applicability of the Intserv/Diffserv architectural framework to mobile hosts and wireless networks environment.

The roadmap of the thesis is as follows:

Chapter 2 is an overview of existing QoS architectures, the Integrated Services architecture and the Differentiated services architecture, with the description of the services they provide. It also gives an overview of the Integrated Services operation over Differentiated Services networks

presented in [BeYa00] and related issues, such as the benefits of such interoperability, the proposed service mapping and possible specific realisations of the framework.

Chapter 3 gives a global design of the Intserv/Diffserv architectural framework, its design goals and requirements.

In Chapter 4 a detailed design of the boundary router architecture is given conforming to the design goals and requirements of the Intserv/Diffserv architectural framework presented in Chapter 3. Additionally, a detailed description of its components with emphasis on those components that are to be implemented from scratch is also given.

Chapter 5 is also result of design goals and requirements presented in Chapter 3. It proposes a mechanism on which RSVP might be carried transparently through the Differentiated Services network and a detailed description this mechanism, which is in conformance with the design goals and requirements given in Intserv/Diffserv architectural framework.

Chapter 6 gives an overview of Linux traffic control mechanisms and the Linux support for Intserv and Diffserv. Also the free software packages used for QoS support under Linux are described.

Chapter 7 describes the implementation design and the testing environment, i.e. the lab setup. It is actually an implementation documentation of the basic prototype implementation developed, of carried out tests and their results. Also the open issues and related future work are given.

Chapter 8 gives a description of the framework for QoS and mobility on the Internet proposed in [ReKa00], [KaRe00] and a possible application of the Intserv/Diffserv architectural framework to this framework.

Finally Chapter 9 discusses the conclusion and the contribution of this thesis and issues for future research.

[2] INTERNET QOS

2.1 Introduction

This chapter gives an overview of the current Internet architectures, i.e. the Integrated Services architecture and the Differentiated Services architecture, focusing on their functionality, characteristics and especially the services they define. The section 2.2 describes the Integrated Services (Intserv) architecture. The service models this architecture defines are given in the subsections 2.2.2 and 2.2.3 respectively. As part of the Integrated Services architecture section the Resource ReserVation Protocol (RSVP) is also described. The Differentiated Services (Diffserv) architecture is described in section 2.3, while its service models are described in sections 2.3.1 and 2.3.2 respectively. This chapter is concluded with a detailed overview of the framework of Integrated Services operation over Differentiated services proposed in [BeYa00] in section 2.4 and followed by an explanation of specific realisations and discussion of the benefits of this framework.

2.2 Integrated Services Architecture

The Integrated Services (Intserv) architecture described in detail in [RFC1633] recommends a set of extensions to the Internet architecture in order to enable services that go beyond the traditional best–effort service, aimed for addressing the real-time applications QoS requirements. QoS in terms of Intserv is associated with the time-of-delivery of packets and is and is characterized by parameters such as bandwidth, packet delay and packet loss rate [RFC1633]. The Intserv architectural design is based on the notion that in order to fulfill the QoS requirements of the applications, network resources should be managed and controlled, which implies that the admission control and resource reservation are the key building block of this architecture. As such the Intserv architecture provides mechanisms by means of which applications can choose between different services for their traffic and explicitly signal QoS requirements per individual flow to network elements (hosts, routers or subnets). The network elements depending on the available resources implement the required services, based on which QoS will be delivered to conforming traffic types in the data transmission path.

The functionality of the Integrated Service architecture can be seen as a composition of two basic elements, Integrated Service model and the reference implementation model, which provides the necessary kit and the accompanied terms for realization of the Integrated Service model. Each of these elements encompass a certain number of functional entities, which are described below:

- ***Integrated Service model***

The Integrated Service model defines two types of services the Controlled Load Service and the Guaranteed Service for usage by the real-time applications. The specific service is invoked by the applications QoS requirements. The application's generated traffic, depending on these QoS requirements, will get the one of two existing service treatment, i.e. either the Controlled Load Service or Guaranteed service. QoS requirements depend on the nature of different applications, that is, whether they are elastic, non-adaptive or adaptive real-time applications. (See Section 1.3).

Further, the Integrated Service model consists of a set of service commitments, related to the service requests. The network commits to deliver service either to individual flows or to collective-aggregate flows. To individual flow the commitments relate to the quality of service delivered to this flow. A service commitment to aggregated flows relates to "aggregate resources made available to the various entities" [RFC1633], i.e. several hosts or domains. This commitment is done based the link sharing model. The link-sharing model is based on sharing the available link bandwidth among the flows belonging to the aggregate flow according to some specified shares given for example by the network administrator, such as sharing the link bandwidth between a number of protocols, or a number of services, etc. (see RFC[1633]).

In order to avoid the danger of failures in providing the agreed service, IS model provides several scenarios where the traffic control is provided implicitly by dropping the packets which are marked as pre-emptible, i.e. less valuable packets within a flow.

The provisioning on usage feedback is an important part of any QoS architecture, since it is related to accounting and billing. Despite its importance the Intserv only mentions the provisioning on usage feedback without giving a lot of details.

Reservation model in Intserv describes scenarios on how the reservations are made and managed. The simplest reservation model is the one where an application request for a particular QoS is either accepted or rejected by the network is the simplest.

- ***Implementation reference model***

For realization of the Integrated Services model the Implementation Reference model defines several mechanisms that encompass the layer 3 (router) scheduling, classification, admission control and resource reservation.

The classification, scheduling and admission control are part of traffic control tools. The classifier determines to which class each packet belongs according to their QoS requirements, i.e. the service that determines the way the scheduler should handle them. The scheduler processes these packets based on their QoS requirements. Each network element in the network performs admission control and policy control to the incoming flows in order to determine whether there are enough resources and whether the flow has permissions to request the specific service.

The RSVP signaling protocol [RFC2205], [RFC2210] was designed as a dynamic mechanism for explicit reservation of resources in Intserv, although Intserv can use other mechanisms as well. The Intserv architecture and RSVP can also function independently of one another. The RSVP signaling protocol is described in section 2.2.1.

And, even though Intserv was designed and provides the means for end-to-end QoS, it is not widely deployed. As it is emphasised so many times by now, due to maintenance and control of per-flow states and classification, reserving resources per-flow introduces severe scalability problems at the core networks, where the number of processed flows is in a millions range. Consequently the usage of the Integrated Services architecture is limited to small access networks where the number of flows using reservations is modest.

The simplified RSVP/Intserv framework is shown in Figure 2.2-1. As it is shown every RSVP aware router in the Intserv will perform RSVP signalling, admission control, scheduling and policing.

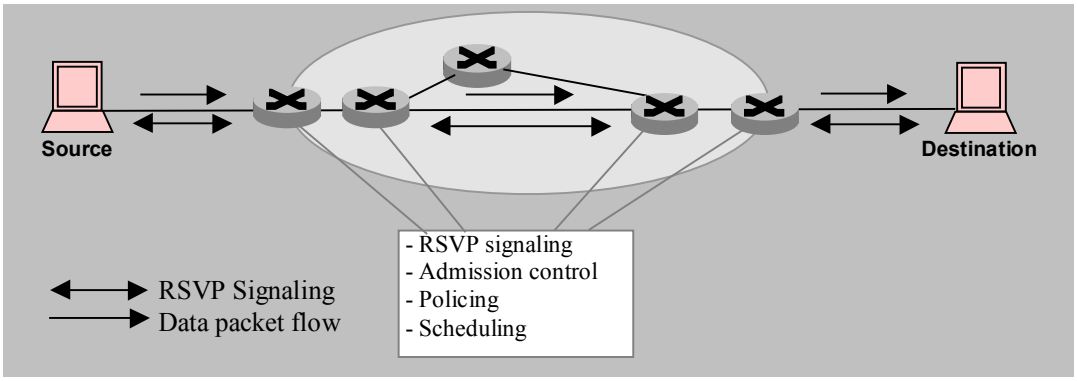


Figure 2.2-1 RSVP/Intserv framework

2.2.1 RSVP

The Resource Reservation Protocol (RSVP) [RFC2205], [DuYa99] is a signaling protocol that can be used by an application to convey its QoS requirements to network elements. RSVP is used only for communication of QoS parameters and it doesn't provide any QoS related functions, that is the RSVP protocol itself has no understanding of the information it carries on QoS requests. RSVP is initiated by an application at the beginning of a communication session. A communication session is identified by the combination of the IP destination address, transport layer protocol type and the destination port number. Each RSVP packet contains details of the session they belong. The resource provisioning is independent of RSVP; that is the admission/rejection of the required resources by means of RSVP for a particular flow is a function of Intserv in this case. Once the resources requested by RSVP are reserved, they will be used by the particular data flow.

RSVP protocol defines seven types of messages, of which the fundamental ones are the PATH and RESV messages. PATH and RESV messages carry out the basic operation of RSVP. The rest of the RSVP messages are used to either provide information about the QoS state or to explicitly delete the QoS states along the communication session path. The RSVP messages and their functions as given in [KaRe00] are listed in Table 2.2.1-1. All RSVP messages are sent through the network as raw IP datagrams with the protocol number 46, and PATH, PATH Tear and RESV Conf should be sent with the router alert option set [RFC2113].

RSVP Messages	
RSVP Message Name	**RSVP Message Function**

PATH	The PATH message is sent by a source that initiates the communication session and it explicitly binds the data path of a flow. Furthermore, it describes the capabilities of the source.
RESV	The RESV message is issued by the receiver of the communication session and it follows exactly the path that the RSVP PATH message has followed hop by hop back to the communication session source. The RESV message on its way back to the source may install QoS states at each hop. These states are associated with the specific QoS resource requirements of the destination. The RSVP reservation states are temporary states, i.e., soft states, that have to be updated regularly
PATH Error	Is used to report errors that are occurring during the installation of a path from the source to the destination of a communication session.
RESV Error	Is used to report errors that are occurring during the installation of the reservation states along the communication session path.
RESV Confirm	It provides a positive indication to the initiator of the communication session informing that all nodes along the communication session path accepted the reservation request. When a receiver originates a reservation request, it can also request a confirmation message to indicate that its request was (probably) installed in the network. Thus, the RSVP Confirmation messages are typically sent by the source of the communication session directly to the destination of this communication session. Intermediate nodes do not process RSVP confirmation messages.
PATH Tear	Is sent by the source of the communication session and it explicitly deletes the stored QoS state information on all nodes included in a communication session path.
RESV Tear	Is sent by the destination of the communication session and it explicitly deletes the stored QoS state information on all nodes included in a communication session path.

Table 2.2.1-1 The RSVP messages

The PATH message is sent by a source that initiates the communication session and it explicitly binds the data path of a traffic stream. The PATH message contains the Sender TSpec, which is used by the source to specify the traffic characteristics of its data flow. All the network elements on the way to destination which PATH message traverses and the destination itself will install the PATH state and will use the traffic characteristics of this particular flow from the TSpec. ADSPEC is another object carried in the PATH message, which is used by network elements to describe the QoS service –specific parameters as well as some default parameters describing the data path supported by the network. Unlike the TSpec object, the ADSPEC is optional and is updated in every network element, i.e. hop-by-hop.

The RESV message is issued by the receiver of the communication session and it follows the same path as the RSVP PATH message, hop by hop back to the communication session source. The RESV message on its way back to the source may install QoS states at each hop associated with the specific QoS resource requirements of the destination. Based on this state the network elements along the path will allocate resources for flows, and police and shape the traffic accordingly. The RESV message contains besides the information about the reservation style,

two objects: Flow Spec and Filter Spec. This set of objects is known as Flow Descriptor. The Flow Spec defines the requirements for the data flow, that is the type of the service requested (CL or GS), the service parameters for invoking QoS (RSpec) and the parameters of the flow requesting the service (TSpec). RSpec is only present in the flow descriptor if the requested service is guaranteed service. The Filter Spec is used to set adequate parameters in the packet-classifier process. The RSVP reservation states are temporary states, i.e., soft states, that have to be updated regularly. This means that PATH and RESV messages will have to be periodically retransmitted. If these states are not refreshed then they will be removed.

2.2.2 Controlled Load Service

Controlled Load (CL) Service [RFC2211] is intended for adaptive real-time applications which are highly sensitive to overloaded conditions in the network. The controlled load service offers only a single function to these applications, it provides the traffic delivery within the same bounds as would have been the case in the environment of "unloaded" (not heavily loaded or congested [RFC2211]) networks. CL does not accept nor use the specific QoS parameters such as packet loss and delay as control parameters. In requesting CL service applications may expect, under the assumption that the network is functioning correctly, that their traffic will be delivered successfully and that the transit delay induced by the network is close to the minimum transit delay of successfully transmitted traffic. The QoS disruption in the delivery service depends on the "burst time". Burst time is defined as the time needed for the transmission of the maximum flow's burst size at the required transmission rate. These parameters are defined in the TSpec of the requester's flow. The short duration of QoS disruption events occur when the average queuing delay is significantly larger than the burst time and they are considered as "normal operation". In a word, if the application's traffic fall outside the parameters of the TSpec, the QoS provided to "exhibit characteristics indicative of overload, including large numbers of delayed or dropped packets" [RFC2211]. But if the congestion loss is significantly larger than the burst time, that is considered as a failure of the resource allocation schemes.

The concrete Controlled Load TSpec parameters as given in [RFC2211] are:

1. r – token bucket rate (measured in bytes/second)
2. b – token bucket size (measured in bytes)
3. p – peak rate (measured in bytes/second)
4. M – maximum datagram size (measured in bytes)
5. M - minimum policed unit (measured in bytes)

The network elements receiving a CL request must provide the necessary bandwidth and packet processing resources for handling the requested level of traffic as given in the TSpec of the requestor. The method a network element uses to determine whether the request can be accommodated is a local matter, and can be implementation dependent as long as the control parameters and message formats are interoperable. It may employ measurement-based approaches or it may employ appropriate scheduling mechanisms.

Independently of which mechanism it uses, the network element provides the CL service only to the traffic conforming to the TSpec given at the flow setup. If this traffic is accompanied by non-conformant traffic, the network element must ensure first to fulfil its service commitments, then it must ensure that the non-conformant traffic doesn't impact other conforming traffic. The network element must attempt to send this traffic as best – effort. In this case there might also be

a problem, since this non-conforming traffic may now impact unfairly the best-effort traffic up to the point of having packets discarded. There are mechanisms, which handle such problems, but those are outside the scope of this thesis.

2.2.3 Guaranteed Service

The Guaranteed Service (GS) [RFC 2212] is an quantitative service which provides bandwidth guarantees and delay bounds and as such it is intended for non-adaptive real time applications with strict QoS requirements. The GS service controls only the maximum delay; thus it does not control the minimum delay or control or minimise the jitter. The delay consists of the fixed delay and the queuing delay. Fixed delay is a path property and is determined by the setup mechanism (e.g. RSVP) during the path set-up, while queuing delay is determined by the GS service. In order to determine specific end-to-end delay bounds, GS service relies on the behaviour of each network element in the path starting from the source. The end-to-end delay bound as given in [RFC2212] is:

$$D = \begin{cases} \dfrac{(b-M)(p-R)}{R(p-R)} + \dfrac{(M+Ctot)}{R} + Dtot & p > R \geq r \\ \dfrac{(M+Ctot)}{R} + Dtot & r \leq p \leq R \end{cases}$$

Where:

r – token bucket rate (measured in bytes/second)

b – token bucket size (measured in bytes)

p – peak rate (measured in bytes/second)

M – maximum datagram size (measured in bytes)

R – flow service rate (or bandwidth) (measured in bytes/sec)

Ctot- end-to-end calculation of C (see below)

Dtot – end-to-end calculation of D (see below)

The source and network elements behaviour differs and is described by means of two models: the token bucket model and the fluid buffer model depicted in Figure 2.2.3-1.

By means of the token bucket model, in particular the token bucket size b and the rate r, the applications (which in fact controls these parameters) has an a priori knowledge about the queuing delay that the guaranteed service will provide. The application's source is allowed to transmit data as long as there are tokens available in the bucket. The packets are released at the token bucket rate r, although it may happen that the source produces more packets than this rate r. In this case the packets are stored in the buffer and than released at rate r. The transmission rate is limited by the peak rate p. When there is no transmission the bucket can accumulate tokens up to size b. The bucket is filled at a constant rate with tokens, until it is full. In case the delay exceeds the expectations the application can modify its token bucket and data rate to reduce the delay.

The fluid buffer model is the model by means of which the network elements will allocate the resource for guaranteed service flows, packets. The flow's service level is described by the buffer size B (measured in bytes) that the flow can use and the flow servicing rate R, which is the flow's

share of bandwidth. The buffer is necessary in cases when the transmission rate p exceeds the flow service rate R.

The guaranteed service relies on the fluid model conforming to the token bucket model to ensure that the queuing delay of any packet in the flow is less than the total delay computed along the flow path:

$$D_t = \frac{b}{R} + \frac{C}{R} + D$$

b—the maximum number of tokens,

R—the service rate the packets are served,

C—additional delay which is rate -dependent, referred also as packet serialisation (unit is in bytes)

D—additional delay which is not-rate – dependent, is a result of time spent waiting for transmission through a node (unit is in microseconds)

The cumulative end-to-end calculation of C and D terms, i.e. Ctot and Dtot defines the flow's deviation from the fluid model. These parameters are built up along the path during flow set-up. Together with the Csum and Dsum, which present the sum of the rate-dependent delay and non-rate –dependent delay respectively, since the last reshaping point in the network, these parameters are necessary for calculating the local buffer requirements and are part of the ADDSPEC object.

Guaranteed service is invoked by specifying the traffic (TSpec) and the desired service (RSpec), thus by the (TSpec, RSpec) to the network elements. The TSpec is represented by the rate (r), the bucket size (b), the peak rate (p), the minimum-policed unit (m) and the maximum datagram size (M). Guaranteed Service uses also the TOKEN_BUCKET_TSPEC as a parameter of TSpec. The RSpec consists of the rate R and a slack term S, where R must be $R \geq r$ and S must be nonnegative. S is used by intermediate network elements to reduce its resource reservation for a particular flow. If at the time of service invocation no slack is specified, than the slack term S is set to zero.

Each network element receives a service request of the form (TSpec, RSpec), where the RSpec is of the form (Rin, Sin). It processes this request and it either accepts the request and returns a new RSpec of the form (Rout, Sout) or it rejects it. The processing rules for generating the new RSpec should satisfy the inequality given below:

$$S_{out} + \frac{b + C_{toti}}{R_{out}} \leq S_{in} + \frac{b + C_{toti}}{R_{in}}$$

where Ctoti is the cumulative sum of the error terms, C, for all the network elements that are upstream of and including the current element, i. In other words, this element consumes (S_{in} - S_{out}) of slack and can use it to reduce its reservation level, provided that the above inequality is satisfied. R_{in} and R_{out} must also satisfy the constraint: $r \leq R_{out} \leq R_{in}$.

The flow with the new RSpec is forwarded upstream to the source.

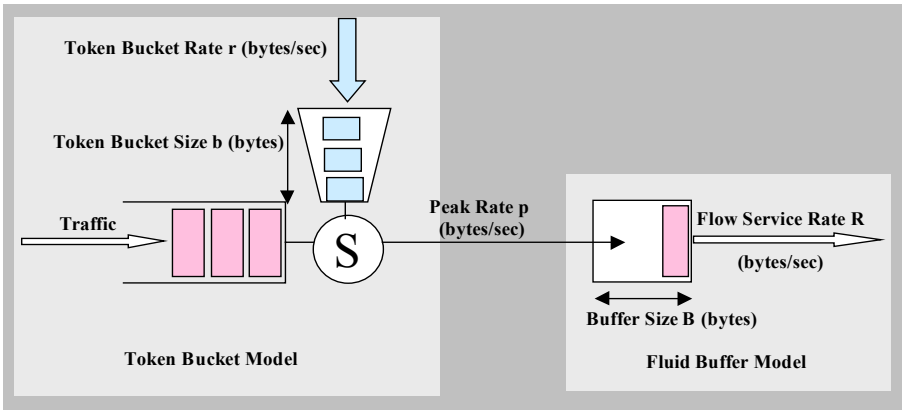


Figure 2.2.3-1 Guranteed Service model

2.3 Differentiated Services Architecture

The Differentiated Services (Diffserv) architecture [RFC2475], [RFC2638], [BeBi99] was introduced as a result of the efforts to avoid the scalability and complexity problems of Intserv. Scalability is achieved by offering services on aggregate basis rather than per-flow and by forcing as much as possible the per-flow states to the edges of the network. The service differentiation is achieved by means of Differentiated Service (DS) field in the IP header and the Per-Hop Behaviour (PHB) as main building blocks. At each node packets are handled according to the PHB invoked by the DS byte in the packet header. The Diffserv divides the entire network into domains, where Diffserv domain as defined in [RFC2475] is a contiguous set of nodes which operate with a common set of service provisioning policies and PHB definitions. The Diffserv domain consists of the interior nodes and boundary nodes, which connect the Diffserv domain to other domains and are responsible for conditioning the traffic according to the service agreement that is in effect between neighbouring boundary domains. The Diffserv domain will provide to its customer, which is a host or another domain, the required service by complying fully with the agreed Service Level Agreement (SLA). A SLA is a bilateral agreement between the boundary domains negotiated either statically or dynamically. The transit service to be provided with accompanying parameters like transmit capacity, burst size and peak rate, is specified in the technical part of the SLA, i.e. the Service Level Specification (SLS). The simplified Diffserv framework is given in Figure 2.3-1.

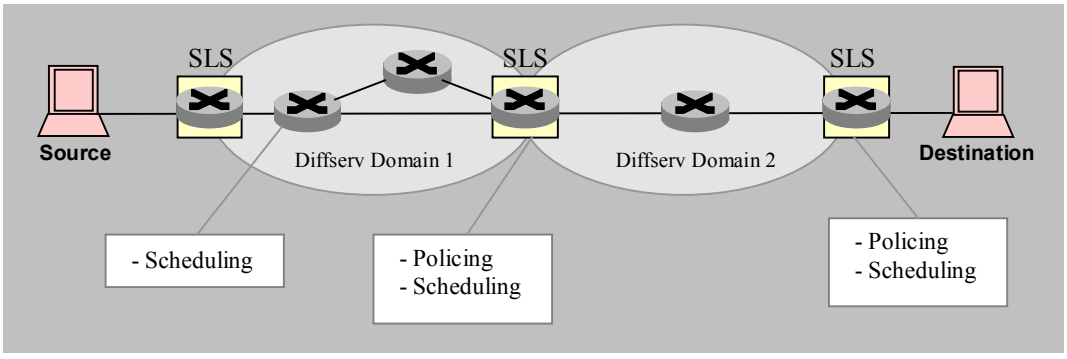


Figure 2.3-1 Differentiated Services Framework

The specific realization of the Diffserv architecture depends on whether the resources are statically or dynamically provisioned and on resource allocation mechanisms. The two-tier resource management model for Diffserv proposes an approach allowing each individual

Diffserv domain to make their own choices on the mechanisms and protocols to be used for internal QoS support, while for the external QoS support they rely on the SLA. By means of this model the end-to-end QoS can be achieved through a concatenation of inter-domain and intra-domain resource allocations, as long as those allocations match the level of the aggregated demand [RFC2638]. Each domain in this model will have a resource manager, i.e. the Bandwidth Broker to take care of the intra and inter domain resource management and traffic control. The two-tier model is depicted in Figure 2.3-2

The Diffserv architecture is certainly promising, but there are a lot of open issues related to the dynamic resource provisioning that need to be defined and researched.

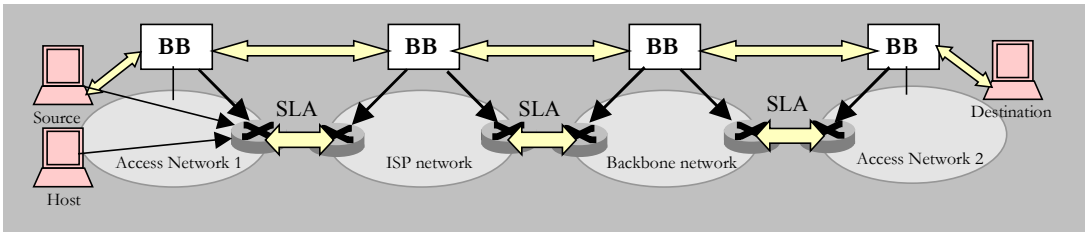


Figure 2.3-2 Two-tier resource Management Model for Differentiated Services Networks

2.3.1 The DS field and PHBs

The Differentiated Service architecture provides service differentiation by means of the Differentiated Service (DS) field in the IP header and the Per-Hop Behaviour (PHB), which defines the externally observable behaviour at the node.

Initially, the Diffserv Architecture had renamed the TOS octet of the IPv4 header and the Traffic Class octet of the Ipv6 respectively as DS field [RFC2474]. This DS field consists of a 6-bit DS codepoint (DSCP) and two "currently unused bits". However the [Gro00] argues rightly that this leads to ambiguities and inconsistencies, especially since the CU bits have not been assigned to Diffserv, but are assigned for experimental use for an explicit congestion notification scheme [RFC2481].

Therefore in [Gro00] a new definition of DS field has been proposed: DS field is the six most significant bits of the IPv4 TOS octet or the IPv6 traffic class octet. And the DSCP is a value encoded in the DS field, which should be used by each node to select the PHB for forwarding the packets.

The DS fields, i.e. the DSCP bits of the IP packet header are set at the network boundaries, i.e. hosts, internal administrative boundaries or inter domain boundaries. Marking of these bits indicates to the internal nodes (core routers) the treatment type in the packet forwarding path. In the packet forwarding path services are realised by mapping the DCSP to a per – hop behaviour (PHB), at each node along the path. The codepoints may be chosen from a set of mandatory values defined as Class Selector codepoint and they must have some PHB associated with it. There are 8 Class selector codepoints one of which is associated with the default PHB, used also in case there is no known mapping. The division of the codepoint space is given in [RFC2474].

After the mapping of the codepoints to the appropriate PHB, based on PHB the network node has enough knowledge on how to allocate the resources to the behaviour aggregates. PHBs can be defined in term of their relative traffic properties (delay, loss), their resources (buffer and bandwidth) or in terms of their priority to other PHBs and since they are defined in terms of

behaviour aggregates they are not dependent on implementation. As such they can be implemented by using buffer management or packet scheduling mechanisms. Multiple PHB can be bundled into PHB groups.

Two PHB groups have been defined, the assured forwarding AF PHB [RFC2597] and the expedited forwarding EF-PHB [RFC2598].

2.3.2 Assured Forwarding PHB Group

The Assured Forwarding (AF) PHB group is used by a Diffserv domain to provide the assured service to the customers that require reliable services even during congestion in the networks. As stated in [XiLi99], SLAs for assured forwarding are usually static, meaning that customers can start transmitting data whenever they want without signalling their providers. This is valid only in case of static provisioning of Diffserv, while for dynamic provisioning SLAs for assured forwarding should be dynamic. Each customer will get a share of bandwidth, which he can spread among his applications as preferred. Assured forwarding (AF) PHB group provides the means to offer different levels of forwarding assurances to IP packets received from a customers domain and possibly different delay and jitter, although the last two are not the primary focus of AF PHB [Chi98]. There are 4 AF classes defined with three levels of drop precedence within each class. The recommended codepoints for these four classes are given in the Table 2.3.2-1, where the first 3 bits represent the AF class number and the last three bits represent the drop precedence.

Drop Precedence Levels	AF Class 1	AF Class 2	AF Class 3	AF Class 4
Low	001010	010010	011010	100010
Medium	001100	010100	011100	100100
High	001110	010110	011110	100110

Table 2.3.2-1 AF Codepoint Values

The level of forwarding assurance of an IP packet depends on the amount of resource assigned to the AF Class, the current load on the class and in case of congestion the drop precedence of packet. The ordering of the codepoints refers to the ordering of the delivered service, the higher level service has the lowest dropping probability.

As described in [RFC2597] AF PHB can be used for instance for "olympic service" implementation, which consists of: "bronze", "silver" and "gold". The packets belonging to "gold" class experience lighter load than the packets mapped to the "silver" class. The 'bronze", "silver" and "gold" service would be mapped to the AF1, AF2 and AF3, with the drop precedence levels mapped to 1, 2, 3.

The AF specification indicates that it is not mandatory to implement all 4 AF classes. Thus in case the Diffserv operator expects that congestion in its domain will occur rarely, an implementation supporting two levels of drop ratio is satisfactory. Further, Diffserv node must allocate a configurable minimum amount of buffer space to each AF class, packets belonging to different AF classes should be forwarded independently of one another, and also a Diffserv node must not reorder the AF packets belonging to the same microflow, regardless of their drop precedence.

2.3.3 Expedited Forwarding PHB

The Expedited Forwarding PHB provides tools to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through Diffserv domains [RFC2598]. This service is also described as premium service in [RFC2638] and it appears to be like a virtual leased line type of service. The SLA (i.e. SLS) specifies a peak bit rate, which customer's applications will receive and it is the customer responsibility not to exceed this rate since than their packets can be dropped. Thus the EF definition assumes that the traffic is a conditioned at the Diffserv borders such that the core routers can forward the packets immediately without the risk of exceeding the negotiated bit rate.

The definition of the EF PHBs can be deduced from the requirements imposed on Diffserv node mechanisms for handling the EF traffic stream. Diffserv node in the forwarding path ensures that the particular aggregate has a minimum departure rate independently of the other traffic at the node. Thus the traffic should be conditioned such that the maximum arrival rate at any node is less than the minimum departure rate to provide a type of EF PHB. EF-PHB implies a strict bit rate control or a fast forwarding treatment to the Diffserv boundary nodes and Diffserv internal nodes respectively.

The recommended codepoint for the EF PHB is 101100. A Diffserv boundary node can remark the EF PHB packets with a new DSCP only if the new DSCP would satisfy the EF PHB requirements.

EF PHB is implemented by different mechanisms, such as simple priority queuing (PQ), weighted round robin scheduling (WRR), class based queuing (CBQ), etc. All of these mechanisms implement the basic EF PHB properties, but different implementation choices result in different properties of the same service, such as for instance jitter as seen by individual microflows.

2.4 Intserv over Diffserv

As described above both Integrated and Differentiated Services architecture are designed to deploy QoS on the best effort Internet, by means of different mechanisms for differentiation of services and each having their own advantages and disadvantages. The framework for Integrated Services operation over Differentiated Services [BeYa00] views the two architectures as complementary towards deploying end-to-end QoS. As noted in this framework Intserv provides means for end-to-end QoS over different heterogeneous networks and it must be supported in different network elements, thus Diffserv network is just a network element in this end-to-end path. It is primarily intended to support the quantitative (guaranteed) services end-to-end, which has not been deployed yet by RSVP/Intserv, due to the lack of scalability.

The benefits of this framework for Intserv is thus rather obvious, since Diffserv aggregate traffic control scalability fills in the lack of scalability of the RSVP /Intserv. On the other hand Diffserv itself will profit by using RSVP as a mechanism to properly provision quantitative services across the networks:

- In Diffserv, "admission control is applied in a relatively static way by provisioning policing parameters at network elements" [BeYa00]. Using RSVP will enable Diffserv to apply resource-based admission control, which will optimise the use of resources in the network. Furthermore, it will enable Diffserv to apply policy-based admission control on users/applications traffic.

- In Diffserv the DSCP codepoint can be set either at the host or at the router and by means of an explicit mechanism such as RSVP DCLASS [Ber99], Diffserv will be able to perform traffic identification/classification straightforwardly.
- Intserv network elements perform per-flow traffic conditioning. Pre-conditioning traffic in this way before they enter Diffserv, "enhances the ability of Diffserv to provide quantitative services using aggregate traffic control" [BeYa00]

Thus, the [BeYa00] claims rightly that Intserv and Diffserv can be used as complementary technologies, e.g. using Intserv at the access networks it enables the hosts to request and reserve resources per flow by means of RSVP, and Diffserv in the core will avoid the RSVP scalability problems.

The reference network for the proposed Intserv/RSVP over Diffserv framework [BeYa00] is shown in Figure 2.4-1.

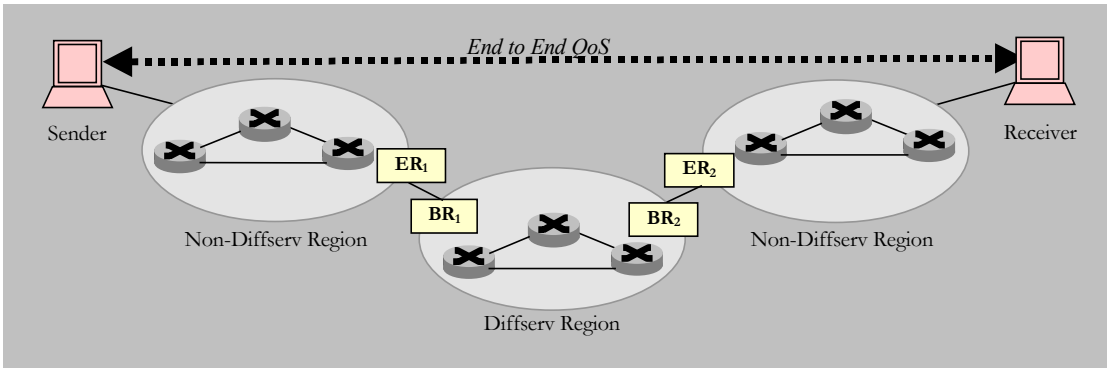


Figure 2.4-1 The reference network for the Intserv/RSVP over Diffserv framework

As shown above the framework consists of the following elements:

- Sender and Receiver are RSVP aware hosts, which can initiate an RSVP process based on the requirements of QoS-aware applications running on the hosts.
- Edge and Border routers are the edge devices whose functionality depends on the specific realisation of the framework.
- Non-Diffserv region is an Intserv capable region containing hosts and other network elements that are Intserv capable and also other network elements.
- Diffserv network region may or may not contain RSVP-aware routers.

2.4.1 Specific Realization of the framework

The [BeYa00] draft proposes several specific realisations of RSVP/Intserv and Diffserv interoperability vaguely, which are leaving a lot of unanswered questions, focusing only on a particular scenario out of several possible scenarios. The specific realisation of the RSVP/Intserv - Diffserv interoperation, depends on Diffserv resource management and on Diffserv network region RSVP awareness. As already mentioned in Section 2.3 resource management in Diffserv can either be static (managed by human agents) or dynamic (via protocols). The Diffserv network region is considered to be RSVP aware if it has standard RSVP routers within its domain. In one

extreme case this router is situated only at the borders of the domain, and at other extreme all the routers within Diffserv are RSVP aware. Which of the routers will be RSVP aware routers depends on the network administrator. In case of statically provisioned resources the owner of the Diffserv region has negotiated a static SLA with its customer, i.e. the Intserv region, so the owner of the Diffserv region will perform a number of actions conforming to the negotiated SLA to comply to service commitments. In case of dynamically provisioned resources, the SLAs are negotiated in a dynamic manner upon aggregated requests from the customers thus, it requires far more sophisticated mechanisms within the Diffserv in order to be able to do the right admission control and resource allocation. These mechanisms are not yet defined and it's an on going research issue. The most favourable is the use of the Bandwidth Brokers (BB) that will perform admission control and resource allocation within each Diffserv domain. The intra-domain communication protocol has not been decided yet. The RSVP, COPS, SNMP, etc are the ones that are looked at for this purpose, while for the inter-domain communication between BB, RSVP aggregation (see Section 5.2.2) and other protocols [TeCh99] are looked at.

The possible scenarios of RSVP/Intserv – Diffserv interoperability as described in [Kil99] are given in Figure 2.4-2. As it is depicted from all the possible scenarios only scenarios 1 and 3 are the ones that make sense and are beneficial for usage, while the rest of the scenarios will have the same scalability problems as on Intserv architecture. Scenario 4, where Intserv is used as a transmission medium for Diffserv traffic can be seen only in case Intserv region is an access network, that Diffserv traffic needs to traverse on the way to its destination.

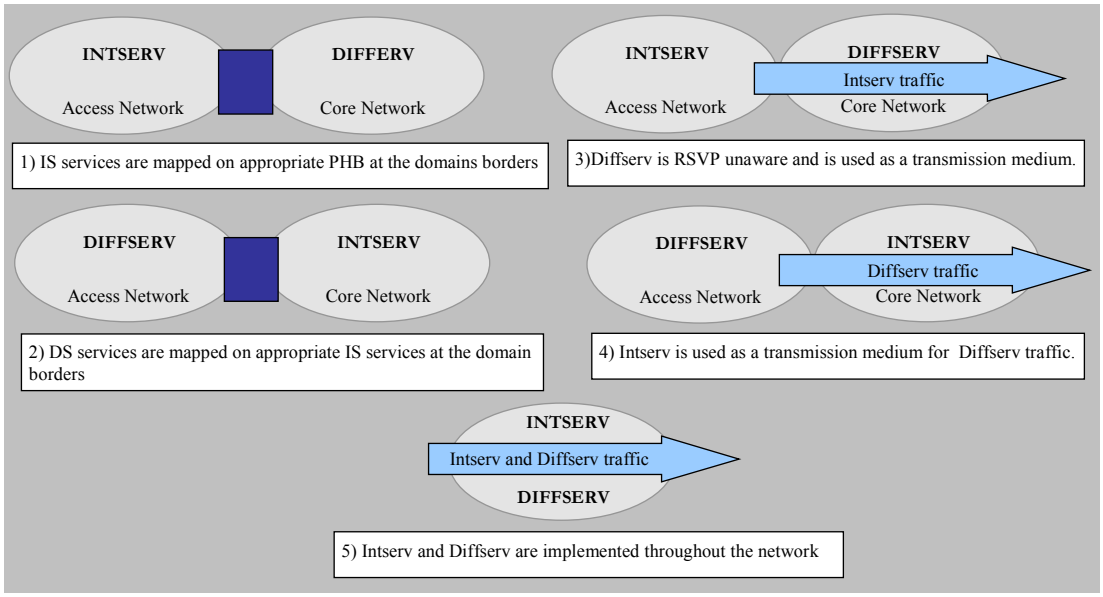


Figure 2.4-2 Possible RSVP/Intserv interoperability scenarios

2.4.2 Service Mapping

Independently of the Diffserv resource management, the service mapping of Intserv- defined services to Diffserv-defined services is essential for Intserv over Diffserv operation, unless the Diffserv is used only as transmission medium. Service mapping depends on appropriate selection of PHB, admission control and policy control on the Intserv request based on the available resources and policies in the Diffserv. There is a known default mapping defined as it is shown in the Table 2.4.2-1. So, the Guaranteed Service is to be mapped to EF PHB, while the Controlled

load depending on whether it is latency tolerant or not can be mapped to AF PHB different priority levels.

Intserv	Qualifier	Diffserv PHB
Guaranteed Service	--	EF
Controlled Load	L	AF (higher priority)
	H	AF (lower priority)

Table 2.4.2-1Default Mapping

The routers that are able to handle both RSVP and Diffserv packets will perform the service mapping above and, those routers can always override it. These routers will usually be situated at the edges of the Diffserv region. If however, there is a marking at the host or routers residing outside the Diffserv region, then the new marking values should be communicated to the marking device by using a mechanism such as RSVP DCLASS object [Ber99].

[3] INTSERV / DIFFSERV ARCHITECTURAL FRAMEWORK

3.1 Introduction

As already mentioned in Chapter 2, the [BeYa00] draft on Intserv operation over Diffserv provides a good basis for end-to-end QoS deployment on the Internet. Intserv is suitable to provide fine-grained QoS on a per user or per application basis, i.e., on a per micro-flow basis. Diffserv does provide a more course grained QoS, i.e., it deals with macro-flows. The interoperation would be even more valuable if it is to be applied in a wireless environment, than to a traditional wired environment. One can envisage a QoS-aware wireless IP network, where both mechanisms are used, that is, RSVP/Intserv in the access network, what is especially valuable in the wireless environment where bandwidth is a scarce resource and Diffserv in the core network. In this way the advantages of both architectures would be brought together.

However, [BeYa00] draft leaves out a lot of open issues related to a development of a single Intserv/Diffserv architecture. Instead it proposes several scenarios of specific realisation mainly due to the ongoing research on the Diffserv architecture for defining optimal mechanisms for resource management. The application of this interoperability framework in a wireless environment is not mentioned at all.

Having this in mind in this chapter an architectural framework for Intserv over Diffserv is described, such that it is independent from the specific resource management in the Diffserv, with the possibility to easily extend it to specific realisations when the Diffserv resource management mechanisms are defined. It is defined for traditional wired networks, with the possibility to apply it in a wireless network environment. This applicability is described later in Chapter 8.

This chapter is organised such that the design goal and objectives are defined first, followed by the requirements ending with the global design of Intserv/Diffserv architectural framework.

3.2 Design Goal and Objectives

The main design goal for Intserv/Diffserv architectural framework is of course the end-to-end QoS deployment in the current Internet architecture. The usage of RSVP/Intserv architecture is restricted only to the access networks, while the Diffserv architecture is to be deployed in the core network. Whether the core network will be pushed all the way to edges or not depends on the network administrator of the Diffserv. The requirements regarding these two architectures

are in full compliance with the ones defined in [RFC1633], [RFC2210], [RFC2475], [RFC2638] and they will not be mentioned here.

The design of Intserv/Diffserv architectural framework is initially motivated by the following objectives:

- *Introducing as little as possible new changes to the already existing architectures*

  Logically for the Intserv/Diffserv interoperation there will be new mechanisms introduced in network architectures, but these mechanisms will not change the actual Intserv or Diffserv architectural design. The necessary mechanisms will be introduced, as entirely new or already existing ones will be enhanced to handle the Intserv/Diffserv interoperability.

- *Independent of specific realisation of the different Diffserv network administrative domains*

  The Intserv architecture is already well defined and there are no intentions in the research world to change. Furthermore, it is a well-known concept and it is clear what sort of mechanisms will be deployed in an Intserv administrative domain.

  The Diffserv architecture on the other hand as a more recent QoS architecture, it is not a clearly defined concept in a sense that a Diffserv network administrator for its intra-domain traffic engineering or resource provisioning, etc can use different mechanisms from another Diffserv domain.

  This objective is related to these specific realisations and the intention is that the Intserv/Diffserv architectural design should be independent of these specific realisations

- *Intserv/Diffserv interoperability occurs at the boundaries between Intserv and Diffserv domains.*

  The main functionality for the Intserv/Diffserv interoperability will be performed at the edge devices either at Intserv or Diffserv. These devices will have the burden handling both RSVP/ Intserv messages and Diffserv packets and perform the necessary functions related to their interoperation.

- *Easily extensible to dynamic resource management mechanisms – dynamic SLA, SLS respectively*

  Initially, Intserv/Diffserv architectural framework is based on static provisioning of resources in Diffserv network domains, mainly due to a lot of open question related to mechanisms for dynamic resource provisioning. But, this architectural framework design should also be able to support dynamic resource management mechanisms, once they are settled for Diffserv, by performing only minor changes to the design.

- *Easily extendible for usage in the wireless network environment.*

  Intserv/Diffserv interoperability architectural framework is designed for traditional wired networks, with the objective to deploy it in a wireless environment where its applicability would be even more valuable.

3.3 Requirements

Based on the above given design objectives there are certain requirements defined that are necessary to be fulfilled by the functional entities of the Intserv/Diffserv architectural framework for its overall functionality:

- QoS aware End-Hosts

Applications running in the end-hosts at the access networks are QoS aware. These applications are either adaptable or non-adaptable. Of course there will always be support for the traditional applications as well. But then this particular architectural framework is concerned with QoS aware applications, which are RSVP aware also. In case of Intserv access networks they should generate appropriate RSVP signalling based on their QoS requirements.

- RSVP/Intserv Access Network

  The access networks have set up RSVP/Intserv architecture, which indicates that it provides RSVP signalling with Intserv parameters, Intserv admission and policy control of the traffic at network elements. This is a mandatory, even if it is applied at only the first hop.

- Diffserv Core Network

  The architecture deployed in the core network is the Diffserv architecture, supporting aggregate traffic control and offering at least two levels of service, e.g. best effort and assured service. It may consist of a single administrative domain or several administrative domains, which may be spread all the way to the edges. By carefully enforcing the traffic contracts between boundary domains the Diffserv architecture is capable to provide QoS across multiple domains.

- Resources in Diffserv core network domains are statically provisioned – static SLA, SLS respectively.

  The resource management mechanisms in the Diffserv domain are not automated; thus the resources are statically provisioned, such that the edge devices are configured manually by network administrators in compliance with the negotiated SLSes.

- Intserv/Diffserv Customer – Provider relationship

  In the Intserv/ Diffserv architectural framework, the RSVP/Intserv access networks and Diffserv core network have a customer – provider relationship, that is the RSVP/Intserv access networks is just another customer of the Diffserv core network. Since the resources in the Diffserv core network are statically provisioned, the contracts between the two domains are negotiated manually; thus the SLSes are static. The SLS may contain only certain bandwidth requirements or it can be a more complex profile containing other traffic-related parameters.

  The SLS contains the technical details of the agreement specified by the SLA. In this thesis the definition and description of SLS given in [Techi99] is to be used. For convenience it is repeated here: "An SLS, more specifically, asserts that the traffic of a given class, meeting specific policing conditions and entering the domain on a given link, will be treated according to a particular (set of) PHB(s). If the destination of the traffic is not in the receiving domain, then the traffic will be passed on to another domain (which is on the path toward the destination according to the current routing table state) with which a similar (compatible and comparable) SLS exists specifying an equivalent (set of) PHB(s)".

- Diffserv is transparent to RSVP/Intserv messages

  Diffserv domains may be RSVP aware, such that they may include in their domain network elements, which are RSVP aware and therefore capable of performing per-flow signalling and admission control. Diffserv may use in this case RSVP for its intra-domain resource provisioning or also for inter-domain dynamic resource provisioning.

Therefore in these cases in order to keep the Diffserv completely transparent of messages coming from the RSVP/Intserv domain, these messages will be tunnelled through the Diffserv domain. Tunnelling the RSVP messages through Diffserv makes it independent of the specific realisation of the Diffserv domain. There will be a detail description of this mechanism in Chapter 5.

- The Diffserv Edge Device (Border Router) will be RSVP/Intserv – Diffserv (RID) Router

  The edge device or the router, which will have the main role in Intserv/Diffserv interoperation, is the boundary router residing at the Diffserv region. This requirement is in line with the design objectives mentioned above and was motivated by the following design choices:

  - The Intserv region is considered a customer of the Diffserv router, so it is quite natural that the owner of the Diffserv region will try to satisfy its customer needs
  - this design will be easily extended for dynamic provisioning of the Diffserv region, when there will be appropriate mechanisms
  - there are no changes necessary at the edges of RSVP/Intserv access
- The Diffserv Edge Device (Border Router) will perform tunnelling functions related to RSVP tunnel

  The Diffserv boundary routers will also be the endpoints of the RSVP tunnel mentioned above, which is logically derived from the previous requirement.

3.4 The General Intserv/Diffserv Architectural Framework Design

Based on the above mentioned design objectives and requirements a general Intserv/Diffserv architectural framework design results in a specific realisation with clearly defined functions of the framework entities.

The general Intserv/Diffserv architectural framework is shown in Figure 3-4.1.

The Intserv/Diffserv architectural framework is based on static provisioning of the Diffserv region, since the dynamic provisioning mechanisms are not yet clearly specified. The Intserv region, which is the customer of the Diffserv region, has negotiated a contract with the owner of the Diffserv region. This is a static contract - a static SLS, which defines the service (and its parameters) to be provided by the Diffserv core network to the RSVP/Intserv access networks. The network administrators statically configure these parameters at the edge devices either manually or automatically (e.g. by means of some protocols) conforming to negotiated SLS. For each customer there is at least one SLS negotiated and configured at the edge devices. The way in which the edge device at the Diffserv domain will handle SLS from different customers depends on the administrator of the Diffserv region. The decision on whether one edge device will handle several customers or whether there will be one edge device handling one customer is depending on the traffic engineering mechanisms deployed in Diffserv domains. In both cases though the network administrator should take care that the negotiated SLS are not violated.

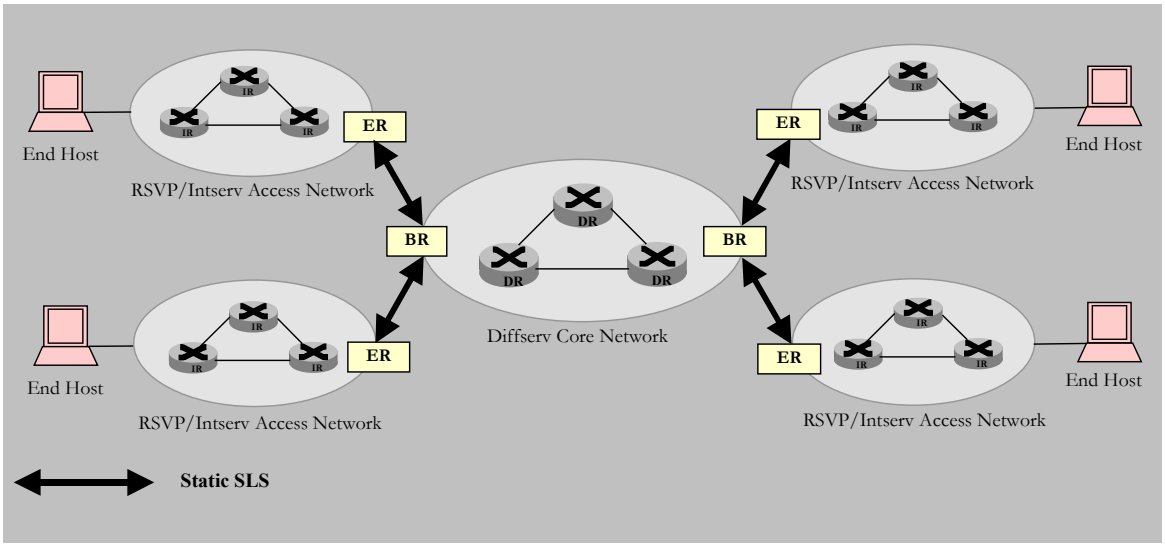


Figure 3.4-1 The Intserv/Diffserv Architectural Framework

3.4.1 The Intserv / Diffserv Architectural Framework Routers

Intserv/Diffserv architectural framework identifies several types of routers (see Figure 3.4-1):

- Intserv Routers (IR), which are standard RSVP/Intserv routers, performing per-flow RSVP signalling, admission control, policing (if set) and scheduling.
- Edge Routers (ER) residing at the border of the RSVP/Intserv are standard RSVP/ Intserv routers same as the other Intserv routers (IR) shown in the RSVP/Intserv access networks. They will also perform per-flow RSVP signalling, admission control, policing and scheduling. Edge Routers (ER) might also perform some additional functions related to Intserv/Diffserv interoperability in case the Intserv/Diffserv architectural framework is extended to support dynamic SLS. For instance communication with the Bandwidth Broker (BB) in Diffserv domain or as Aggregator/Deagreggator if RSVP aggregation is used etc, depending on the neighbouring domain resource management mechanisms. The RSVP/Intserv router architecture is explained in detail in Section 4.1.1.
- The Diffserv Core routers (DR) routers are standard Diffserv routers, which should apply appropriate PHB to packets based on their DS codepoint. They may perform some limited traffic conditioning functions e.g. Diffserv remarking and also more complex traffic conditioning functions on the internal intra-domain traffic, in which case they will be analogous to the Diffserv boundary network element.

  Since, RSVP signalling messages are encapsulated, the Diffserv Core Router will not process them, even if they are RSVP aware.
- Diffserv Border Routers (BR) interconnect the Diffserv network either to other Diffserv domains or to RSVP/Intserv domains. These routers are required, to perform traffic conditioning functions as defined by a traffic conditioning agreement (TCA that is part of SLS) between their Diffserv domains and the peering domain which they are connected [RFC 2475]. Further, they perform the RSVP/Intserv-Diffserv interoperability functions when they are connected to RSVP/Intserv domains, that is they handle both RSVP/Intserv and Diffserv messages, perform admission control on RSVP messages for the entire Diffserv

domain and service mapping. At the same time they will also perform tunnelling functions on RSVP messages. Further in this thesis this router will be referred as RSVP/Intserv-Diffserv (RID) border router as defined in the requirements above.

Just as in [RFC2475] these RID Routers act both as a Diffserv ingress node and as a Diffserv egress node for different directions of traffic. Traffic enters the Diffserv network at a ingress RID Router that is responsible for ensuring that this traffic conforms to any TCA between it and the other access network to which the ingress node is connected, i.e. RSVP/Intserv access network or another Diffserv network. Traffic leaves the Diffserv network at the egress RID router which may perform traffic conditioning functions on traffic forwarded to a directly connected peering domain, depending on the details of the TCA between the two domains.

In general, independently on their location in the network, the design of RSVP/Intserv routers and Diffserv routers architecture is global, such that the building blocks and their overall functionality are defined regardless of their network positioning. Although in the case of Diffserv routers some of these building blocks functionality will not be necessary, e.g. interior routers or particular interfaces of border routers.

An explanation of the main functional building blocks of the RSVP/Intserv and Diffserv router architectures is given in sections and respectively.

3.4.2 Intserv/Diffserv Architectural Framework - Example

The sequence of events happening from the moment when the QoS aware application at Host1 initiates the RSVP process until the moment when this host receives the recognition of granted resources is shown in Fig. 3.4.1-1 and explained below:

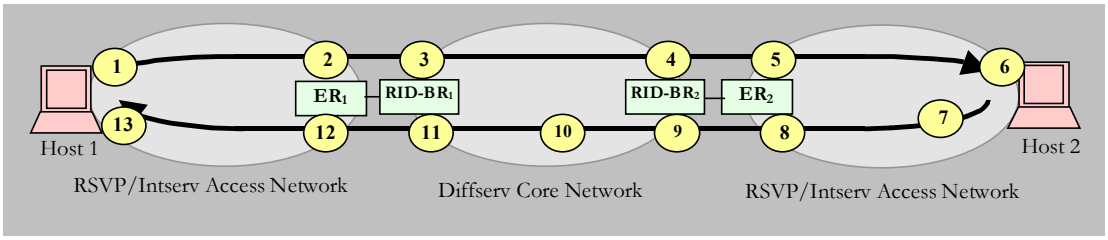


Figure 3.4.2-1 Sequence of events in the Intserv/Diffserv architectural framework during end-to-end communication

1. Host1 generates RSVP PATH Message
2. PATH message installed in ER1 and sent to Diffserv
3. RID-BR1 processes PATH message, encapsulates it and tunnels it through Diffserv
4. RID-BR2 receives the PATH message, processes it, decapsulates it and sends it to ER2
5. PATH message processed in ER2 in standard RSVP/Intserv processing manner
6. PATH message reaches Host2, which in turn generates the RESV message
7. RESV message is carried back to the to Host1
8. ER2, processes the RESV message in a standard manner and if it is not rejected it is sent to RID-BR2
9. RID-BR2 processes RESV message, encapsulates it and tunnels it through Diffserv

10. Diffserv core routers carry encapsulated RSVP messages transparently, without processing them
11. In RID-BR1 RESV message once decapsulated, triggers admission control according to the static SLA/SLS respectively. If the requested resources are available then the proper classifier is configured for classifying the data flow into appropriate PHB based on the configured mapping. Afterwards RSVP message is sent to ER1.
12. ER1 processes the RESV RSVP message in an Intserv manner and once the reservation is accepted, RESV message continues upstream
13. Host1 receives the RESV RSVP message and starts sending data packets that will be processed by the Diffserv as standard Diffserv packets
14. The flow data packets arriving at the RID-BR1 will be classified and remarked with an appropriate DSCP based on the configured service mapping.
15. In the Diffserv core routers they will be processed as standard Diffserv packets
16. Once they arrive in the Intserv they will be treated in an Intserv manner.

The overall flow of signalling and data packets within a single Diffserv is shown in Figure 3.4.1-2. As it is shown the RSVP signalling messages are carried transparently thought the Diffserv Core routers (DR), that is, these routers do not process these packets since they can't see them. RSVP signalling messages are carried via the RSVP tunnel. The incoming and outgoing data packets from/to RSVP/Intserv access networks will be processed as standard Diffserv packets. Thus, border routers (RID-BR1 and RID-BR2) have RSVP control plane and a Diffserv data plane.

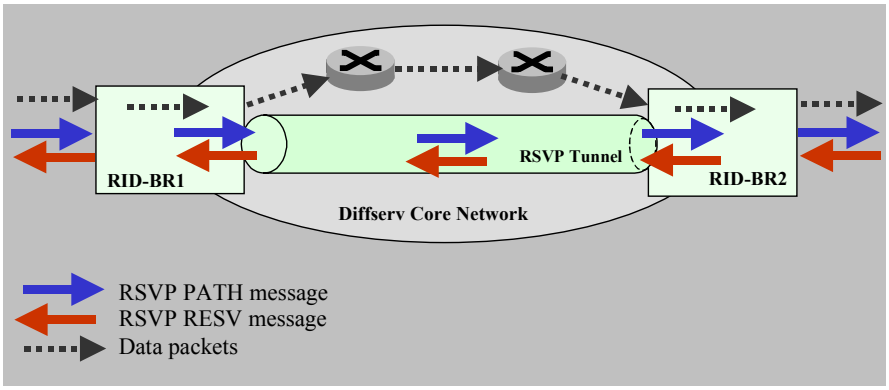


Figure 3.4.2-2 Signaling and Data packets flow in the Diffserv Core Network

[4] RSVP/INTSERV – DIFFSERV (RID) BORDER ROUTER

4.1 Introduction

A requirement of the Intserv/Diffserv architectural framework is that the main interoperability functions are to be performed by the boundary network element residing at the Diffserv domain. These boundary network elements are the Diffserv Border Routers. The Border Router is the Diffserv RSVP aware router that conformant with the design objectives and requirements will be the RSVP/Intserv-Diffserv translator. This router will also be performing tunnelling functions related to the RSVP tunnel.

This chapter focuses on the RSVP/Intserv-Diffserv (RID) Border Router architecture. Firstly, a description of the standard RSVP router and Diffserv router building blocks and their functions is given. Then the conceptual model of the RSVP/Intserv-Diffserv (RID) Border Router is described, followed by the RID Border router architecture. The building blocks of this router architecture are also described with the special focus on the service mapping as the main building block for the Intserv/Diffserv interoperability. The role of this router as part of tunnelling functions is explained in Chapter 5.

4.1.1 RSVP/Intserv Router Architecture

The RSVP/Intserv router architecture is designed in such a way that on its interfaces it should support the RSVP process jointly with the routing process, that is the RSVP process will get access to the routing process also. The RSVP process will also need to interact with the traffic control mechanisms and also the admission control and policy control mechanisms.

The RSVP/Intserv router architecture with its building blocks is depicted in Figure 4.1.1-1.

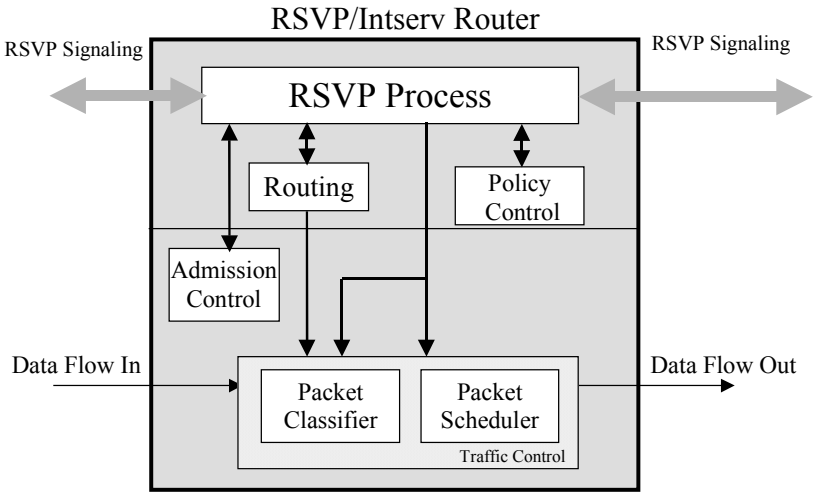


Figure 4.1.1-1 RSVP/Intserv Router Architecture

The functionality of each of these building blocks is explained briefly in the following paragraphs:

RSVP process

The RSVP process in the router will have to take care of the processing of the RSVP messages and accordingly to their interoperation utilise appropriate router functions. This process is responsible for the RSVP state maintenance that is it will maintain the path and reservation states, but it will also process the corresponding error and tear messages.

A PATH message will trigger the formation of PATH State Block (PSB), while RESV message will cause the creation of the RESV state block (RSB).

The PSB will maintain the information corresponding to the PATH message, the related outgoing interface and whenever there will be new PATH messages coming in, the PSB will be checked in order to determine whether these incoming messages are refresh messages, updates or entirely new.

Like the PSB, the RSB will maintain all the relevant information contained in the RESV message and all incoming RESV messages will be compared with the RSB in order to determine whether they are new, updated or refreshed. For new and updated RESV messages in a normal functioning there should always be an applicable PSB, since Path State is installed before the RESV is issued.

Besides maintaining the PSBs and RSBs, RSVP process will interact with traffic control mechanisms to establish adequate handling of flows. Furthermore it will perform admission control on an outgoing interfaces. Finally by installing adequate flow information in the traffic control components, the RSVP process takes care of proper traffic provisioning.

Routing

Routing in the RSVP/Intserv router relies on interaction with the RSVP process. All RSVP messages and data packets belonging to the same flow will be forwarded on the same interface, which is, determined by the RSVP process itself. RSVP process will interact with the routing process to select the appropriate local previous hop (PHOP) interface.

Policy Control

The Policy control in the RSVP/Intserv router is responsible for enforcing network administrator's policies on the network resources. This block determines who gets the rights to make the reservations. As a component for controlling policy it is also responsible for interoperation of the RSVP Policy data object.

Admission Control

The admission control component on the incoming RESV message determines whether this request can be granted or not, by taking into account all the currently provisioned RSVP reservations. If there are enough resources available, the admission control component will grant this request, otherwise a RESV error will be issued.

Traffic Control

The traffic control component provides the specific link layer classification, scheduling and policing capabilities. These components are actually the ones providing the QoS functionality.

The packet classifier of the RSVP/Intserv router will perform the necessary filtering of packets based on the header information of the data packet. The appropriate filter parameters are created as soon as the reservation flow has been accepted by the admission control mechanisms.

The packet scheduler is related to the outgoing interface and it will handle the classified packets, such that their QoS requirements are satisfied. The scheduler will take care that the incoming either bursty or non-bursty traffic will be transmitted in the proper service profile on the outgoing link.

4.1.2 Diffserv Router Architecture

The Diffserv router architecture is designed in such a way that it should always on its interfaces support the service model applied by the service provider, i.e. the service model applied by the network administrator of the Diffserv domain. The Diffserv architecture enables a variety of services that can be deployed in a network, which are specified by means of an SLS. The standard Diffserv router can have separate virtual transmit interfaces for each SLS with a different customer or it can have a single interface for all its customers, but then it also provides the mechanisms to recognise the different customer flows. Therefore the customer and/or applications do not observe the network directly. Instead, they only look at the structure of the SLS specified by the service provider. In order to provide such services, Diffserv architecture identifies a set of components for traffic handling in Diffserv routers.

Diffserv architecture provides service differentiation by combining the functionality of the traffic handling components. The appropriate combination of these components is chosen on packet basis based on the DS codepoint. Therefore, Diffserv can be seen as a logical structure, because it relies on the combination of the functionality of the traffic handling components and not on their specific characteristics.

The traffic handling components are part of each Diffserv router interface. These components are depicted in Figure 4.1.2-1. Thus, each Diffserv router interface provides network service differentiation by means of traffic classifiers, traffic conditioning components and schedulers that support the per-hop behaviours [RFC2475]. Each of these components has their specific design and characteristics, which are to be explained below:

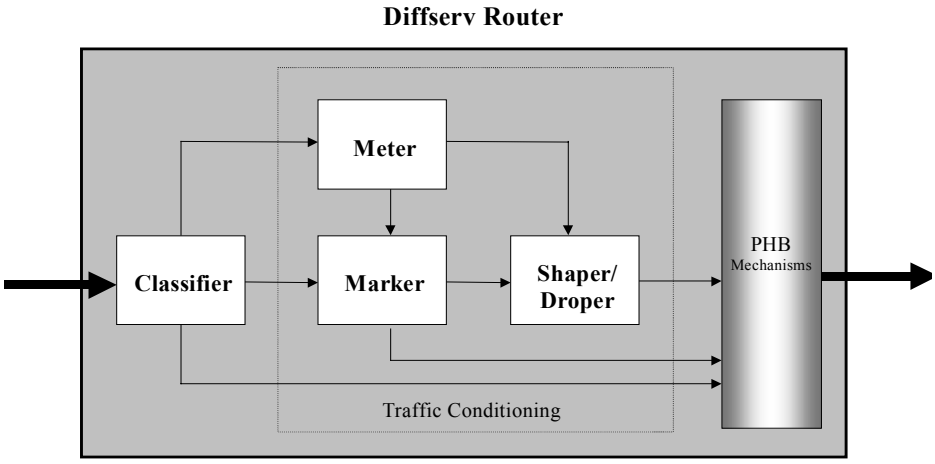


Figure 4.1.2-1Diffserv Router Architecture

Classifier

Classifier is the component, which classifies the incoming traffic into several output traffic streams by selecting the PHB class for each traffic stream. Incoming packets are sorted into several output streams by filters, which sort the incoming packets by matching the header content (a single or several fields in the header) of the packets, which determines the output stream of the packet.

A filter consists of a set of conditions on the component values of a packet's classification key (header values, contents, and attribute relevant for classification) [BeSm00].

There are two types of classifiers: the behaviour aggregate (BA) classifier and the MultiField (MF) classifier. In the BA classifier packets are classified based on only one header value, that is the DS field. In the BA classifier filters specify the exact match conditions on the value of DSCP. The advantage of this classifier is that they are very simple and there will be no scalability problems, since the number of states needed to be maintained by the router is limited.

In the case of the MF classifier the classification is to be performed based on several fields in the packet's header. The common type of MF classifier is a 6-tuple classifier, whose filter is parameterised by 6 header fields (source address, destination address, IP protocol, source port, destination port and DSCP). The MF classifier may also classify packets based on other fields from the packet's payload such as higher layer protocol fields, MAC addresses etc. These types of classifiers are advantageous for use in cases where the incoming stream belongs to different SLS. But because of the number of states it needs to hold, there may be scalability problems introduced.

An example of a classifier as given in [BeSm00] is depicted in Figure 4.2.2-2.

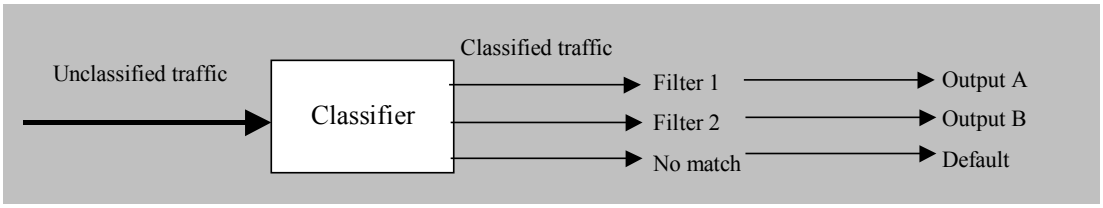


Figure 4.1.2-2 Classifier

This classifier may be a BA classifier or a MF classifier. In the case of BA, Filter 1 and Filter 2 are parameterised such that they perform exact match on DSCP values as given in Figure 4.1.2-3. The packets that do not match any of these values will be treated by default, which is to be set by the network administrator. Default can be for e.g. best effort service.

Filter	**DSCP**	**Output Stream**
Filter1	101010	A – EF PHB
Filter2	111111	B – AF PHB
No match	xxxxxx	Default (Best Effort)

Figure 4.1.2-3 BA Classifier's Filters

If the classifier is a MultiField classifier then Filters 1 and 2 are parameterised by a 6-tuple value as shown in Figure 4.1.2-4.

Filter 1 of type IP DSCP 1
IP version (e.g. 4)
DSCP byte (e.g. 101010)
Destination Address
Source Address
Destination Port
Source Port

Filter 2 of type IP DSCP 2
IP version (e.g. 6)
DSCP byte (e.g. 111111)
Destination Address
Source Address
Destination Port
Source Port

Figure 4.1.2-4 MF Classifier's Filters

Which type of classifier is going to be used depends on the specific requirements to be met by a certain Diffserv domain, thus it is a decision of domain administrators, which will configure the classifiers to comply with the negotiated SLAs.

Meter

There are two definitions regarding Meters. By definition of [RFC2475], meter is the element responsible for monitoring the arrival times of packets of a traffic stream and for sending control signals to action elements to modulate their behaviour based on the conformance of the packet dynamically. The [BeSm00] on the other hand specifies metering as a function of monitoring the arrival times of packets of a traffic stream and determining the level of conformance to a pre-established traffic profile. Further in this thesis we adopt the last definition of metering as a stronger one.

The meter' s functionality is based on the temporal profile and the conformance levels, related to certain metering outputs. The temporal profile (i.e. average rate, burst size) is chosen by the Diffserv network providers based on SLS, within which the customers receive a service for their traffic. Based on this profile the meter determines whether the customer's traffic is conforming or not, i.e. whether it is in-profile or out-of-profile. Depending on the policy decisions of the Diffserv network providers the conformance levels may be used to trigger actions in the other traffic conditioning blocks, i.e. marker or shaper/dropper.

The basic functionality of the meter is given in Figure 4.1.2-5. Say this meter is the average rate meter, which is the simplest one. This meter measures the rate at which the packets are submitted and based on the temporal profile (Profile A Figure 4.1.2-5) determines whether the packets are conforming. Profile A consists of an average rate say A_r = 100 kbps and Δ = 0.5 msec. If N – total number of packets arriving between T (actual time) and T - Δ, the meter would continually

count N. If for a packet arriving at time t, where $t = T + \Delta$ and T (actual time), $A_r(t) \leq A_r$, than the traffic will be in profile that is conforming, otherwise it will be non-conforming.

The [BeSm00] gives a whole list of different possible realisations of meters. It is worth mentioning the null meter, which is a meter with no profile; i.e. all packets are always conforming.

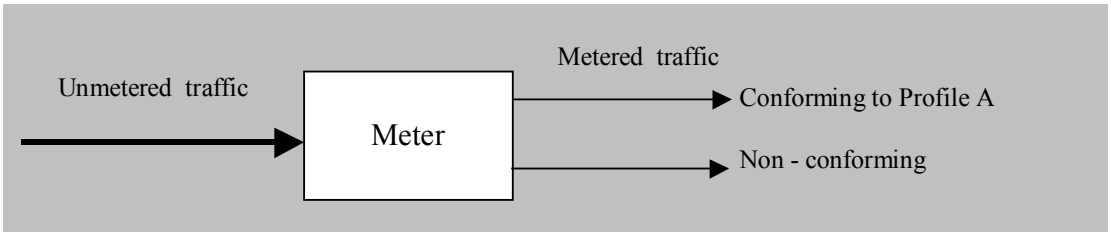


Figure 4.1.2-5 Meter

Marker

Marker sets the DSCP in an IP header in the case of unmarked packets (submitted with DSCP of zero) or may re-mark previously marked packets. The DSCP set in a packet will determine its subsequent treatment in downstream nodes of a network, and possibly in subsequent processing stages within the router (depending on the configuration). The main objective of packet marking functionality is to map packets into the PHB class used by the flow. The marking itself depends on the measurements results on particular PHBs, the amount of traffic sent by the customer and also traffic load on the network [Kil99]. The Marker has only a single parameter, i.e. the 6 – bit DSCP that needs to be marked.

Shaper

Shapers are used for shaping traffic according to a certain temporal profile that is defined by the meter [RFC2475]. In the [BeSm00] shaper's functionality is embedded in the queuing components. As such shapers perform functions necessary for delaying the departure of packets that are already declared as non-conforming by the meter. The departure time of the packet is set such that the outgoing packet is in-profile. Delaying packets may result sometimes in packet loss due to unavailable buffer space in the shaper, which would happen in cases when the incoming traffic is very different from the temporal profile.

Dropper

Dropper is the element that simply discards the packets based on some predefined rules, in order to keep in-profile with certain temporal profiles. The dropper has no parameters and it may use information from other elements like the classifier, marker and meter to make a decision on packet dropping.

Queuing Block

PHB mechanisms define locally how the packets are going to be forwarded at the egress interface based on the PHB they belong to. The [BeSm00] defines a queuing block as a logical abstraction of a queuing system that is used to configure PHB related parameters. This block modulates the transmission of packets belonging to different traffic flows by reordering, storing them or in some cases also discarding them. Thus the queuing block are constructed by the elements performing these functions:

- First-In-First-Out (FIFO) which is a data structure that permits items to be removed only in the order they where inserted. It has one or more inputs and only one output

- Scheduler which takes care of the departure of each arriving packet according to a service discipline. It consists of several inputs and exactly one output, where each input has an upstream element to which it is connected and a set of parameters that affects the scheduling itself. The service discipline is an algorithm, which may take as its input static parameters (e.g. token bucket rate for maximum or minimum rates) associated with each of its inputs, or parameters associated with a packet present at its input or absolute time and /or local time. There are several categories possible for service discipline, such as First-Come-First-Served, strict priority, Weighted fair Queuing (WFQ), Weighted Round Robin (WRR), etc.
- Discarder is the element which discards the packets selectively based on a discarding discipline. The discarding discipline is the algorithm that makes a decision on whether a certain packet at its input should be discarded or forwarded. The examples of discarding disciplines are Random Early Detection (RED), RIO [ClWr97], dropping on threshold algorithms. The Discarder takes as its parameters a set of dynamic parameters, static parameters and possibly parameters associated with the packet, like the PHB as set by the classifier.

4.2 RID Border Router

RID Border Router is the router that will carry out the necessary functions related to the RSVP/Intserv and Diffserv interoperability. Conformant with the design decision this router will have to handle RSVP messages and Diffserv packets at the same time, that is, it will be able to translate the RSVP/Intserv services into Diffserv services. It is can be seen as a Diffserv RSVP aware router, where RSVP is used as a signalling protocol, thus only active in the control plane. While in the data plane the RID is still a purely Diffserv router.

The design of the RID Border Router architecture should be in full compliance with the design goals and requirements of the Intserv/Diffserv architectural framework (see 3.2,3.3).

4.2.1 Conceptual Model of the RID Border Router

Since the RID Border Router is a Diffserv router its conceptual model will be similar to the concept model of Diffserv Router described in the [BeSm00], which proposes a generalised conceptual model of Diffserv routers, independent of whether they are border routers or intermediate routers. The RID Border router concept model should also include the necessary components for RSVP/Intserv to Diffserv translation.

Generally, custom routers consist of a routing core and several interfaces connected to this routing core, which actually represents an abstraction of their functionality. The Diffserv routers also consist of a routing core and several interfaces connected to it. The interfaces depending on the direction of the traffic will function either as ingress or as egress interfaces, each of them containing the functional elements for traffic handling in the router as described in the Section 4.1.2. Thus each interface will contain classifiers, traffic conditioning elements and queuing components that support the traffic conditioning a PHBs. The abstract representation of the Diffserv router concept model is depicted in

Figure 4.2.1-1. The management and configuration data (Mgt&Config data) is the abstraction for the interface via which Diffserv will perform monitoring and provisioning of parameters such as classification rules, traffic conditioning and PHB configuration parameters. Network

administrator interacts with the Diffserv Mgt&Config data by means of mechanisms and protocols, such as SNMP, COPS or Telnet consoles, etc.

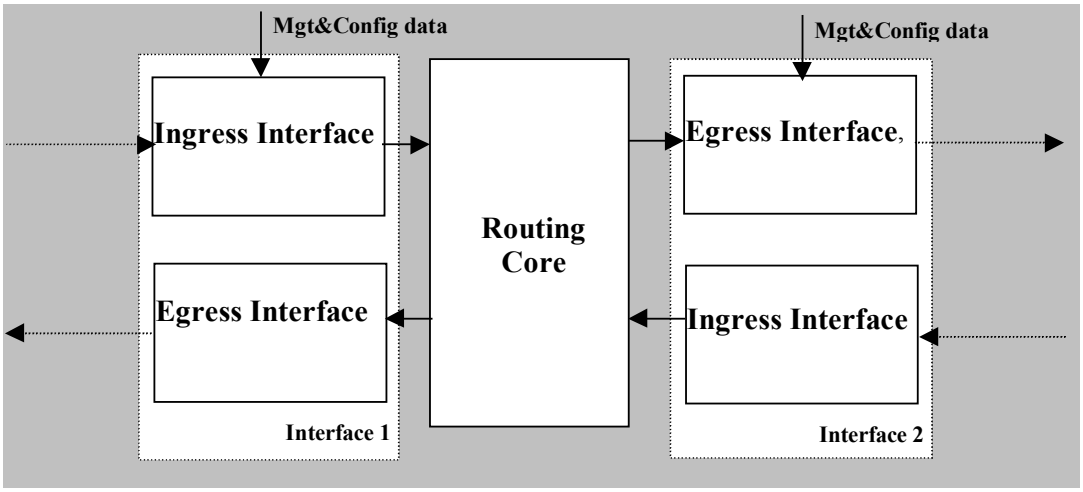


Figure 4.2.1-1 Abstract representation of Diffserv Router's Egress/Ingress Interfaces

Figure 4.2.1-1 is also valid as a representation of the RID Border Router conceptual model. But than in case of this router the ingress and egress interfaces will have additional elements for traffic handling of the RSVP signalling messages and for performing appropriate service translation from RSVP/Intserv to Diffserv. The traffic handling functional elements in this case is to be divided into Diffserv packet handler and RSVP message handler. Diffserv packet handler consists of the standard Diffserv traffic conditioning elements and queuing elements as described in Section 4.1.2, while the RSVP message handler will perform a limited set of a standard RSVP /Intserv router functions, which have been already described in 4.1.1. The decision on whether the incoming traffic is to be treated in the RSVP message handler or Diffserv packet handler is executed by the Ingress interface, which is configured to classify the RSVP messages and data packets. The RSVP messages and Diffserv packets will be handled separately in the router by the RSVP message handler and by the Diffserv packet handler. Based on the above description the concept model of the RID Border Router is shown in depicted in Figure 4.2.1-2. The main functional elements are Ingress classifier, RSVP message handler and standard Diffserv packet handler.

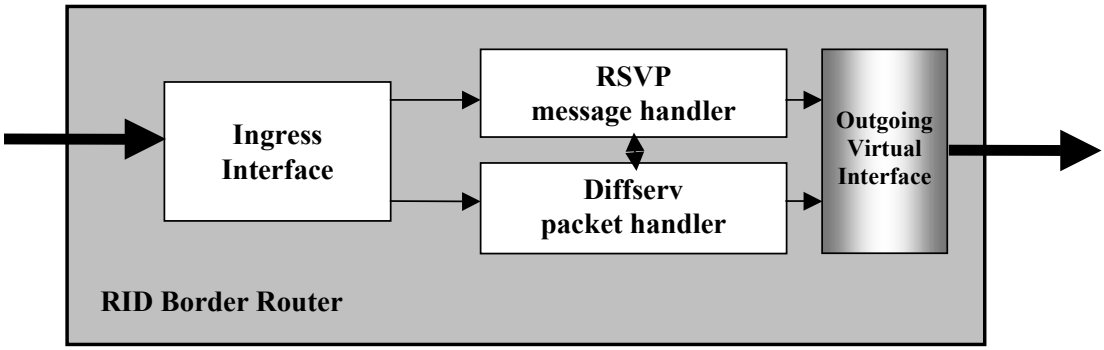


Figure 4.2.1-2 Conceptual Model of the RID Border Router's functional elements

4.2.2 RID Border Router Architecture

The architecture of the RID Border Router is in full compliance with the design goals and requirements of the Intserv/Diffserv architectural framework. The RID Border Router architecture and its functional entities are based on the conceptual model described above as it can be seen from

Figure 4.2.2-1.

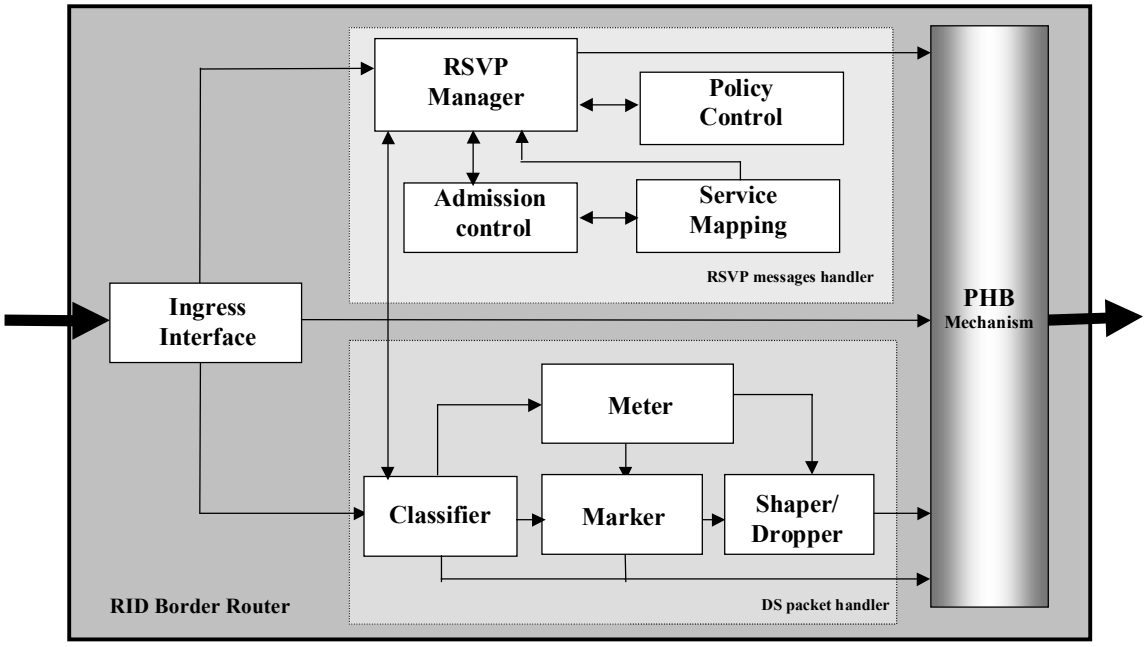


Figure 4.2.2-1 RID Border Router Architecture

The RID Border Router architecture functional elements for traffic handling can be divided into traffic classifiers, RSVP messages handler elements, Diffserv traffic conditioning elements and queuing elements for handling PHB related parameters.

Diffserv traffic conditioning elements and queuing elements for handling PHB related parameters are exactly the same as in Diffserv standard routers described in Section 4.1.2. Logically, the network administrator of the Diffserv domain will have to configure these elements appropriately in order to comply with the SLA and SLS respectively negotiated with the Intserv access network and other domains.

The traffic classifier of the RID Border Router has also the same functionality as the classifier in the standard Diffserv routers (see 4.1.2). In the case of the RID Border Router the classifier should be configured such that it classifies the RSVP signalling messages and the Diffserv packets. This classifier can be configured as BA Classifier or as a MF classifier. Depending on whether the classifier is BA or MF, the filters may be parameterised by a single parameter or by several parameters. Filters for classifying RSVP signalling messages and Diffserv packets and their resulting output streams are given in Figure 4.2.2-2.

Filter	Output Stream
Filter 1	A – RSVP packet handler
Filter 2	B – Diffserv packet handler
No match	Default (Best Effort)

Figure 4.2.2-2 RID Classifier's Filters

If the classifier is a BA Classifier than the Filters 1 and 2 are configured to perform matching only on a single field from the packet header. Thus, Filter 1 will perform matching on the protocol ID filed:

```
if ( protocol_ID = 46)
   OUTPUT_STREAM A
else
   DEFAULT
```

While Filter 2 will perform matching on DSCP bits:

```
if ( DSCP = 101010)
   OUTPUT_STREAM B
else
   DEFAULT
```

Filter 1 and Filter 2 can also perform matching on several packet fields as for e.g. given in Figure 4.2.2-3, where the IP version can be 4 or 6:

Filter 1 of type IP RSVP
IP version
Protocol ID (46)
Destination Address
Source Address
Destination Port
Source Port

Filter 2 of type IP DSCP
IP version
DSCP byte
Destination Address
Source Address
Destination Port
Source Port

Figure 4.2.2-3 RID Classifier's Multifield Filters

The RSVP message handler elements include the RSVP manager, Admission Control, Policy Control and Service Mapping. The RSVP manager performs similar functions as the RSVP Process described in Section 4.1.1; thus it will take care of the processing of the RSVP messages and accordingly to their interoperation utilise appropriate router functions. It will maintain the path and reservation states of the incoming RSVP messages, but it will also process the corresponding error and tear messages. Upon receiving RSVP messages it will trigger policy and admission control. RSVP manager will also in co-operation with routing determine on which interfaces the RSVP messages are to be forwarded and tunnelled. Policy Control and Admission control have the same functionality as in standard RSVP/Intserv routers, but now they will be configured to perform admission control on RSVP messages based on the resources of the entire Diffserv domain, conforming to the SLA, SLS respectively. Once a reservation request is accepted the Service Mapping will be triggered to perform functions necessary for mapping of RSVP/Intserv services to Diffserv services.

The Diffserv architecture as it is already said can be seen as a logical architecture, where the service differentiation is performed by combining a set of functional elements. The same can be said also for the Intserv/Diffserv architectural framework. Yet, the Service Mapping functional element is very important for the overall functionality of the Intserv/Diffserv architectural framework. This element and related issues are explained in more detail in the following section.

4.2.2.1 Service Mapping Element

There are different ways on how to map the Intserv services to Diffserv services as described in [BeYa00], [WrCh00], [AlSi00]. The service mapping element will perform the mapping of Intserv

to Diffserv services that was chosen by the designer of the RID router in accordance with the design objectives and requirements of the Intserv/Diffserv architectural framework. As described in Section 2.4.2 service mapping in [BeYa00] is still an open issue. The default mapping, which is mentioned in [BeYa00], is already explained in Section 2.4.2. [WrCh00] describes service mapping of Intserv into Diffserv services and gives a detail description on how the Intserv services can be implemented using Diffserv network behaviours and mechanisms.

In this thesis the service mapping element will perform the functions that will be implemented using the [WrCh00] as the main reference. Thus, the service mapping element will perform mapping of Intserv services to Diffserv services on the RSVP RESV packets that already passed the policy and admission control and accordingly set the appropriate PHB. The result of the service mapping will be sent further to the classifier. Eventually this RESV message will arrive at the appropriate host, if it is not refused upstream. The data packets transmitted arriving at the RID border router will be identified by the classifier and marked with the proper DSCP byte based on the info received from the service mapping element. Further data packets will be processed in a standard Diffserv manner.

The implementation proposal of [WrCh00] of Intserv services to Diffserv services is summarised in

Intserv Services	**Diffserv PHB**
Controlled Load (CL)	AF PHB
	EF PHB
Guaranteed Services (GS)	EF PHB

Table 4.2.2-1 Service mapping as proposed in [WeCh00]

4.2.2.1.1 Implementation of Controlled Load Service

As it is shown above, the [WrCh00] proposes implementation of Controlled Load service to AF PHB but also to EF PHB. In this thesis the service element of RID Border Router will perform mapping of CL to AF PHB, that will be explained in the following paragraphs. The mapping of CL to AF PHB is also recommended by [WrCh00] that is the mapping of CL to EF PHB should be used only if AF PHB is not available. However, in the Intserv/Diffserv architectural framework, the availability of AF PHB is not questionable, that is both AF PHB and EF PHB must be available at all times.

The Controlled Load (CL) Service traffic can be sorted out in delay classes based on the token rate b/r given in the CL TSpec parameters (see Section 2.2.2). Flows with a higher burstiness, that is larger token rate will experience more queuing delay that the flows belonging to the less bursty flows. So a reasonably effective CL implementation method will group the flows with roughly the same queuing delays into sub-classes, which will be handled independently [WrCh00]. The TSpecs of flows belonging to the same sub-class will be aggregated into a single TSpec, based on which this traffic will be policed and marked accordingly with the appropriate DSCP indicating the selected AF instance and highest priority forwarding within that instance. The non-conformant packets will be marked with a DSCP indicating the selected AF instance and lowest priority forwarding within that instance.

This will be explained by the following example: for each AF class and highest priority instance there is a TSpec aggregate defined say T_{a1}, T_{a2}, T_{a3}, T_{a4}. In order to associate single TSpec flows with an aggregate flow b/r ratio is checked:

```
      if b/r < 2 than Ta1
else
      if  b/r <4 than Ta2
else
      if b/r < 6 than Ta3
else
      if b/r < 8 than Ta4
```

where $T_{a1} = \Sigma T_{ai}$ for all TSpec that have b/r <2. It is the same for Ta2, Ta3 and Ta4 also. Flows belonging to the T_{a1} will be marked with AF1.1 DSCP byte. Similarly flows belonging to T_{a2}, T_{a3}, T_{a4} will be marked with AF2.1, Af3.1 and AF4.1 DSCP byte.

The Diffserv network as stated in [WrCh00] should be configured such that for each AF instance:

- the queue size is set accordingly to the queuing delay of the class's target
- the dropping parameters for low priorities are set such that these packets are dropped as soon as they are not fitting into the queue size set for their class
- service rate is set to a bandwidth sufficient to meet the delay and the loss behaviour requirements of the CL spec when only high priority class packets are present
- the admission control mechanism should ensure that at each hop in the network the level of traffic offered to each AF instance shouldn't exceed the level of the one provisioned in the previous requirements.
- The relationship between different AF instances, between AF PHB and best effort and between AF and EF PHB should be more tightly constrained than is required in the AF specification itself [RFC2597], because of the requirements of the service mapping functions, which need to be precisely defined in relation to TSpec token bucket rate.

4.2.2.1.2 Implementation of Guaranteed Service

As explained in Section 2.2.3 the Guaranteed Service (GS) offers strict delay bounds, assuming only that the network is functioning correctly. The delay values C (rate dependent delay - packet serialisation delay) and D (not-rate – dependent delay – propagation delay) are provided by the network element to the customer such that it can calculate the necessary bandwidth to achieve specific queuing delay target. There are two important issues needed to be considered when implementing the Guaranteed Service:

- providing traffic scheduling, policing and shaping mechanisms needed to provide hard bounds on performance
- determining the delay values such that the customer can accurately determine the network path and deduce what level of resources must be requested.

The EF PHB is appropriate for implementing Guaranteed Service, but than still there is a need to set shaping and policing mechanisms in order to get the necessary delay bounds. The delays in the Diffserv domain can be classified into propagation and serialisation delay, shaping/reshaping delays at the boundaries and queuing delay inside the domain. While in Intserv C and D values are local to a single network element, in Diffserv domain C and D can be defined as the delay values

for the whole domain including all types of delays mentioned above C_dc and D_dc [WrCh00]. These delay values depend not only on the topology but also on the characteristics of all EF traffic in the Diffserv domain. This means that the knowledge about topology and traffic characterisation should be centralised. Also the knowledge about the dependence of the delay bounds on traffic characteristics implies the existence of a policy which defines the traffic characterisation rules and implementation mechanisms in all ingress network elements. All these considerations imply that the delay bounds should be performed as part of e.g. traffic management algorithms, based on knowledge of the topology, traffic patterns, shaping policies and other relevant mechanisms.

In the Intserv/Diffserv architectural framework these bounds can be set as part of SLS between the Intserv and Diffserv domain, such that the Diffserv domain provides the appropriate implementations of the C_dc and D_dc values within its domain. Assuming that these values are set and appropriately implemented as suggested in [WrCh00], the Service Mapping element will perform the mapping of the RSVP RESV messages of GS to EF PHB in a straightforward manner and accordingly inform the classifier.

[5] RSVP TUNNEL

5.1 Introduction

Usually RSVP messages are sent through the network as raw IP packets with the protocol ID 46 and router alert option set to indicate to the routers that these messages require special processing. When the RSVP messages traverse non-RSVP cloud they are just ignored by the routers. In the same manner RSVP messages can also traverse the Differentiated Services domains region transparently, where the Diffserv routers will ignore them.

But, the Diffserv architecture may use RSVP as a signalling mechanism for its intra–domain resource provisioning, in which case RSVP messages coming from outside the Diffserv domain will not be carried transparently any more. Instead they will be processed by each RSVP aware router within the Differentiated Service architecture, which will introduce undesirable behaviour at the core routers in the Diffserv domain. There will be an interference of the intra-domain Diffserv RSVP signalling flows with Intserv RSVP signalling flows and also the processing of Intserv RSVP per flow at the Diffserv core routers will introduce the same scalability problems as in Intserv.

The Intserv/Diffserv architectural framework is designed such as to provide Diffserv transparency to RSVP messages coming from Intserv independently of what sort of mechanisms (dynamic or static) Diffserv uses for its resource provisioning. This transparency is provided by tunnelling RSVP messages in IP tunnels through Diffserv and can be used whenever it will be necessary, independently of static or dynamic provisioning of resources in Diffserv domains.

This chapter proposes a mechanism for setting of RSVP tunnels, which is motivated strongly by IP tunnels [RFC1853], [RFC2003]] simplicity and functionality and partially by [RFC2746]. However, it is in its own right entirely independent of them. It proposes a relatively simple way of tunnelling RSVP messages within a single domain or through multiple Diffserv domains. Additionally in this chapter an example of this RSVP tunnel functionality is given. Other ways of providing Diffserv transparency to RSVP messages are also described.

5.2 Diffserv Transparency to RSVP

There are several mechanisms either designed or suitable for providing Diffserv transparency to RSVP, such that RSVP messages will be carried invisibly through the Diffserv domains. These mechanisms are explained in the following sections:

5.2.1 RSVP Operation within IP tunnels

RSVP operation within IP tunnels, is a mechanism for reserving resources in IP tunnels, in order to extend RSVP usage to fixed and wireless networks [RFC2746]. It enhances IP tunnels with

RSVP capability, such that the end-points of the tunnel send RSVP PATH/RESV messages respectively in order to reserve resources for end-to-end sessions within the tunnel. Also the end-points of the tunnel have to map the per-flow end-to-end sessions to the tunnel session.

This RSVP over IP tunnels can also be used in the Intserv/Diffserv architectural framework such that the RSVP/Intserv packets traverse the Diffserv domains until they reach their destination domain, that is their exit point of the tunnel. In this case the Edge Routers (ER) (see Figure 3.4-1) will be the end-points of the tunnel.

However, even though this mechanism has its advantages in providing RSVP signalling in IP tunnels, it does not fit well with the design requirements of the Intserv/Diffserv architectural framework. Because, the intention is to only use the IP tunnels as pipes for carrying RSVP messages and not having RSVP aware IP tunnels.

5.2.2 RSVP Aggregation*

The RSVP aggregation [BaIt00] concept is used to reduce the Intserv scalability problems by extending the RSVP protocol with facilities for aggregation of individual reserved sessions into a common class and across transit domains. It describes mechanisms for dynamic creation of the aggregate reservation, classification of the traffic for which the aggregate reservation applies, determination of the bandwidth needed to achieve the requirement and recovery of the bandwidth when the sub-reservations are no longer required. A RSVP aggregated session is identified by the IP destination address, the protocol ID and the DSCP information. Furthermore, for classification and scheduling of traffic supported by aggregate reservations Diffserv mechanisms are used. Diffserv DSCPs are used to identify traffic covered by aggregate reservations and one or more Diffserv PHB's are used to offer the required forwarding treatment to this traffic.

The first router in the transit domain that handles the aggregated reservations is called Aggregator, while the last router in the transit domain that handles the reservations is called Deaggregator. An RSVP aggregation region is consisted by routers that are capable of managing the RSVP aggregated states.

The RSVP aggregation concept can be used either when RSVP is applied end to end or edge to edge. In the later case the Aggregator can use a policy that can be based on local configurations and local QoS management architectures, to set the DSCP packets that are passing into the aggregated region. For example, the Aggregator may be a PSTN (Public Switched Telephone Network) gateway that aggregates a set of incoming calls and makes an aggregate reservation across one or more Diffserv domains up to the Deaggregator that can be e.g., another PSTN gateway. In this situation the call signalling is used to establish the E2E reservations.

In the following paragraph the RSVP aggregation concept is used when RSVP is applied end to end.

Each end point can initiate a per-flow and end to end RSVP resource reservation procedure. The per-flow RSVP resource reservation messages are referred as E2E PATH/RESV and the aggregated resource reservations messages are referred as aggregated PATH/RESV. The E2E PATH/RESV messages are carried though the RSVP aggregation region transparently, since their RSVP protocol Id at the aggregator will be replaced with a new protocol number, i.e., RSVP-E2E-Ignore that is expected to be standardised. Each E2E RSVP resource reservation

* Sections 5.2.2, 5.2.3 on RSVP aggregation and MPLS are also part of [KaRe00]

request that arrive at the ingress router, i.e., Aggregator, of an RSVP aggregated region can initiate or resize a RSVP aggregated reservation. In the RSVP aggregated region the aggregated RSVP messages are managing the reservation of the resources for a set of per-flow RSVP reservations. The E2E RSVP reservation states are temporary states, i.e. soft states that have to be updated regularly. This means that E2E PATH and E2E RESV messages will have to be periodically retransmitted. If these states are not refreshed then they will be removed. These states may also be removed by using the E2E PATHTear and E2E RESVTear messages. When these E2E states are removed then their FLOWSPEC information must be removed from the allocated portion of the aggregate reservation, such that this same bandwidth will be re-used for other traffic in the near future. Furthermore, their SENDER_TSPEC information must also be removed from the aggregated state.

It is important to note that in [BaIt00] it is suggested that a predefined policy should exist to maintain the amount of bandwidth required on a given RSVP aggregated reservation that is taking into account the sum of its underlying E2E reservations, but that will protect it from changing it frequently. This can be achieved by using some level of trend analysis.

From the above information it can be deduced that based on a certain policy the Aggregator and Deaggregator will decide when the RSVP Aggregated states will be refreshed or updated and therefore this triggering time is not completely defined by the E2E RSVP messages.

Within an aggregation region three possible scenarios can be distinguished:

- Signalling Flow for the first E2E flow
- Signalling Flow for subsequent E2E Flow without reservation resizing
- Signalling Flow for subsequent E2E Flow with reservation resizing

The RSVP aggregation mechanism can be used in the Intserv/Diffserv architectural framework by using the E2E ignore for the RSVP messages, or in case of extension of the Intserv/Diffserv architectural framework to support dynamic SLAs it can be used as a single mechanism for inter-domain resource provisioning

5.2.3 MPLS as means of RSVP tunnel setup

The MultiProtocol Label Switching (MPLS) [RoVi00] is specified by the IETF to mainly be used in combination with the Diffserv concept and is an advanced forwarding scheme that extends routing with respect to packet forwarding and path controlling (see also [XiHa00], [Will99]).

The packet forwarding is used to create topology driven paths through the network. Each MPLS packet has a header that can contain a label. An MPLS-capable router, called also Label Switching Router (LSR), accomplishes the forwarding of the packets that it receives by examining the label and possibly another field in the header, called experimental field. At the ingress of an MPLS-capable domain the IP packets are classified and routed based on a combination of the local routing information maintained by the LSRs and of the information carried on the IP header. At this point an MPLS header is inserted for each packet. Each LSR within the MPLS-capable domain will use the label as the index to look up the forwarding table of the LSR. By using the packet forwarding procedure Label Switch Paths (LSPs) are established between pairs of nodes in the network.

The path controlling can be achieved by using traffic engineering. In this case the LSP establishment, maintenance and release is strictly controlled and managed by using signalling.

This signalling carries information related to the required characteristics for each LSP. Each node must ensure that these requirements are satisfied. An LSP can include the following requirement characteristics:

- Path Selection (based on QoS constraints);
- Delay and delay variation;
- Bandwidth, including sustained peak data rates and maximum burst sizes.

MPLS is a very promising traffic engineering mechanism and as such Diffserv can use it also for setting RSVP tunnels. But, a disadvantage is that MPLS can be used to set RSVP tunnels only locally to a single Diffserv domain, due to limitation of MPLS for usage in inter-domain environment. Actually, the MPLS can be used inter-domain also, depending on geographical scope defined, but it is unlikely that different network administrators of Diffserv domains would agree to share their facilities in such a way.

5.3 RSVP-in-IP Tunnel

Tunnelling RSVP messages in IP tunnels through Diffserv domain enables RSVP support with traffic aggregation independently of the intra-domain signalling mechanism that Diffserv uses, what is also the design goal and requirement of the Intserv/Diffserv architectural framework.

RSVP-in-IP tunnel is to be used only for carrying RSVP messages and after the proper service mapping (Intserv/Diffserv) is set the Intserv data packets are going to be treated as part of Diffserv behaviour aggregate. The RSVP-in-IP tunnel is a unidirectional tunnel, where the bi-directional functionality between the same endpoints is enabled by two unidirectional tunnels carrying traffic in the opposite directions. The endpoints of the RSVP-in-IP tunnel are either an ingress or egress of RSVP traffic, that is either an entry point or an exit point of the tunnel. The RSVP-in-IP tunnel acts as a best-effort link for RSVP messages transferred between the end points of the tunnels.

The edge devices at the boundaries between Intserv and Diffserv, that is the RID Border routers residing at the Diffserv domain are the routers that will, apart from handling both RSVP messages and Diffserv packets, perform tunnelling functions also. Thus, these routers are the end-points of the RSVP-in-IP tunnel, RID-BR-Entry or RID-BR-Exit, and they will encapsulate and decapsulate the RSVP messages in IP packets, by adding an outer IP header to the RSVP packets. Within the Diffserv domains there may be other RSVP aware routers, but they will not process tunnelled RSVP messages traversing them, since these packets are going to be encapsulated in RSVP-in-IP packets with an outer standard IP header (see Figure 5.3.2-1).

The endpoint routers of the tunnel (RID-BR-Entry and RID-BR-Exit) may be residing in the same Diffserv domain (Figure 5.3-1) or may be residing in different domains (Figure5.3-2). The RSVP-in-IP independently of whether its endpoints are residing within the same domain or different domains is foreseen to function as a pipe for RSVP messages. So the intermediate Diffserv domains or more precisely, intermediate Diffserv border routers will not decapsulate the RSVP messages, unless the destination address in the IP header matches theirs. The pipe model tunnel is depicted in (Figure 5.3-2).

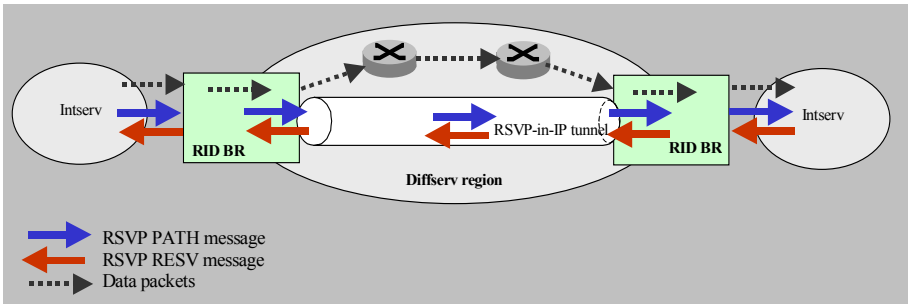


Figure 5.3-1RSVP-in-IP tunnel in a single Diffserv Domain

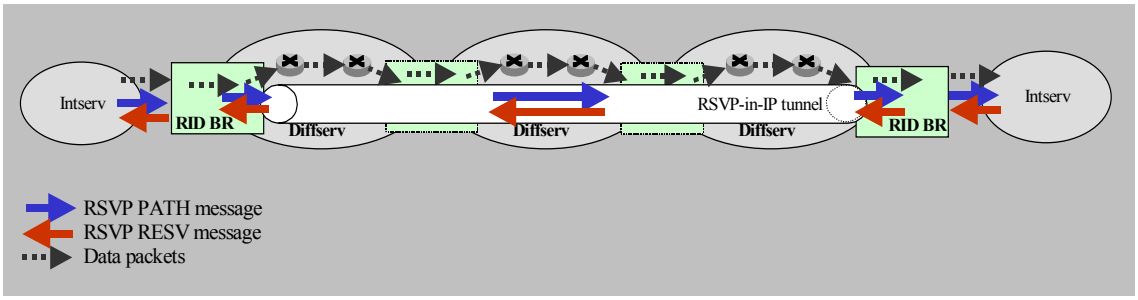


Figure 5.3-2 RSVP-in-IP tunnel in Multiple Diffserv Domain

When the endpoints of the tunnel are within the same Diffserv domain their addresses can be configured easily by the network administrator. But when they are residing within different Diffserv domains, than RID-BR-Entry doesn't have any information about the whereabouts of the RID-BR-Exit, unless there are mechanisms in place to take care of this. Since RID-BR-Entry can encapsulate the RSVP messages in IP packets only when RID-BR-Exit is known, than it is essential that RID-BR-Entry have the knowledge of the exact RID-BR-Exit router of the tunnel.

This thesis gives a proposal of a relatively simple mechanism that will avoid the occurrence of the above stated problem.

Each endpoint of the RSVP-in-IP tunnel, i.e. each RID-BR, will maintain an RSVP-in-IP Tunnel table consisting of tunnel entries (Table 5.3-1). Each RSVP-in-IP tunnel entry is identified by a Tunnel_ID, which is related to a Destination-Intserv domain address and a related RID-BR-Exit address for this specific domain.

RSVP-in-IP Tunnel Entries		
Tunnel_ID	**Destination-Intserv Domain Address**	**RID-BR-Exit address**
1	IP Address and/or AS number	IP address
2	IP Address and/or AS number	IP address
…		

Table 5.3-1 RSVP-in-IP Tunnel table

Whenever an RSVP message is coming in, RID-BR-Entry checks its RSVP-in-IP Tunnel table to find whether there is a Tunnel_ID entry for the specified destination address in the RSVP

message by performing the "longest match" on the RSVP destination address with the Destination-Intserv domain address field.

When the proper match is found, the RID-BR-Entry will encapsulate the RSVP messages in an RSVP-in-IP packet with its own address as a source address and the RID-BR-Exit address as the destination address.

Border routers between different domains will process these packets as best effort packets. They will not decapsulate these packets unless the destination address specified matches their own address. There are two possible scenarios:

- *Destination address in RSVP-in-IP packet matches the BR address*

If the destination address matches the destination address of the RID BR, than the packet has reached its destination, i.e. the exit-point of the tunnel, the RID-BR-Exit. The RID-BR-Exit decapsulates the RSVP message and forwards it upstream to the Intserv domain.

- *Destination address in RSVP-in-IP packet doesn't match the BR address nor the next-hop address*

If the destination address in the outer RSVP-in-IP header doesn't match the BR address than the RSVP-in-IP packet is not decapsulated and it is forwarded to the appropriate domain.

All the RSVP flows destined to the same Intserv addresses will be encapsulated with the RID-BR-Exit router address as a destination address.

5.3.1 RSVP-in-IP Tunnel Table

The RSVP-in-IP tunnel table looks rather trivial, but its maintenance and appropriate update of the tunnel entries indicates that knowledge of the topology of the overall Internet is a requirement, which unnecessary to say is not possible. Thus, there have to be mechanisms applied in order to avoid the situation where there will be no Tunnel Entry for the incoming RSVP messages.

The Internet is divided into Autonomous Systems or administrative domains each applying their own administrative policies and using their own protocols for intra domain routing, such as RIP or OSPF, etc. The global connectivity between AS domains is enabled by BGP (Border Gateway Protocol) protocol. BGP is used to exchange the necessary network reachability information. Which information is exchanged depends on the AS domain location in the network. For e.g. if several, say 5 AS domains share the same net prefix this information can be aggregated, such that a remote AS domain doesn't need to store all of these AS numbers, but instead it has a single entry for all 5 of them.

A single AS domain can be configured as a single Intserv or Diffserv domain, but it can also be configured such that it has both Intserv and Diffserv domains within its own AS domain. Consequently, the RSVP-in-IP tunnel entry, Destination-Intserv domain address, may next to the IP address, which is the IP subnet address of the Intserv domain within an AS, contain also the AS number.

The Intserv/Diffserv architectural framework is based on a concept of customer/provider relationship either between the Intserv domains and Diffserv domains or between multiple Diffserv domains. Adjacent domains negotiate SLA, SLS respectively with one another, which means that these domains will have the knowledge on the neighbouring domain. A definition of SLS given in [Ch00] distinguishes between SLSes with specified destination (fully or partially) for the intended destination domain of the traffic covered by the SLL, and SLSes with unspecified destinations. The SLS with specified destination is important for the domain's resource allocation, routing and the relationship with the domain's egress SLS.

For the Intserv/Diffserv architectural framework, the SLS with fully specified destination as described in [Chi00] is particularly desirable, since the Diffserv domain by means of this SLS will have the info on intended domain by customer's traffic. And since the Intserv/Diffserv architectural framework requires static SLSes, the owner of the Diffserv domain, knowing the destination domain, will acquire the appropriate address of the RID-BR-Exit, i.e. the address of the last exit routers address before the customer's traffic reaches its destination domain. This info initially should be exchanged between the network administrators of the domains. And automatization of this process by means of a protocol or other mechanism is outside the scope of this thesis and is left for future work.

To conclude, if the SLSes are negotiated and the networks are functioning accordingly, than there should always be a tunnel entry in the RSVP-in-IP tunnel table for the incoming RSVP messages.

5.3.2 RSVP in IP encapsulation

The RSVP messages will be encapsulated in IP packets and tunnelled through the Diffserv region via the RSVP-in-IP tunnel. An outer IP header will be added to the RSVP message containing the source and destination address of the tunnel endpoints, and other header fields specific for the tunnel configuration. The protocol ID will be 4 to identify the RSVP-in-IP tunnel as an IP tunnel. The RSVP-in-IP tunnel packet format is given in Figure 5.3.2-1. Through Diffserv domains the encapsulated packets will be forwarded as best-effort packets.

Optionally the router alert option can be set to indicate that these packets are signalling packets and as such they shouldn't be discarded in case of congestion in the router.

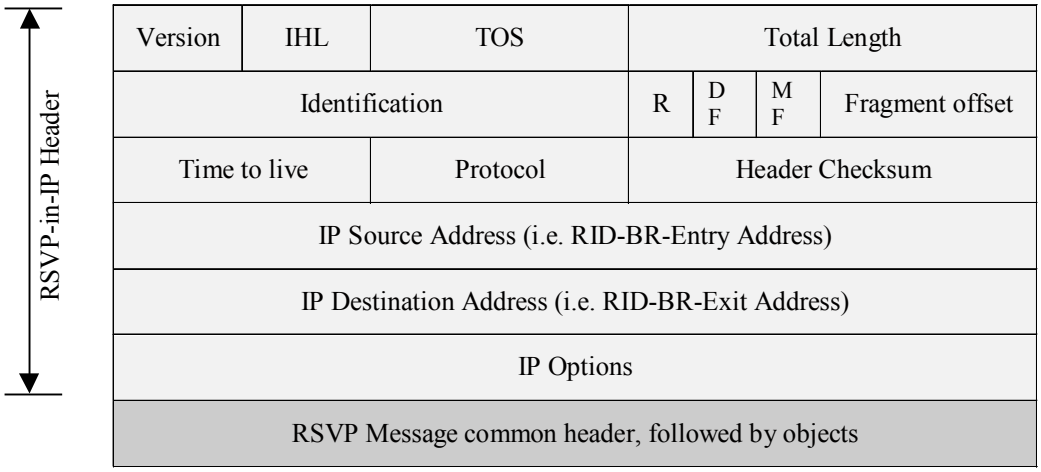


Figure 5.3.2-1 Encapsulated RSVP message

When the RSVP messages arrive at the RID Border router, there already is a Tunnel_ID in the RSVP-in-IP tunnelling table for the domain the RSVP packets are destined to. After the RID-BR-Entry has processed the RSVP PATH message coming from the Intserv domain and stored its state, than it will encapsulate it in an RSVP-in-IP packet with an outer RSVP-in-IP header as given above. That is with its own address as IP source address and the chosen RID-BR-Exit address as IP destination address. RID-BR-Exit address is chosen based on the longest match of the destination address of the RSVP message and the Destination-Intserv domain addresses in RSVP tunnel table. When the RSVP PATH message arrives at the RID-BR-Exit, it will be decapsulated, processed and then forwarded to the Intserv region. The RSVP RESV message arriving from Intserv region will be processed by the RID-BR-Entry (RID-BR-Exit of the PATH message) in the same way encapsulated in an RSVP-in-IP packet and forwarded to the Diffserv region. It will be decapsulated upon the arrival at the exit-point of the RSVP-in-IP tunnel, processed and forwarded to the Intserv region.

The rest of the RSVP message types concerning path and reservation errors (PathERR, ResvErr), path and reservation teardown (PathTEAR and ResvTEAR) and confirmation on a requested resource (ResvCONF) will be encapsulated/decapsulated in the same manner as PATH and RESV message.

The tunnel is "soft-state" and since it is an IP tunnel it is managed and maintained by means of ICMP messages [RFC1853], [RFC2003]. It will be maintained until there are RSVP message exchanges for the same pair of endpoints. Once there is no SLA in place with the Intserv customer the Tunnel_ID entry will no longer be stored in RSVP-in-IP tunnel table.

5.3.3 Broadening

The RSVP messages can also first be aggregated into a single bundle and then encapsulated and after they arrive at the destination decapsulated and deaggregated and sent to the Intserv regions. This approach while it will be more scalable it will also add up to the complexity of the RID Border Router architecture and functionality. So the RID Border Router residing at the Diffserv region will be aggregator/deaggregator and encapsulator/decapsulator at the same time. Aggregating per-flow RSVP messages before encapsulating them will be more beneficial for the processing in the intermediate routers, even if these packets are going to be practically invisible to them and since they are signalling message they will not create a burden in the core network.

Once the RSVP-in-IP tunnel is set, it can be used by Diffserv network administrator for other purposes apart from traversing RSVP signalling messages, like for sending other control messages for managing the network or for packets with special requirements or even for best-effort service packets.

Also, the described way of having a tunnel table with tunnel entries can be applied to IP tunnelling in general, whenever an network administrator of a particular domain sees a need for it, for example for tunnelling Ipv6 packets through IPv4 domains.

5.4 RSVP tunnel example

The purpose of this section is to show by means of an example the RSVP tunnels functionality. Figure 5.4-1 illustrates the Intserv/Diffserv architectural framework components. The Diffserv shown is and abstraction of multiple domains between the Intserv 1, Intserv 2 and Intserv 3. H1, H2 and H3 are RSVP aware hosts. Intserv 1 and Intserv 2 domain is identified with an IP net

address, while Intserv 3 domain is identified with an IP net address and AS number as well. The IP addresses of the components in this example are given in Table 5.4-1.

Each of the RID border routers RID-BRA, RID-BRB and RID-BRC shown in Figure 5.4-1 have their own RSVP-in-IP tunnel tables, with a list of possible tunnel entries depending on the SLSes with the customers. The tunnel entries for each of these RID-BR are shown in Table 5.4-2.

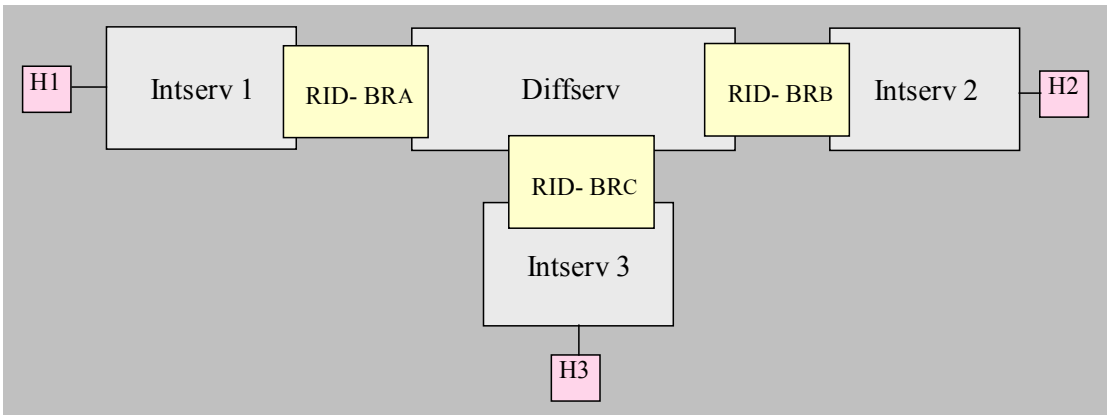


Figure 5.4 -1 RSVP Tunnel Example

Host IP address		Intserv Domain address and/or AS number	RID-BR address	
1	197.27.45.10	197.27.45/27	A	178.44.15.71
2	198.8.3.45	198.8.3/24	B	192.23.5.7
3	193.32.16.12	193.32.16/16 (1045)	C	144.21.44.14

Table 5.4-1 IP addresses for the components shown in Figure 5.4 -1

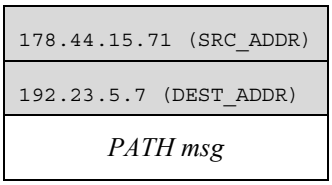
RID - BR	Tunnel_ID	Destination-Intserv Domain address	RID-BR-Exit address
A	1	198.8.3/24	192.23.5.7
	2	193.32/24 (1045)	144.21.44.14
	3	130.45.0/16	133.34.17.3
B	1	197.27.45/27	178.44.15.71
	2	125.24.6/16 (1045)	144.21.44.14
	3	132.44.55/16	123.15.17.6
C	1	197.27.45/27	178.44.15.71
	2	198.8.3/24	192.23.5.7
	3	133.14.5/16 (1701)	138.15.23.45

Table 5.4-2 RID-BR RSVP-in-IP Tunnel Entries

Host 1 (H1) in domain Intserv 1 initiates the communication with host 2 (H2) residing in domain Intserv 2. Intserv 1 and the downstream Diffserv domain have already in place the appropriate SLSes. The steps related to the RSVP tunnel functionality are explained below:

- PATH msg from H1 arrives at RID- BR A destined for H2

The RID-BR_A will check its RSVP-in-IP tunnel table for the longest match for IP destination address given in H1 IP header, i.e. 198.8.3.45. The longest match found is the Intserv 2 address - 198.8.3/24. Thus, the RID-BR_A will encapsulate the PATH message in a packet with its own address as source address and the RID-BR_B (192.23.5.7) as destination address, since this routers address is the one related to the Intserv 2 address in a tunnel entry of RID-BR_A RSVP-in-IP tunnel table. Thus it will encapsulate the PATH message in an IP packet with the header fields for src and dest address as shown below:

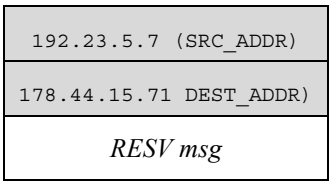


- Encapsulated PATH msg from H1 arrives at RID- BRB destined for H2

Since the IP Dest_addr is the RID_BRB router address and the Protocol ID is 4 to identify the IP tunnel packet, the RID_BRB will decapsulate this packet, store the PATH state and send it further to the Intserv 2 domain, i.e. H2.

- RESV msg from H2 arrives at RID- BRB destined for H1

After the H2 has received the PATH message, it will send a RESV message to H1. Eventually if there are enough resources in Intserv 2 this message will arrive in RID-BRB. RID-BRB will store the RESV message state and will perform the longest match on the destination address given in RESV message, which will match the Intserv 1 domain address. Thus, same as what RID-BRA did with the PATH message, it will encapsulate the RESV message in an IP packet with the header fields for src and dest address as shown below:



- Encapsulated RESV msg from H2 arrives at RID- BRA destined for H1

Since the IP Dest_addr is the RID_BRA router address and the Protocol ID is 4 to identify the IP tunnel packet, the RID_BRB will decapsulate this packet, store the RESV state and send it further to the Intserv 1 domain, i.e. H1.

[6] INTEGRATED SERVICES AND DIFFERENTIATED SERVICES IN LINUX

6.1 Introduction

Open source UNIX – like and the most popular operating system freely available, Linux is a phenomenon on the Internet. Programmers around the world are continually implementing new features of the code or improving or fixing the bugs of the current ones, contributing in this way to the Linux development and open source movement at the same time. Linux was developed by a student Linus Torvalds as a "hobby project" and until a couple of years ago it was not more than a "students" written software. These days, however, Linux is a stable and effective operating system and it represents a strong competition to other UNIX versions and especially to Windows NT in the small to medium server market [Sh99]. For many universities, research institutes and even companies it is a tool for daily computing needs.

One of Linux's main advantages is the great flexibility of its networking code. A Linux box can be used as a gateway, router, or firewall at the same time and beginning with the kernel versions 2.1.x with the introduction of traffic control Linux has support for QoS algorithms also. Consequently both IP QoS architectures Integrated Services and Differentiated Services are supported by Linux.

This chapter gives an overview of the Linux networking and traffic control, the supported QoS algorithms necessary for implementation of the Integrated Services and Differentiated Services, and also a brief description of the existing available tools serving this purpose.

Note that the resources available on Linux traffic control, i.e. QoS are limited. Therefore this chapter is based fully to the existing documents [AlSa99][Al99][Ra99] and represents the understanding of these documents and tools.

6.2 Linux Networking and Traffic control

This section gives an overview of Linux networking and also of the traffic control code, which represents the basis for QoS support under Linux.

6.2.1 Overview of Linux Networking

The Linux networking layer model is based on the 7-layer OSI model. This enables Linux to support protocols other than the standard TCP/IP, such as AppleTalk or IPX or even IPv6, although the latter is still in an experimental phase. Working in an environment of multiple

protocols is one of Linux's greatest advantages. The functionality related to higher layers, i.e. the application layer, the presentation and the session layer is handled by the user space processes running on top of the Linux kernel. The kernel itself is responsible for handling the functionality of the transport, network and data link layer, while the physical layer is outside the scope of the kernel. The division of the Linux operating system responsibilities related to the OSI-layers functionality is shown in Figure 6.2.1-1a) as given in [Sh99]. Similar to the TCP/IP 4-Layer model, the Linux networking layer model itself is divided into 4 layers, as it is depicted in Figure 6.2.1-1b):

- ***Transport Layer***

  The most common transport layer protocols are TCP and UDP. The transport layer protocols represent the intermediate layer between applications in the user space and the kernel. TCP and UDP in Linux have the same functionality as in any other operating system, i.e. TCP allows more connection oriented complex operations, while UDP provides a framework for connectionless operations.

- ***Network Layer***

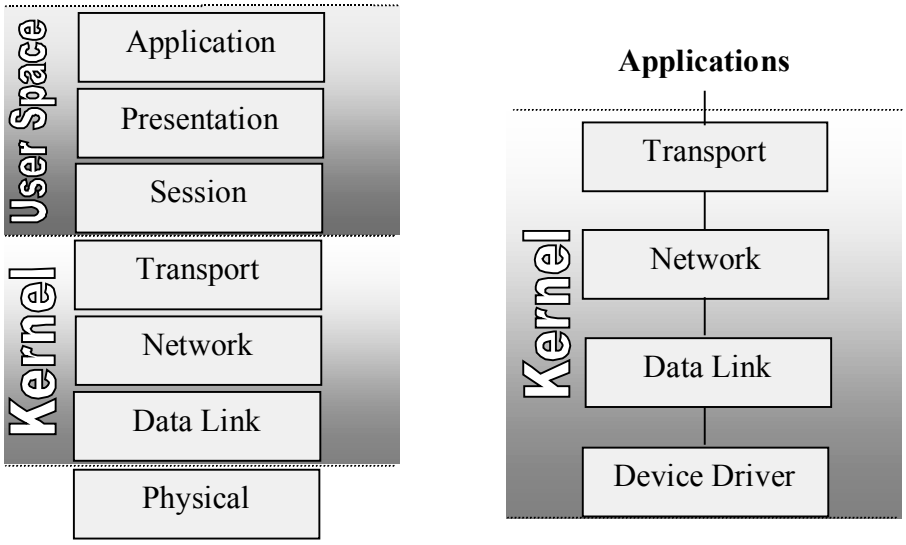
  Network layer protocols are responsible for performing routing, forwarding, fragmentation, encapsulation, setting up firewalls, etc. Thus it is the foundation for the higher layer protocols such as TCP, UDP. Although Linux can support other protocols also as mentioned earlier, the Internet Protocol (IP) is the standard network layer protocol.

- ***Data Link***

  The purpose of the data link in Linux is to perform the tasks related to the handling of the incoming and outgoing packets to the network interface. For incoming packets it determines the appropriate network layer protocol to be used and for outgoing packets it performs enqueuing to the appropriate network interface.

- ***Device driver***

  The device driver is considered as an abstraction of the interface between the kernel and the network interface cards (NIC) i.e. hardware. This layer is the software that controls a variety of different network interface cards (NIC). For each supported network interface card it creates an appropriate entry providing thus the kernel with an interface to the network interface card for handling of the incoming and outgoing packets to and from the physical medium.



a) 7-Layer OSI Model and Linux b) 4-Layer Linux Networking Model

Figure 6.2.1-1 The Network Layering Model

Generally the kernel processes will handle the packets coming from the physical layer (e.g. Ethernet link) as well as from the application layer (e.g. applications). All the incoming packets it receives from the physical layer, via the device driver-data link layer are examined at the network layer. If the Linux box is enabled to work as a router, based on the destination address of the packets it will forward the packets to another outgoing interface. Otherwise, if the packets have reached their destination, the kernel will forward them to the higher layers in the protocol stack. Also applications and transport protocols will generate packets and will require from lower layers processing such as routing or transmission. Thus, after appropriate selection of the outgoing interface the packets will be sent through the physical layer, e.g. Ethernet link.

6.2.2 Network Traffic Control in Linux

Linux is providing various traffic control functions starting from the 2.1.x kernel versions, which provide a foundation for Linux support of IP QoS mechanisms. The kernel traffic control functions and also the user-space programs to control them are implemented by Alexey Kuznetsov [Al99], one of the key developers of the networking code for Linux. Preserving the ability of Linux to function in an multi-protocol environment Alexey Kuznetsov implemented the traffic control functions, i.e. QoS support into the device data link layer (see Figure 6.2.1-1) In this way the traffic control functions will apply to the whole traffic passing through, independently of whether the network protocol is IP or IPX for example.

Figure 6.2.2-1 depicts the kernels functionality on processing of the incoming and outgoing traffic as given in [Al99]. As it is shown the traffic control functions are implemented just before the packets are sent to the device drivers. Once the appropriate interface is selected by the forwarding block (see Section 6.2.1), the traffic control will perform queuing, set up classes and filters that control the packets that are sent on the output interface.

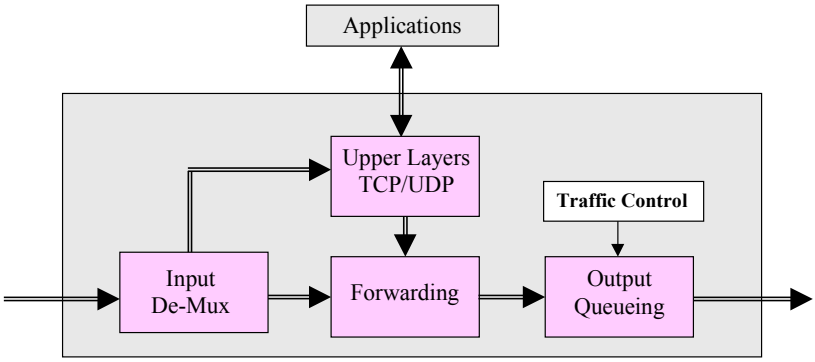


Figure 6.2.2-1 Kernel processing of incoming and outgoing network traffic

The traffic control consists of the following major building blocks (see Figure 6.2.2-2):

- ***Queuing discipline***

  The queuing discipline is the basic building block for QoS support in Linux. Each network device has a queuing discipline attached to it, which controls the treatment the enqueued packets are to receive. Currently there are several queuing disciplines supported in Linux such as: Class Based Queuing (CBQ), Stochastic Fair Queuing (SFQ), Random Early Detection (RED), Token Bucket Flow (TBF), etc.

  Simple queuing disciplines such as First In First Out (FIFO) have no classes or filters related to it, while more complicated queuing disciplines may use filters to distinguish among different classes of packets. Packets can be stored in classes based on assigned priorities; i.e. a traffic class with a higher priority will be favoured to the class with lower priority. Packets are sent out to the transmission medium according to these classes and priorities. A refined queuing discipline as given in [Al99] with multiple classes is depicted in Figure 6.2.2-2.

- ***Classes***

  Queuing discipline and classes are closely coupled. Classes and their semantics represent key properties of a queuing discipline. Classes itself do not store packets, instead they use another queuing discipline for such a purpose. These queuing discipline can be one of the above-mentioned queuing disciplines, a simple one with no classes or a more refined one contain classes as well. Usually each class owns a queue, but a queue can be also shared by several classes, depending on the queuing discipline used (see Figure 6.2.2-2). The packets themselves do not carry any class-related information.

- ***Filters (or Classifiers)***

  As depicted in Figure 6.2.2-2 filters are used to classify the enqueued packets into classes. They use different packet properties such as source address, destination address, port numbers, and TOS byte in the IP header, etc as classification parameters. There are several types of classifiers implemented in Linux: fw – based on the firewall marking of the packets, u32 – based on packets header fields, route – based on the IP destination address, rsvp/rsvp6 – a more refined classifiers using the five tuples RSVP.

- ***Policing***

  The purpose of policing is to ensure that the traffic does not exceed the certain bounds, e.g. delay or bandwidth bounds. Policing is a function which can be implemented together with

filtering / classification (see Figure 6.3-2, 6.4-1). Filter will classify only packets matching to the classification criteria. Policing is implemented either as a classification criteria of a filter, or as criteria for enqueuing packets into a class, or dropping packets within an inner class.

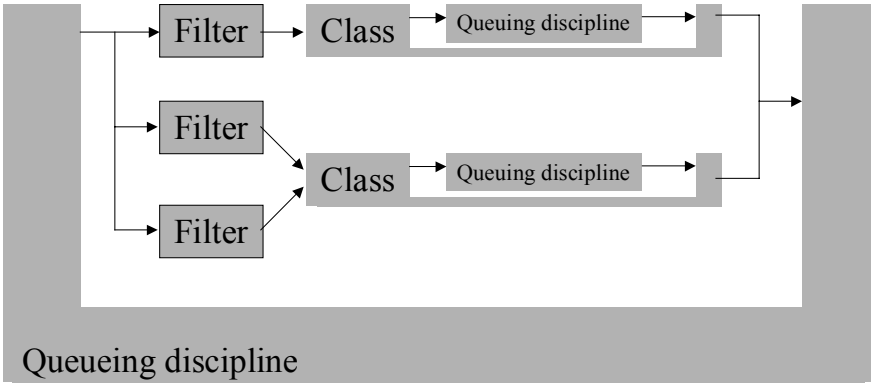


Figure 6.2.2-2 Queuing discipline with multiple classes

For building and configuring the traffic control, Alexey Kuznetsov provided also a user space tc tool [Sof5], which enables communication with the kernel. The tc tool and the kernel communicate via the netlink socket. The tc tool is implemented in such a flexible way that one can, depending on the requirements, create different scripts to build specific traffic control on the desired network interface (see Appendix A).

6.3 RSVP and Integrated Services in Linux

RSVP is a signalling protocol that does not transport any application data, rather it can be characterised as an Internet control protocol such as ICMP or IGMP. However, in Linux RSVP is implemented on the user space on top of the Kernel. The well-known implementation of RSVP is the ISI RSVP daemon, which is free software available from [Sof1].

This RSVP daemon in general provides three types of interfaces: application programming interface (the API), traffic-control kernel interface and a routing interface, by means of which the RSVP daemon communicates with the applications, the routing core and also the traffic control in the kernel to establish the reservations.

The exact functionality of the RSVP daemon will depend on whether it is running on a host or on a router. Its role in both cases is depicted in Figure 6.3-1. The RSVP daemon on a host will have to communicate with the applications while it may not require any traffic control related functionality, i.e. host may rely on overporvisioning of its outgoing link. The RSVP daemon running on a router will have to, communicate with the kernel via its traffic control interface, in order to establish reservations for Intserv style flows.

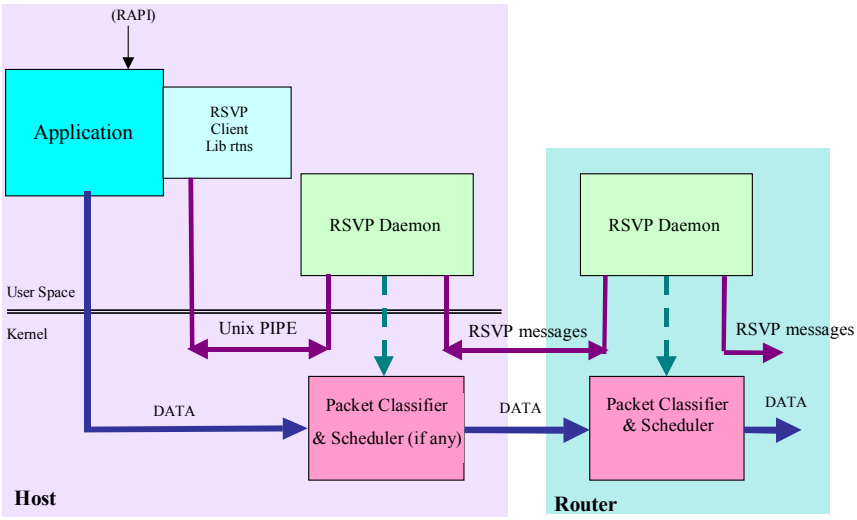


Figure 6.3–1 RSVP daemon role in the host and router

As already said by means of traffic control Linux has support for the Integrated Services architecture. Figure 6.3–2 depicts the relation of the Intserv node parameters to traffic control building blocks in Linux. Thus the responsibilities of Classifier and Policer are implemented by means of filters. Afore mentioned filters such as fw and u32 are able to perform policing also. The packet scheduler functions in Linux are to be implemented as queuing discipline functions. The queuing discipline, i.e. the scheduler is the one that joins everything together in Linux. The queuing discipline used for Intserv support is Class Based Queuing (CBQ).

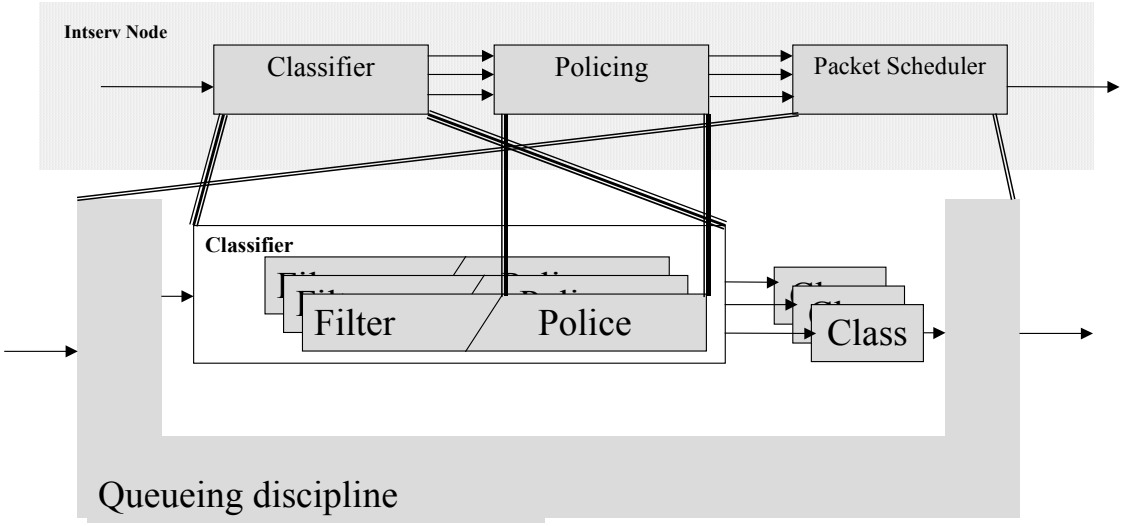


Figure 6.3-2 Relationship between the Linux traffic control and Integrated Services building blocks

6.3.1 Class Based Queuing (CBQ)

CBQ is a link sharing mechanism [FlJa95] that provides the possibility to protocol families or traffic types to share the traffic that can be supported by a network link.

A main characteristic of the CBQ mechanism is its hierarchical design. An example of the CBQ's hierarchical design is shown in Figure 6.3.1-1. The maximum available bandwidth of the network

link is assigned on the top of the hierarchy, i.e. the root. Each root's child node defines a class that requests a certain amount of the total available bandwidth, e.g. RSVP class in Figure 6.3.1-1. Subsequently, each node's child subdivides the bandwidth available to their parent. For instance, in Figure 6.3.1-1 the "CL class" node may require 65% of the bandwidth assigned to its parent, "RSVP class" in this case.

Another characteristic of the CBQ mechanism is its possibility of sharing the unutilised network link bandwidth in a predefined way with streams of traffic that need it. This is performed by giving the opportunity to a class that needs more bandwidth than initially assigned to it, to utilise bandwidth from the leaf "brother" classes (classes with the same parent) that are not using all the bandwidth allocated to them, i.e. they are underlimit. The leaf "brother" classes may start utilising bandwidth from its parent's brother class as long as the latter is not using all the bandwidth assigned to it, i.e. it is underlimit. This process is known as "borrowing".

The assignment of the bandwidth to classes can be done dynamically by using an estimator. In Figure 6.3.1-1 the "GS Class" represents such a class.

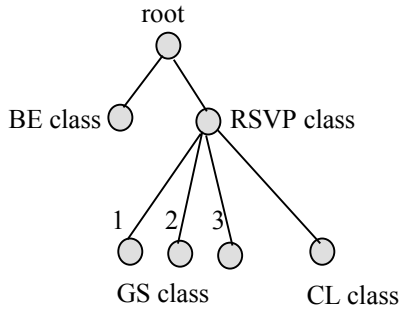


Figure 6.3.1-1 Intserv CBQ classes

6.3.2 RSVP daemon under Linux

The ISI RSVP daemon release rel4.2a4 defines a traffic control interface for communication with the kernel in order to establish reservation, as already said. However, the ISI RSVP daemon does not support any admission control functionality. Alexey Kuznetsov has ported the ISI RSVP daemon to Linux and has implemented the admission control functionality, creating in this way full support for Integrated Services architecture. Release rel4.2a4-1 of the ISI RSVP daemon supports the Linux based traffic control specific objects (see Figure 6.3.2-1) In order to communicate with the traffic control of the Linux kernel.

Figure 6.3.2-1 depicts the RSVP daemon building blocks and how they are connected together as given in [AlGi00].

The RSVP daemon release comes together with the RTAP, which is an real time application program used as an interactive RSVP test application for exchanging Intserv manner flows but also for setting debugging and logging levels of the RSVP daemon. It communicates with it via the RTAP socket. Besides RTAP the RSVP daemon can also communicate with other applications via the RSVP API, that is via the API socket. The processing of signalling messages PATH/RESV is shown as two separate blocks Send/Receive PATH and Send/Receive RESV. The communication with the Linux kernel is handled by the RSVP socket that is in fact a routing interface to the network layer processing of RSVP messages. By means of the Netlink socket the RSVP daemon communicates with the kernel traffic control.

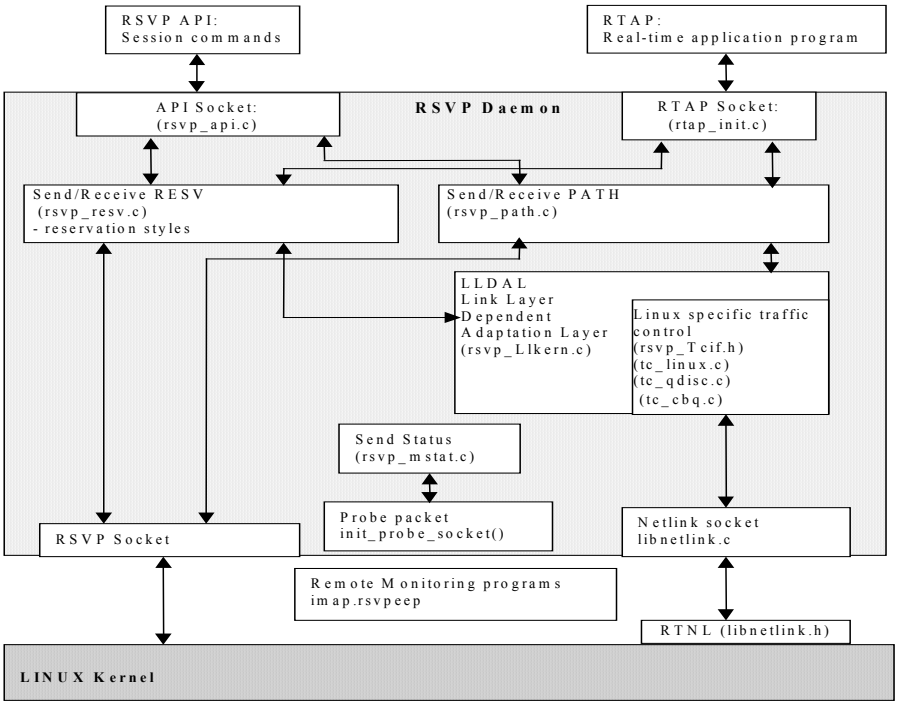


Figure 6.3.2-1 RSVP daemon functionality and building blocks

Before running the daemon and exchanging RSVP messages, first the kernel should be enabled to perform the admission control on the Intserv flows. The appropriate queuing disciplines classes and filters should be added to the network interfaces.

As mentioned CBQ is the one used as a queuing discipline on the outgoing Intserv interface. Using the tc tool [Sof5], a shell script (see Appendix A) is applied to the outgoing network interface of the network element (router and sometimes hosts), such that it has the CBQ scheduler enabled. The total amount of the link bandwidth is shared between the BE effort traffic class and RSVP traffic class. The RSVP class is further separated into CL class and for each new GS flow accepted a new GS class is added (see Figure 6.3.1-1 flows 1, 2, 3).

After setting up the scheduler and classifier, the RSVP daemon will handle RSVP signalling messages and establish reservations. The RSVP daemon does not support data transmission, thus any other application for generating traffic can be used for this purpose. The data transmission, for which the reservation was established, will be handled by the traffic control blocks already configured on the outgoing interfaces.

6.4 Differentiated Services under Linux

The support for Differentiated Services under Linux [AlSa99] is a design and implementation extension of the already existing traffic control code. The implementation supports classification based on DS field and configuration of the EF and AF PHB as defined in the Diffserv working group (WG). The relation of the Differentiated Services node functionality blocks and traffic control building blocks is depicted in Figure 6.4-1. The Classifier is implemented by means of filter while the meters functions are performed also by filters hat are having policing abilities (e.g. fw, u32, rsvp). The marking, shaping and dropping functions and consequently the PHBs are implemented as part of queuing disciplines.

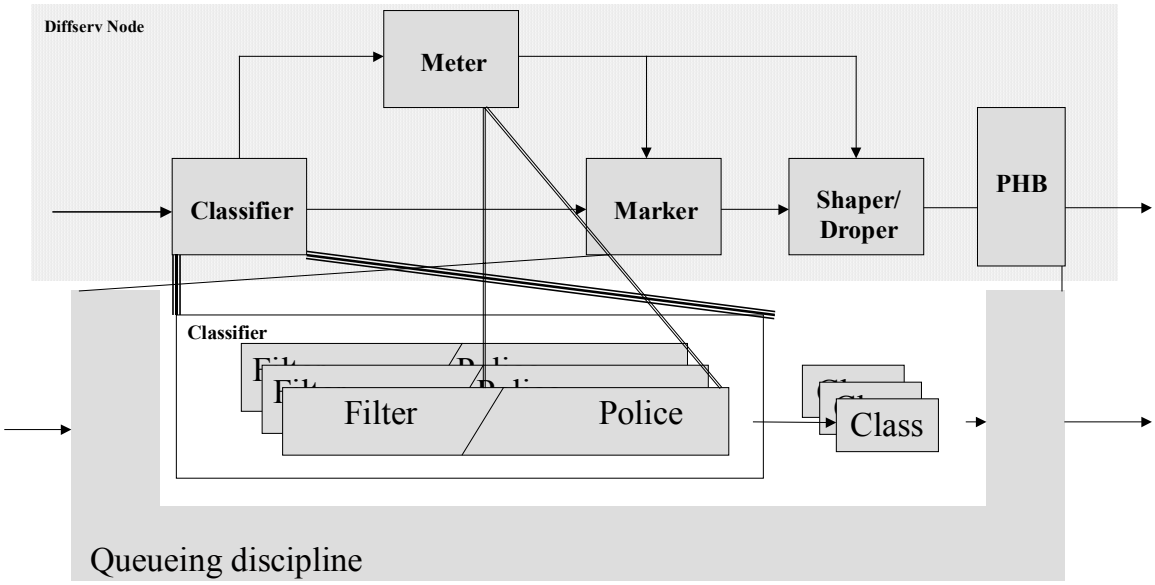


Figure 6.4-1Relationship between the Linux traffic control and Diffserv node building blocks.

For the purpose of extending the traffic control in Linux with Diffserv support, there were three new traffic control elements added:

- the queuing discipline DSMark (sch_dsmark [AlSa99]) responsible for the retrieval of the DSCP
- classifier which uses this information for aggregated classification TCindex (cls_tcindex [AlSa99])
- new queuing discipline Generalised RED (GRED) in order to support different drop priority levels for AF PHB.

The newly added components are shown in Figure 6.4-2

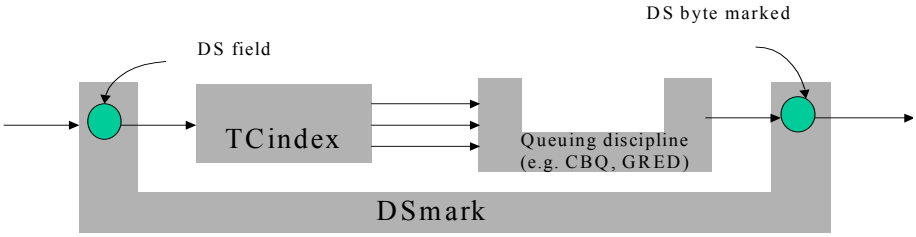


Figure 6.4-2 Description of the DSmark

6.4.1 DSmark and TCindex

The DSMark queuing discipline (see Figure 6.4-2) is specific to Diffserv, while the other ones can be used also for other purposes.

The sch_dsmark is responsible for the following actions on the enqueued packets:

- the extraction and the storage of the DSCP into the storage buffer field "skb->tc_index". skb->tc_index is a new field in packet buffer descriptor sruct sk_buff, (see [AlSa00]). Note that the storage buffer skb-tc_index relates a predefined class ID to a DSCP. This relation between class ID and DSCP is also stored into the skb->tc_index.

- the identification of the class that belongs to this DSCP
- remarking and changing the DSCP of the enqueud packets.

The Tcindex classifier (see Figure 6.4-2) uses the DSCP information stored into the skb->tc_index to select a class. It calculates a handle (or key) that is used to inquire and retrieve from the skb->tcindex the class ID of the corresponding class that satisfies the stored DSCP.

6.4.2 GRED

The other new component the Generalised Random Early Detection (GRED) Queuing discipline has been implemented in order to extend the Random Early Detection (RED) [FlJa93]], to provide drop probabilities necessary for implementing AF PHB drop precedence.

GRED provides the possibility to use multiple RED queues, i.e., create more than one RED queues within a physical queue (see Figure 6.4.2-1.). Each RED queue, called Virtual Queue, operates as a standard RED queue. Each Virtual Queue is selected based on predefined configuration parameters passed on by the user and stored in the storage buffer skb->tcindex. A user can configure preferential treatment of virtual queues by setting for example the physical queue limit as parameter or the user can assign preferences by means of the priorities. For more information see [the Internet draft: Differentiated Services on Linux: the http site].

The usage of GRED with the TCindex is depicted in Figure 6.4.2-1.

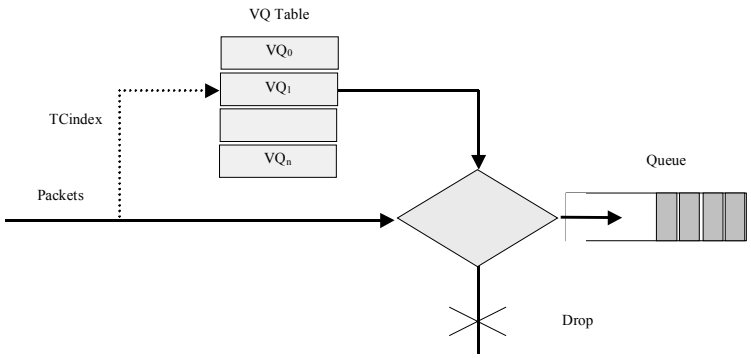


Figure 6.4.2-1 Generalized RED and TCindex

6.4.3 Diffserv Forwarding Path Structure

Based on what is said above, the general structure of the Diffserv node forwarding path (see Figure 6.4-1) in Linux, can be presented as in Figure 6.4.3-1[AlSa99]. Marking thus can be done in several places depending on how the PHBs are implemented. Independently, if the configuration is correct once the packet is leaving the queuing discipline it will always be marked with an appropriate DSCP (see Figure 6.4.3-1)

EF-PHB and AF-PHB can be implemented by combining the newly added components with other traffic control components by means of tc tool (shell/perl) scripts (see Appendix A). There is a certain amount of the examples scripts provided by the authors [Sof2].

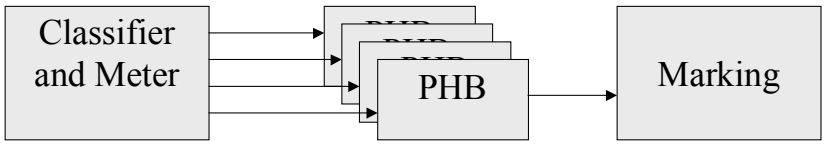


Figure 6.4.3-1 Diffserv node-forwarding path

[7] IMPLEMENTATION ENVIRONMENT, DESIGN AND REPORT ON TESTING

7.1 Introduction

The design of the RSVP/Intserv – Diffserv (RID) border router, supporting the Integrated Services and Differentiated Services architecture is given in detail in Chapter [4]. Based on this design a basic prototype of the RID router mechanisms has been developed in Linux using the QoS mechanisms and tools already described in Chapter [6].

Linux is the operating system chosen for testbed setup, first because of the Wireless Multimedia research (WMR) department Testbed configuration and secondly because of the flexibility of its networking and traffic control code described in chapter [6].

The initial implementation design is fully based on the architecture design of the RID border router. However, due to the problems encountered while testing the existing software packages [Sof1], [Sof2], there were some implementation design decisions made in order to have a basic working prototype of the RID border router. This has resulted in a deviation of the working prototype from the desired initial implementation design.

This chapter documents the implementation design and the above mentioned implementation design decisions. Further it describes the lab environment setup. It also reports testing results of the basic working prototype.

The open issues related to the prototype implementation of the RID border router concepts are depicted in this chapter as well. Finally, these open issues led to the ideas for future work, which are also part of this chapter. .

7.2 RID Border Router Implementation Design

Compatible with the RID Border router design (see Figure 4.2.2-1), the implementation prototype of the RID Border Router will have to support the necessary functions related to the RSVP/Intserv and Diffserv interoperability.

In terms of the Linux QoS tools described in Chapter [6], this means that the RID border router prototype implementation will use the RSVP daemon for signalling RSVP messages and the traffic control, i.e. QoS support of the kernel, for Intserv traffic control and the extension of the latter (see section 6.4) for Diffserv PHB implementation. Figure 7.2-1 depicts the concept of the RID border router prototype implementation design. Recalling here the RID border router conceptual model given in Figure 4.2.1-2 and comparing it with the Figure 7.2 –1 one can see

that the RSVP message handler functionality is to be performed by the RSVP daemon, while the DS packet handler functionality is to be performed by the Intserv/Diffserv Linux traffic control functionality. To avoid the confusion the Intserv traffic control is related to data traffic and not RSVP messages and since the RID border router's data plane is Diffserv, this traffic control functions can be considered as part of the DS packet handler, while in fact both Intserv and Diffserv traffic control functions are part of the Linux kernel. Accordingly the DS packet handler can be associated to be the Linux kernel traffic control.

Thus there is the RSVP daemon running on top of the kernel, which will at the same time support both Intserv and Diffserv.

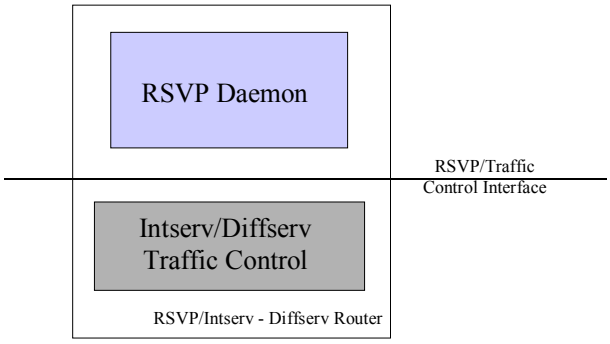


Figure 7.2 -1 RID Border Router Implementation Design Concept

The knowledge gathered during the studying and testing of the tools described in Chapter [6] and taking into consideration the global architectural design (see Chapter 3) and RID border router design led to the following implementation design choices for the prototype implementation of the RID border router:

- On the interface towards the Intserv side the RID router performs the same functions as a standard RSVP/Intserv router, thus the RSVP daemon establishes reservations and the Intserv classifier and scheduler is set up for traffic control mechanisms
- On the interface towards the Diffserv side, the RID router will encapsulate the RSVP messages into UDP packets and route them to the endpoint of the RSVP tunnel, i.e. the RID-BR-Exit (see Table 5.4 -1)
- On the interface towards the Diffserv side the RID router will have to set the appropriate queuing disciplines and classifiers for remarking of the packets conforming to its static SLS with the customers.
- The RSVP daemon will handle the Intserv / Diffserv interoperability functions, such that upon receiving of the RESV message from the Diffserv side (see Figure 3.4-1) it will install a MultiField classifier, which will classify data packets coming from Intserv side into appropriate classes towards the Diffserv side
- The Intserv / Diffserv Service Mapping is implemented at the kernel level, by dynamically setting up the MultiField classifier, i.e. the classifier will be installed upon acceptance of an RESV message and removed upon deletion of a flow (upon receiving of an RESV-TEAR message). Note that in the design of the RID router (see Figure 4.2.2 –1) the Service Mapping element is part of the RSVP messages handler. The actual implementation of the service mapping is done in the kernel, but than it is triggered by the RSVP daemon. Thus,

the Service Mapping element should not be confused with the actual service mapping implementation.

- The data traffic belonging to this particular flow will be classified and marked with the appropriate PHB and conforming to SLS.

7.2.1 The Core of the Implementation Design – the RSVP daemon

Based on above said the RSVP daemon will be the core module for building an implementation prototype of the RID border router under Linux. As such it requires some changes of its traffic control interface functions:

1. The admission control of the RSVP/Intserv flows should be done conforming to the agreed SLS for the entire Diffserv domain.
2. This admission control function (changes to TC_AddFlowSpec) should install the appropriate MultiField classifier on the appropriate outgoing interface, once there is a RESV message coming from the Diffserv side. The MultiField classifier should be set according to the retrieved RESV Flowspec parameters, such that it will classify data packets to the appropriate PHB
3. Upon release of the resources the installed MultiField classifier should be removed (changes to the TC_DelFlowSpec function)
4. The disable function of the RSVP daemon should be implemented in order to disable the daemon on specific interfaces facing Diffserv, because otherwise the RSVP daemon traffic control module will produce an traffic control error if the interface has DSMark queuing discipline instead of appropriate CBQ used for RSVP/Intserv.

7.2.1.1 The Encountered RSVP daemon problems

While testing the RSVP daemon release rel4.2a4-1 will full Intserv support there were some problems encountered, which led to the limitation of its usage with full Intserv capability.

The problem relates to some runtime errors the RSVP daemon was producing (see Appendix B for these errors) after an RESV message was received and before performing admission control. In order to continue with further implementation the decision was made to disable the Linux traffic control support, i.e. Intserv admission control and to use the RSVP daemon only for setting up RSVP sessions and exchanging of RSVP messages without establishing any reservations. Therefore, further in this chapter, the traffic control interface refers to the "dummy" interface (tc_test.c), thus there is no Intserv style admission control function performed.

Because of this the disable function will not be necessary, i.e. the daemon will not check for traffic control functions (classifier and scheduler) on the interfaces.

The only changes made from the ones listed above are changes 2 and 3.

Considering the above and as it will be explained later this prototype implementation is in fact an implementation of the RID border router concept, rather than an implementation of the whole RID functionality.

7.2.1.2 Service Mapping Implementation

The Intserv to Diffserv Service mapping has to be implemented according to the Section 4.2.2.1. This means that based on RESV Flowspec parameters an appropriate classifier should be set as mentioned in 7.2.1.

However, due to the problems with the RSVP daemon the service mapping could not have been implemented as desired. Without the admission control on the RSVP/Intserv flows first there is no knowledge whatsoever about the available resources and second the processing of the Flowspec parameters necessary to implement service mapping would have been implemented as part of the Linux specific traffic control files (see Figure 6.3.2-1), which were not used.

Therefore, in order to have at least a basic working prototype of the RID border router the following was done:

- The Diffserv domain administrator will configure the RID beforehand appropriately to conform with the agreed SLS.
- This administrator will also configure the MultiField classifier based on the static SLS that the Diffserv has negotiated with its Intserv customer. The MultiField classifier will filter the packets based on IP header fields, i.e. the IP source address, IP destination address and destination port number. The classifier used for this purpose is u32. What this means is that the Intserv customer flows will be classified based on their IP header fields.
- The MultiField Classifier is installed dynamically upon "acceptance" of the RESV message and deleted upon "release" of the resources, i.e. upon receiving of RESV Tear message.
- This was implemented simply just by making a system call to the executable tc scripts written for this purpose. The system calls are made in the rsvp_llkern.c file (see Figure 6.3.2-1) as shown below:

```
#ifdef SCHEDULE
         handle = TC_AddFlowspec(kp->tcs_OIf, rp->rs_spec,
                        &Path_Te, adspecp, TC_kflags, &Fwd_specp);
         kp->tcs_rhandle = handle;
         system("/local/QoS/scripts/correct/filterbuffy.eth1");
#endif
```

and

```
#ifdef SCHEDULE
         TC_DelFlowspec(kp->tcs_OIf, kp->tcs_rhandle);
         system("/local/QoS/scripts/correct/delbuffy.eth1");

#endif
```

- Further, since RTAP was used for initiating RSVP sessions between the hosts (see Section 7.3) there was no possibility to create several RSVP session at the same time for e.g. between the same pair of hosts on different port numbers, since with RTAP there is can be only one RSVP session at a time. And because of the specific lab setup (see Figure 7.3-1), there was no possibility to have multiple hosts connected to the same router.

Thus, what can be derived from above is that this completed implementation of the RID router prototype is very basic. It is to be seen only as the implementation of the RID router concept and not as a real RID router implementation.

7.3 The RSVP/Intserv and Diffserv Test Bed

Initially for testing of the tools described in chapter [6] and later for implementing and testing the RID router the RSVP/Intserv and Diffserv testbed was setup as depicted in Figure 7.3-1.

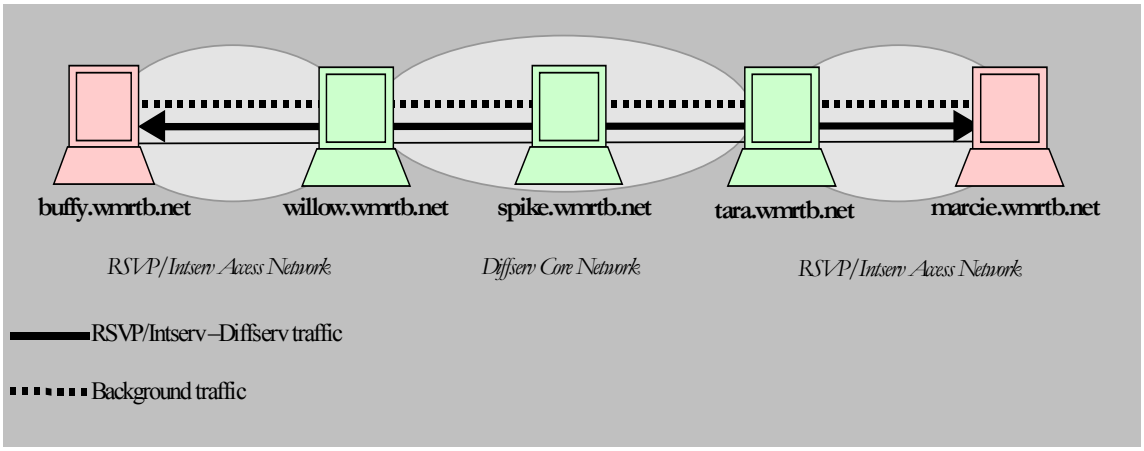


Figure 7.3 –1 RSVP/Intserv and Diffserv WMR Test Bed

All five PC are running Linux Operating System, Mandrake 7.0 and 7.1 respectively, kernel versions 2.2.14 and 2.2.16. Besides Buffy all the machines are Diffserv enabled (patched with ds-8 [Sof2]) and apart from spike all the machines are RSVP enabled, i.e. the RSVP daemon is compiled and running. RTAP is used for generating RSVP/Intserv flows and a ttcp [Sof4] for generating UDP traffic. There is also a simple generator used for generating background traffic. The configuration of the PC is as hosts and routers:

- Buffy and Marcie are configured as RSVP aware hosts, which initially use RTAP for creating, modifying or releasing RSVP sessions. They belong to two different subnets that are configured to play the role of RSVP/Intserv access networks.
- Willow and Tara are configured as RID border routers and are running the modified version of the RSVP daemon in order to handle the Intserv / Diffserv interoperability. They have one of the interfaces configured as Intserv and the interfaces towards Diffserv are configured to handle traffic aggregates. The Diffserv will perform policing, shaping, dropping and remarking on incoming packets according to the agreed SLSes and in this way it will protect the Diffserv domain. Appropriate behaviour is configured by means of the tc shell scripts, which are installed before the RSVP daemon initialisation. These scripts are given in Appendix A. Further, both Willow and Tara will encapsulate RSVP packets into UDP on the interfaces facing the Diffserv side. For this purpose the RSVP daemon configuration file will have the udpencap function next to the interface.
- Spike is configured as Diffserv core router, such that it checks the DS field of the incoming packets in order to retrieve the DSCP byte and assign them to the appropriate class. The packets with no DSCP set are to be assigned to the best effort class. So the encapsulated RSVP packets will be treated by spike as best effort. This tc script implementing the core router is given in Appendix A.

7.3.1.1 Static SLS between Intserv and Diffserv

Initially the service mapping of the Controlled Load to AF PHB is to be performed. In the testbed, the Diffserv is configured to support two AF classes (AF1 and AF2) with three-drop priority levels. This is implemented by means of the configuration scripts given in Appendix A. The Diffserv domain in the test bed can support maximum 10Mbit bandwidth.

This bandwidth is divided between the Best Effort and AF classes. The support for AF1, AF2 and BE is implemented by means of CBQ, while AF1 and AF2 support is implemented by means of GRED with assigning GRIO [ClWr97], [AlSa99] priority levels to the VQ (see Appendix A).

On the Intserv side the CBQ (as described in 6.3.1) is configured in order to handle RSVP/Intserv traffic although there is no traffic control interface modules in the RSVP daemon to use it. This bandwidth is shared between the best effort class and the reserved class.

In the agreed SLS the amount of bandwidth for the reserved class and the amount of bandwidth given to the AF class should be the same. Each is assigned 5Mbit/s. Further the AF class bandwidth should be divided between the different classes. The AF1 class has 3 Mbit/s assigned and a higher priority and AF2 has 2Mbit/s assigned and a lower priority. The customer (e.g. Buffy) in this SLS requires different drop priorities for the traffic generated to the same destination address (e.g. Marcie) on different port numbers. This is depicted in Table 7.2.1-1. Note that the TSpec parameters are measured as given in Section 2.2.2.

Reserved Class bandwidth	AF class bandwidth		CL TSpec parameters and destination port numbers	
5 Mbit/s	AF 1.1 [0x0a]	3 Mbit/s	[CL: 1280 256 64 1280 128]	5000
	AF 1.2 [0xoc]		[CL: 640 256 64 1280 128]	5001
	AF 1.3 [0x0e]		[CL: 512 256 64 1280 128]	5002
	AF 2.1 [0x12]	2 Mbit/s	[CL: 384 256 64 1280 128]	5003
	AF 2.2 [0x14]		[CL: 256 256 64 1280 128]	5004
	AF 2.3 [0x16]		[CL: 128 256 64 1280 128]	5005

Table 7.2.1-2 Static SLS parameters negotiated between Intserv and Diffserv

The association of the controlled load flows (CL) to the AF PHB is based on Section 4.2.2.1.1 and also on the type of applications to be used. In Section 4.2.2.1.1 it is said that the packets with larger token rate b/r , i.e. highly bursty flows will experience more than the less bursty flows.

The real-time applications, such as e.g. VoIP do not generate bursty traffic, while the other applications such as for e.g. web browsing are highly bursty applications. The real-time applications in terms of AF PHB will be assigned the highest level class, e.g. AF 1.1. Therefore the non-bursty Controlled Load flows with the smallest b/r ratio such as [CL: 1280 256 64 1280 128] will be mapped to the AF1.1 And the high bursty flows such as [CL: 128 256 64 1280 128] will be mapped to AF 2.3.

7.4 Testing scenario and results

The above test bed was initially designed with a particular testing scenario in mind, that is the scenario given in section 3.4.2. This scenario describes the sequence of events happening from the moment when the QoS aware application at one host initiates the RSVP process until the moment when this host receives the recognition of the granted resources and starts sending data. Thus, only a unidirectional data transmission is tested, where one host is the sender and the other is the receiver. In order to explain the testing scenario and show the results showing the appropriate RID behaviour, this same example is repeated relating to the test bed above. Note that in the Intserv/Diffserv testbed there is no Edge Router.

Buffy and Marcie are the hosts initiating RSVP, where Buffy is the sender and Marcie is the receiver. In this case it is Willow that will have the functionality of the RID router.

1. Buffy and Marcie initiate an RSVP session via RTAP, using UDP and port number 5000:

   ```
   T1> dest udp marcie/5000
   ```

2. Buffy is a sender and it generates an RSVP PATH Message

   ```
   T1> send buffy/5000 [t 1280 256 1280 64 128]

   ------------------------------------------------------------------------
   ---

   PATH: Sess: buffy.wmrtb.net/5000[17] R: 30000   PHOP: <(API) LIH=0>
   marcie.wmrtb.net/5000

   T=[1.28K(256) 1.28KB/s 64 128] Adspec( 0 hop InfBW 0us 65535B, G={br!},
   CL={br!})

   Snd Raw  PATH: Sess: buffy.wmrtb.net/5000[17]    R: 30000   PHOP: <(API)
   LIH=0> marcie.wmrtb.net/5000  T=[1.28K(256) 1.28KB/s 64 128] Adspec( 0
   hop InfBW 0us 65535B, G={br!}, CL={br!})
   ------------------------------------------------------------------------
   ------
   ```

   where the TSpec parameters are 1280 (KB/s) - the token bucket rate r, 256(KB) – token bucket size b, 1280 (Kb/s) –peak rate, 64B –minimum policy unit m, 128B – maximum packet size M

3. Willow processes the PATH, encapsulates it on a UDP packet and tunnels it through Diffserv to Tara. The statistics on Willow traffic control on both interfaces are:

   ```
   [root@willow correct]# ./stats.eth0
   --------- qdisc parameters ---------------------
   qdisc sfq 8002: quantum 1514b limit 128p flows 128/1024 perturb 15sec
          Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
   qdisc sfq 8001: quantum 1514b limit 128p flows 128/1024 perturb 15sec
          Sent 377664 bytes 309 pkts (dropped 0, overlimits 0)
   qdisc cbq 1: rate 10Mbit cell 8b mpu 64b (bounded,isolated) prio no-
   transmit/8
          weight 10Mbit allot 1514b
          level 3 ewma 5 avpkt 1000b maxidle 23us
          Sent 377664 bytes 309 pkts (dropped 0, overlimits 0)
          borrowed 0 overactions 0 avgidle 624 undertime 0
       ---- Class parameters----------
   class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
   class cbq 1:1 parent 1: rate 10Mbit prio no-transmit
   class cbq 1:2 parent 1:1 leaf 8001: rate 5Mbit prio 6
   ```

```
class cbq 1:3 parent 1:1 leaf 8002: rate 1Mbit prio 2
   class cbq 1:7fff parent 1:7ffe rate 1Mbit prio 6

   class cbq 1:7ffe parent 1:1 rate 5Mbit prio no-transmit

------------------------------------------------------------------------
------
[root@willow correct]# ./stats.eth1
---- qdisc parameters Egress  ----------
qdisc red 8005: limit 60Kb min 15Kb max 45Kb ewma 3 Plog 17 Scell_log 9
   Sent 603 bytes 7 pkts (dropped 0, overlimits 0)

 qdisc gred 8004:
 DP:1 (prio 2) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 21 Scell_log 9
 DP:2 (prio 3) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 20 Scell_log 9
 DP:3 (prio 4) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 19 Scell_log 5
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc gred 8003:
 DP:1 (prio 2) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 21 Scell_log 9
 DP:2 (prio 3) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 20 Scell_log 9
 DP:3 (prio 4) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 19 Scell_log 9
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

qdisc cbq 2: rate 10Mbit cell 8b mpu 64b (bounded,isolated) prio no-
            transmit/8
        weight 10Mbit allot 1514b
        level 1 ewma 5 avpkt 1000b maxidle 23us
   Sent 603 bytes 7 pkts (dropped 0, overlimits 0)
   borrowed 0 overactions 0 avgidle 605 undertime 0

qdisc dsmark 1: indices 0x0040 set_tc_index
       Sent 603 bytes 7 pkts (dropped 0, overlimits 0)

 ---- Class parameters Egress  ----------
class cbq 2: root leaf 8005: rate 10Mbit (bounded,isolated) prio no-
transmit
class cbq 2:1 parent 2: leaf 8003: rate 3Mbit prio 4
class cbq 2:2 parent 2: leaf 8004: rate 2Mbit prio 5
class cbq 2:6 parent 2: rate 5Mbit (bounded,isolated) prio 7
class dsmark 1:1 parent 1: mask 0x03 value 0x0a
class dsmark 1:2 parent 1: mask 0x03 value 0x0c
class dsmark 1:3 parent 1: mask 0x03 value 0x0e
class dsmark 1:4 parent 1: mask 0x03 value 0x12
class dsmark 1:5 parent 1: mask 0x03 value 0x14
class dsmark 1:6 parent 1: mask 0x03 value 0x16
class dsmark 1:7 parent 1: mask 0x03 value 0x00
```

```
---- filter parameters Egress ----------
```

4. Spike as a core router treats these packets as best – effort
5. Tara receives the PATH message, processes it, decapsulates it and sends it to Marcie
6. Marcie receives a PATH message:

```
------------------------------------------------------------------------
---
Rcv Raw PATH marcie.wmrtb.net/5000[17] eth0<=0 <tara2.97.169.195/64
PATH: Sess: marcie.wmrtb.net/5000[17]    R: 30000   PHOP:
<tara2.97.169.195.in-addr.arpa LIH=1> buffy.wmrtb.net/5000  T=[1.28K(256)
1.28KB/s 64 128]
------------------------------------------------------------------------
---
```

7. Upon receiving of the PATH message, Marcie generates a RESV message:

```
T1> reserve marcie ff  buffy/5000 [cl 1280 256 1280 64 128]

------------------------------------------------------------------------
---
Snd Raw  RESV      marcie.wmrtb.net/5000[17] 0=>eth0 >
tara2.97.169.195/63
  RESV: Sess: marcie.wmrtb.net/5000[17]    R: 30000   NHOP:
<marcie.wmrtb.net LIH=1>
    FF   buffy.wmrtb.net/5000    [CL T=[1.28K(256) 1.28KB/s 64 128] ]
------------------------------------------------------------------------
---
```

7. Tara receives the RESV message, encapsulates it and forwards it to Spike
8. Spike carries these packets as best effort and sends it to Willow
9. Willow upon receiving of the RESV message and once this message is accepted (and in this implementation it is always accepted) installs, by making a system call, a MultiField classifier for this flow. This classifier will match the packets coming from Buffy (ip src 195.169.97.178) going to Marcie (ip dst 195.169.97.210) on port number 5000 (ip dport 0x1388). Classid 1:1 is defined in the script in Appendix A. Thus matched packets will be marked with the AF 1 DP 1 [0x0a] DSCP byte:

```
------------------------------------------------------------------------
---
$TC filter add dev eth1 parent 1:0 protocol ip prio 4 handle 1: u32
divisor 1
# AF Class 1 DP 1
$TC filter add dev eth1 parent 1:0 prio 4 u32 match ip src 195.169.97.178
\
match ip dst 195.169.97.210 match ip dport 0x1388 0xffff classid 1:1
------------------------------------------------------------------------
---
```

After the admitted RESV message, in order to see whether the filter is installed, the statistics on Willow's interface are run again:

```
---- filter parameters Egress ----------
filter protocol ip pref 4 u32
filter protocol ip pref 4 u32 fh 1: ht divisor 1
filter protocol ip pref 4 u32 fh 800: ht divisor 1
filter protocol ip pref 4 u32 fh 800::800 order 2048 key ht 800 bkt 0
flowid 1:1
  match c3a961b2/ffffffff at 12
  match c3a961d2/ffffffff at 16
  match 00001388/0000ffff at 20
```

```
    ------------------------------------------------------------------
    ---
```

10. Buffy receives the RESV message.

```
------------------------------------------------------------------
---
Rcv Raw  RESV buffy.wmrtb.net/5000[17] eth0<=0 < buffy.wmrtb.net/61
  RESV: Sess: buffy.wmrtb.net/5000[17]    R: 30000   NHOP:
<buffy.wmrtb.net LIH=0>
    FF   marcie.wmrtb.net/5000    [CL T=[1.28K(256) 1.28KB/s 64 128] ]
------------------------------------------------------------------
---
```

11. Upon receiving RESV message Buffy assumes that there are enough resources available and it will start sending UDP data to Marcie on port number 5000. For this purpose ttcp [Sof4] was used:

    The command used to initiate Buffy as a sender, i.e. in transmission mode:

```
./ttcp –t –u –p 5000 –s –l 64 –n 100000 Marcie
```

12. On the other hand marcie is also running ttcp in a receiver mode : program

```
./ttcp –r –u –p 5000
```

13. The tcpdump tool [Sof3] was used to check whether the received packets are marked with the appropriate DSCP byte. The output of the tcpdump tool was:

```
------------------------------------------------------------------
--
tcpdump: listening on eth0
18:04:22.554265 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp 4 [tos
0x0a]
18:04:22.554572 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp 64 [tos
0x0a]
18:04:22.554930 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp 64 [tos
0x0a]
18:04:22.555174 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp 64 [tos
0x0a]
------------------------------------------------------------------
---
```

    Note that [tos 0x0a] represents AF1.1 (see Table 7.2.1-2).

14. Afterwards the resources were released by Marcie

```
------------------------------------------------------------------
-
Snd Raw  RESV-TEAR marcie.wmrtb.net/5000[17] 0=>eth0 >
tara2.97.169.195/63
RTEAR: Sess: marcie.wmrtb.net/5000[17]    NHOP: <marcie.wmrtb.net
LIH=1> FF   buffy.wmrtb.net/5000    [ ]
------------------------------------------------------------------
--
```

15. Upon receiving of the RESV-Tear message Willow will delete the MultiField classifier:

```
------------------------------------------------------------------
-
# AF Class 1 DP 1
```

```
$TC filter del dev eth1 parent 1:0 prio 4 u32 match ip src
195.169.97.178 \
match ip dst 195.169.97.210 match ip dport 0x1388 0xffff classid
1:1
------------------------------------------------------------------
--
```

The statistics that were run afterwards on Willow show that the filter is not present any more.

16. And if there is traffic again generated in the same way as before by means of the ttcp, the tcpdump on Marcie shows that the arriving packets are just treated as best effort packets by the in between routers in between:

```
------------------------------------------------------------------
--
18:05:38.375853 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp
4
18:05:38.376136 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp
64
18:05:38.376347 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp
64
18:05:38.376544 buffy.wmrtb.net.1034 > marcie.wmrtb.net.5000: udp
64
------------------------------------------------------------------
--
```

In order to test whether the marked packets are receiving the preferential treatment, in the core router, Spike, some statistics were run by means of the tc tool. The tc tool is continuously counting the number of packets in each leaf class, starting from the moment that the tc scripts are initiated. In other words the measurements are continuously cumulative.

From the statistics from below it can be depicted that AF1 queue, is represented by leaf class number: 8004 and the Best Effort queue, represented by leaf class number: 8006.

These measurements have been accomplished in two subsequent phases. Note that in each of these phases the same amount of traffic is sent from the sender Buffy to the receiver Marcie:

- First Phase

During the first step, the RSVP session is still ongoing. This enables the RID router - Willow, to mark AF1 packets. The core router - Spike, is then treating these packets in a right way by enqueuing the AF1 traffic into the AF1 queue and the Best effort traffic into the Best effort queue. This can be seen by looking into the below results. The packets marked as AF1 packets are enqueued in the leaf class 8004. In particular, 3680 AF1 packets are counted. While the Best Effort are enqueued in the leaf class 8006. In particular, 39 BE packets were sent.

```
 [root@spike tc]# ./tc -s -d qdisc show dev eth1
qdisc red 8006: limit 60Kb min 15Kb max 45Kb ewma 3 Plog 17 Scell_log 9
       Sent 9604 bytes 39 pkts (dropped 0, overlimits 0)
qdisc gred 8005:
       Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
qdisc gred 8004:
       Sent 389720 bytes 3680 pkts (dropped 0, overlimits 0)
qdisc cbq 2: rate 10Mbit cell 8b mpu 64b (bounded,isolated) prio no-
       transmit/8 weight 10Mbit allot 1514b
```

```
       level 1 ewma 5 avpkt 1000b maxidle 23us
       Sent 399450 bytes 3722 pkts (dropped 0, overlimits 0)
       borrowed 2479 overactions 0 avgidle 31 undertime 0
qdisc dsmark 1: indices 0x0040 set_tc_index
       Sent 399450 bytes 3722 pkts (dropped 0, overlimits 0)

[root@spike tc]# ./tc -s class ls dev eth1
class cbq 2: root rate 10Mbit (bounded,isolated) prio no-transmit
       Sent 399450 bytes 3722 pkts pkts (dropped 0, overlimits 0)
       borrowed 2479 overactions 0 avgidle 624 undertime 0
class cbq 2:1 parent 2: leaf 8004: rate 3Mbit prio 4
       Sent 389720 bytes 3680 pkts (dropped 0, overlimits 0)
       borrowed 2479 overactions 0 avgidle -4644 undertime -1.93682e+09
class cbq 2:2 parent 2: leaf 8005: rate 2Mbit prio 5
       Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
       borrowed 0 overactions 0 avgidle 75826 undertime 0
class cbq 2:6 parent 2: leaf 8006: rate 5Mbit (bounded,isolated) prio 7
       Sent 9604 bytes 39 pkts (dropped 75, overlimits 1605)
       borrowed 0 overactions 17 avgidle 18956 undertime 0
--------------------------------------------------------------------------------
--
[root@spike tc]# ./tc -s filter ls dev eth1
filter parent 1: protocol ip pref 1 tcindex hash 64 mask 0xffff shift 0
fall_through
filter parent 1: protocol ip pref 1 tcindex handle 0x0000 classid 1:161
filter parent 1: protocol ip pref 1 tcindex handle 0x000a classid 1:111
filter parent 1: protocol ip pref 1 tcindex handle 0x000c classid 1:112
filter parent 1: protocol ip pref 1 tcindex handle 0x000e classid 1:113
filter parent 1: protocol ip pref 1 tcindex handle 0x0012 classid 1:121
filter parent 1: protocol ip pref 1 tcindex handle 0x0014 classid 1:122
filter parent 1: protocol ip pref 1 tcindex handle 0x0016 classid 1:123
filter parent 1: protocol ip pref 4 tcindex hash 0 mask 0x00fc shift 2
pass_on
--------------------------------------------------------------------------------
---
```

- Second Phase

During the second phase, the RSVP session is released. This disables the RID router - Willow, to mark AF1 and Best Effort packets. Willow, and the core router Spike treat all packets as Best Effort packets. This can be observed in the following results:

```
[root@spike tc]# ./tc -s -d qdisc show dev eth1
qdisc red 8006: limit 60Kb min 15Kb max 45Kb ewma 3 Plog 17 Scell_log 9
       Sent 133359 bytes 1192 pkts (dropped 75, overlimits 75)
qdisc gred 8005:
       Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
qdisc gred 8004:
       Sent 389720 bytes 3680 pkts (dropped 0, overlimits 0)
qdisc cbq 2: rate 10Mbit cell 8b mpu 64b (bounded,isolated) prio no-
       Sent 523289 bytes 4877 pkts (dropped 75, overlimits 1588)
       borrowed 2479 overactions 0 avgidle 624 undertime 0
qdisc dsmark 1: indices 0x0040 set_tc_index
       Sent 523289 bytes 4877 pkts (dropped 75, overlimits 0)

[root@spike tc]# ./tc -s class ls dev eth1
class cbq 2: root rate 10Mbit (bounded,isolated) prio no-transmit
       Sent 390518 bytes 3699 pkts (dropped 0, overlimits 0)
       borrowed 2479 overactions 0 avgidle 624 undertime 0
class cbq 2:1 parent 2: leaf 8004: rate 3Mbit prio 4
       Sent 389720 bytes 3680 pkts (dropped 0, overlimits 0)
```

```
       borrowed 2479 overactions 0 avgidle -4644 undertime -1.93682e+09
class cbq 2:2 parent 2: leaf 8005: rate 2Mbit prio 5
       Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
       borrowed 0 overactions 0 avgidle 75826 undertime 0
class cbq 2:6 parent 2: leaf 8006: rate 5Mbit (bounded,isolated) prio 7
       Sent 133359 bytes 1192 (dropped 75, overlimits 1605)
       borrowed 0 overactions 17 avgidle 18956 undertime 0
```

During the second phase the number of the AF1 packets that were counted during the first phase did not change, i.e., 3680 AF1 packets. The number of the BE packets is during the second phase increased from 39 packets to 1192 packets.

Note, that EF PHB would have been implemented in the same way as AF PHB and according to Section 4.2.2.1.2

7.5 Test result analysis and future work

The above tests show the correct behaviour of the RID basic prototype implementation. But as it is said this implementation represents the implementation of the concept and not the real RID router functionality.

In order to have a real prototype implementation, with full capabilities of the RID router, such that it will support full interoperability of the Integrated and Differentiated Services, regardless of the static or dynamic resource provisioning first would require a correct admission control behaviour in the Intserv manner. That is the problems with the RSVP daemon rel4.2a4 –1 should be solved. Next the TC_AddFlowSpec function will have to be modified in order to enable service mapping related functionality. At the same time the disable function should be implemented.

All these steps would result in a full RID prototype implementation at least in the case of static SLSes and it would provide a good foundation for further work in enabling dynamic SLSes as well, e.g. support for a protocol to communicate with the Bandwidth Broker (BB). This would be part of the future work activities.

Furthermore, by using performance experiments, the behaviour of the implemented end to end Intserv/Diffserv QoS management mechanisms under congestion situations will be studied. What would also be very desirable are performance tests in order to check the scalability of this router and also to see by means of performance values how well the Intserv Services are mapped to Diffserv PHB and also the relation between these services among themselves.

[8] WIRELESS INTERNET QOS

8.1 Introduction

Enabling QoS in Internet is difficult, and it becomes even tougher when one is introducing QoS in an environment of mobile hosts, wireless networks, and different access technologies, because of wireless networks dynamically changing topologies and resources. Yet, the need for QoS mechanisms in this environment is greater due to scarce resources, unpredictable available bandwidth and variable error rates.

The rapid growth of mobile systems indicates that the future Internet will have to deal with mobile users that will use the same diversity of applications as fixed users. Thus, solutions for enabling QoS over IP should take into account mobility issues also, in order to be able to fulfil these upcoming requirements of future Internet users.

However, the current work on the QoS over IP architectures, i.e. Integrated Services and Differentiated Services seems to leave out mobility support, despite its importance. Therefore, in this chapter a framework for QoS and Mobility in the Internet is introduced. The framework presented integrates various QoS architectures and mobility protocols and will offer the freedom to users to choose between different wireless and wired access technologies based on certain predefined criteria, e.g. QoS parameters. The QoS and mobility framework architecture is initially intended to be a flexible and open architecture suitable to be applied for a large variety of applications with different QoS demands, different access technologies, i.e. wireless and wired, and protocols.

This chapter refers to the framework presented in [KaRe00], [ReKa00] and shows by an exemplification how the Intserv/Diffserv architectural framework described in Chapter 3 fits into the wireless environment, related to the framework. But firstly, an overview of IP mobility and session protocols is given, and also a proposal for the interoperation between the RSVP and these protocols. A general introduction of the framework, its entities and protocols is also given. There are also several QoS mobility service classes identified and described.

8.2 Session Protocols

Framework for QoS and Mobility besides for negotiation, managing and controlling of sessions, foresees the usage of session protocols also as means of session layer negotiation of QoS parameters. Different QoS parameters can be negotiated via the session protocols, e.g., the bandwidth, peak rate, etc. The following section gives a short description of the Session Initiation Protocol (SIP) as the most appropriate for usage in the framework proposed.

The Session Initiation protocol (SIP) [RFC2543] is an application layer control (signalling) protocol for creating, modifying, and terminating sessions between multiple participants. SIP

uses among others two gateway proxies. One of them is called SIP proxy server and the other one is called SIP redirect server. The messages used in the SIP protocol are listed in Table 8.2-1.

SIP Messages	
SIP Message Name	**SIP Message Function**
INVITE	Invitation of a user to a session. The message body contains the session description, e.g. using SDP. SDP contains: session name and purpose, time(s) the session is active, the media comprising the session, information to receive those media (addresses, ports, formats and so on) and it may contain additional information about the bandwidth to be used by the conference [RFC2327].
ACK	A confirmation to any response. It is also the only message that doesn't trigger a response upon reception.
BYE	A user is leaving the session.
OPTIONS	Queries a server about its capabilities
CANCEL	Cancel a pending request
REGISTER	Register with a SIP server

Table 8.2-1The SIP messages

The SIP proxy server is used to locate the destination party of an invitation to a session and it remains in the signalling path for the duration of the session setup. Figure 8.2-1 views the SIP session setup when the SIP proxy server is used. Upon the receipt of an INVITE request message the SIP proxy server will query its location database to find out the location of the destination (called) terminal. All the subsequent signalling is routed via the SIP proxy server. The user data is transported through the transport network directly between the sending (calling) and receiving (called) terminals.

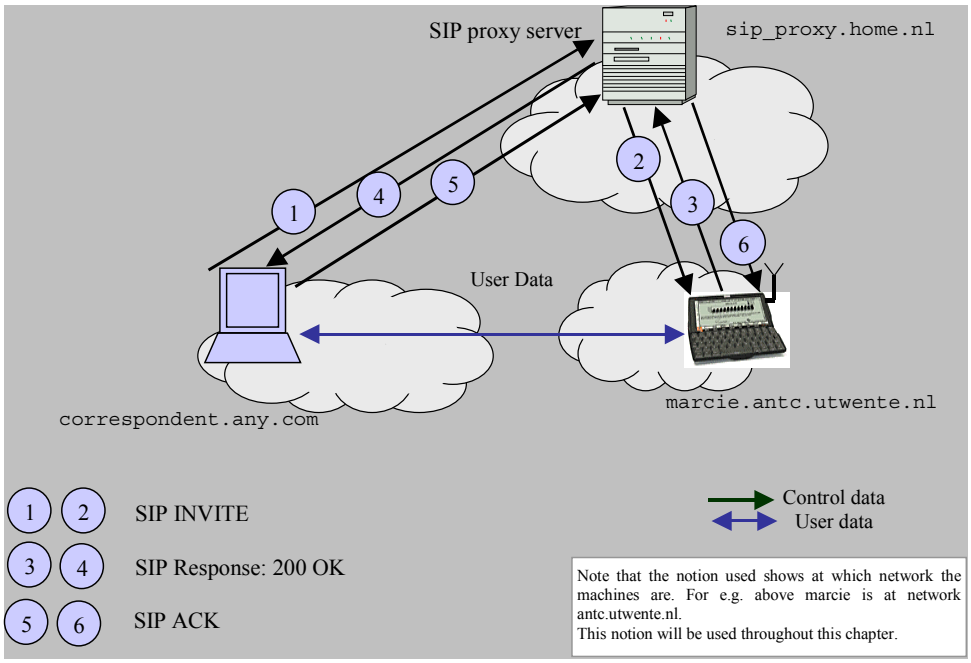


Figure 8.2-1 SIP session setup using the SIP proxy server

Another proxy server used by the SIP protocol is the SIP redirect server. This server (Figure 8.2-2) is used to locate the called party to a session but it does not remain in the signalling path for the duration of the session setup. After finding out the location of the called party it sends this information back to the calling party. Afterwards, the user data is transported through the transport network directly between the sending (calling) and receiving (called) terminals.

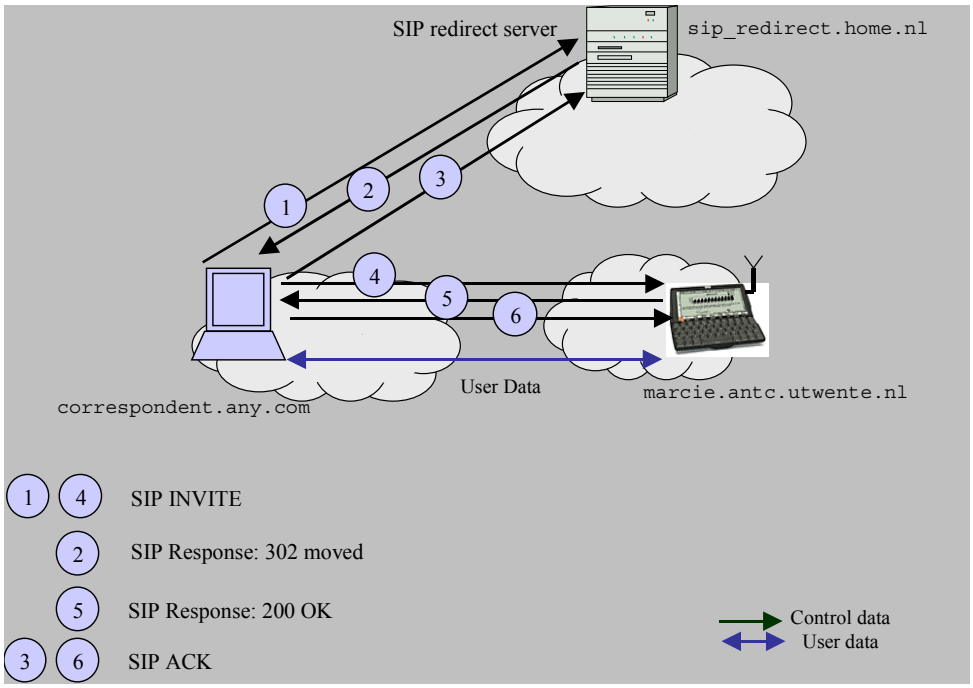


Figure 8.2-2 SIP session setup using the SIP redirect server

8.3 Protocols for Mobility Support

Enabling mobile devices seamless communication and access to the Internet via their wireless network interfaces, independent of their roaming in other networks, requires efficient protocols that will be able to inform the network about the changes in their network attachments. In the following sections Mobile IP is explained and also SIP for mobility support.

8.3.1 Mobile IP

The Mobile IP protocol is the most common protocol for providing mobility support at the IP layer, transparently to the layers on top, e.g. TCP. The key feature of the Mobile IP [RFC2002] design is that all required functionality for processing and managing mobility information is embedded in well-defined entities, the Home Agent (HA), Foreign Agent (FA), and Mobile Node (MN). The Mobile IP protocol allows the MNs to retain their IP address regardless of their point of attachment to the network. This can be fulfilled by allowing the MN to use two IP addresses, the home address which is static and is mainly used to identify higher layer connections, e.g. TCP, and the Care-of Address, which has to identify the mobile's new point of attachment with respect to the network topology. In Mobile IPv4 the Foreign Agent manages the Care-of Address. Mobile IP functionality is realised by using three mechanisms (for detailed descriptions of these mechanisms see [Per98] and [Per97]):

- Discovering the Care-of Address (Figure 8.3.1-1):

  The Care-of address discovery procedure used in Mobile IP is based on the ICMP (Internet Control Message Protocol) Router Advertisement standard protocol, specified in RFC 1256 [RFC1256]. In Mobile IPv4, the router advertisements are extended to also contain the required Care-of Address. These extended router advertisements are known as agent advertisements. Home Agents and Foreign Agents typically broadcast at regular intervals (e.g., once a second, or once every few seconds) and in a random fashion, agent advertisements.

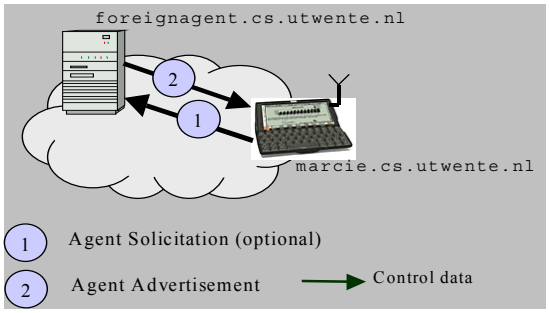


Figure 8.3.1-1: Care of Address Discovery

- Registering the Care-of Address (Figure 8.3.1-2):

  After the Mobile Node gets the Care-of Address it will have to inform the Home Agent about it. In Mobile IP this can be accomplished by using the registration procedure. The Mobile Node sends a registration request (using the User Datagram Protocol (UDP)) with the Care-of Address information. This information is received by the Home Agent and normally, if the request is approved it adds the necessary information to its routing table and sends a registration reply back to the Mobile Node.

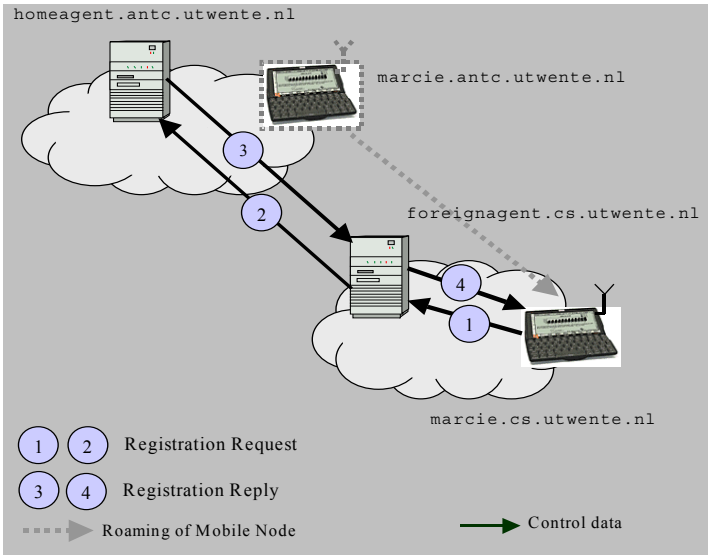


Figure 8.3.1-2: Registering the Care-of Address

- Tunnelling to the Care-of Address (Figure 8.3.1-3):

  Is accomplished by using encapsulation mechanisms. All mobility agents, i.e., Home Agents and Foreign Agents, using Mobile IPv4 must be able to use a default encapsulation

mechanism included in the IP within IP protocol [RFC2003]. By using this protocol, the source of the tunnel, i.e., Home Agent, inserts an IP tunnel header, in front of the header of any original IP packet addressed to the Mobile Node's home address. The destination of this tunnel is the Mobile Node's Care-of Address. In IP within IP [RFC2003] there is a way to indicate that the next protocol header is again an IP header. This is accomplished by indicating, in the tunnel header, that the higher-level protocol number is '4'. The entire original IP header is preserved as the first part of the payload of the packet. By eliminating the tunnel header the original packet can be recovered.

When the Mobile IP packet flow, follows a route similar to the one seen in Figure 8.3.1-3, then the routing situation is typically called triangle routing. The packets sent by the host, called correspondent host (CH), follow the path 1,2 and 3, while the packets sent by the Mobile Node follow routes 4 and 5.

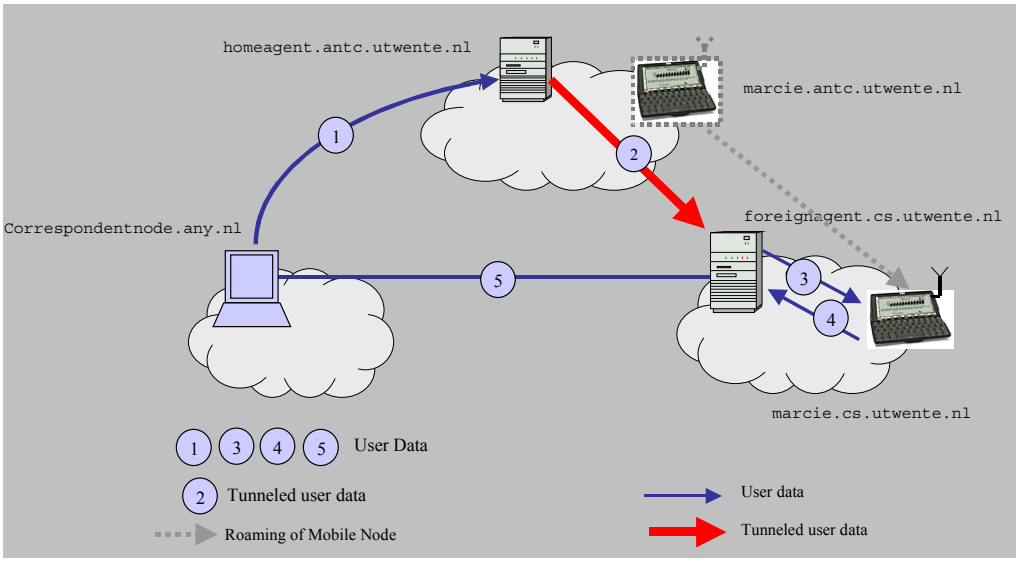


Figure 8.3.1-3: Tunneling to the Care-of Address

8.3.1.1 Triangle routing and route optimization

In [PeJo00] (see e.g., [Per97], [Per98] and [Kar99]), the operation of the base Mobile IP protocol is extended to allow for more efficient routing procedures, such that IP packets can be routed from a correspondent host to a Mobile Node without going to the Home Agent first.

These extensions are referred to as route optimisation (Figure 8.3.1-4), wherein new methods for IP nodes, e.g., correspondent hosts, are provided. The correspondent host receives a binding update message from the mobile node's Home Agent that contains the Mobile Node's Care-of Address. The binding specifies the association of the home address of a Mobile Node with a care-of address for that Mobile Node, along with the remaining lifetime of that association. This binding is then stored by the correspondent host in a binding cache and is used to tunnel its own IP packets directly to the care-of address, bypassing the Mobile Node's Home Agent. In this way, the triangular routing situation, explained in Section 8.3.1 is eliminated. However, in the initiation phase, the IP packets sent by the correspondent host still use the triangle routing until the moment that the binding update message sent by the Mobile Node's Home Agent, is received by the correspondent host.

In addition to the Binding Update message, the route optimisation procedure is using the following messages:

- A *binding warning* control message is usually sent by a node (e.g., Mobile Node or Correspondent Host), to the Home Agent (i.e., recipient), indicating that a Correspondent Host (i.e., target) seems to be unaware of the Mobile Node's new Care-of Address;
- A *binding request* message is send by a Correspondent Host to the Home Agent at the moment it determines that its binding should be initiated or refreshed. Note that if the home agent for a certain reason (e.g. the Mobile Node is in its home domain), can not find or does not want to inform the correspondent host about the MN's Care_of Address, then the Home Agent will send a Binding Update message also to the CH. However, this message will include a Care_of Address that is set equal to the MN's home address and the association lifetime is set to zero. The CH will then have to delete the binding cache entry for that particular MN.
- A *binding acknowledgement* message can be requested by a Mobile Node from a Correspondent Host that has had received the *binding update* message.

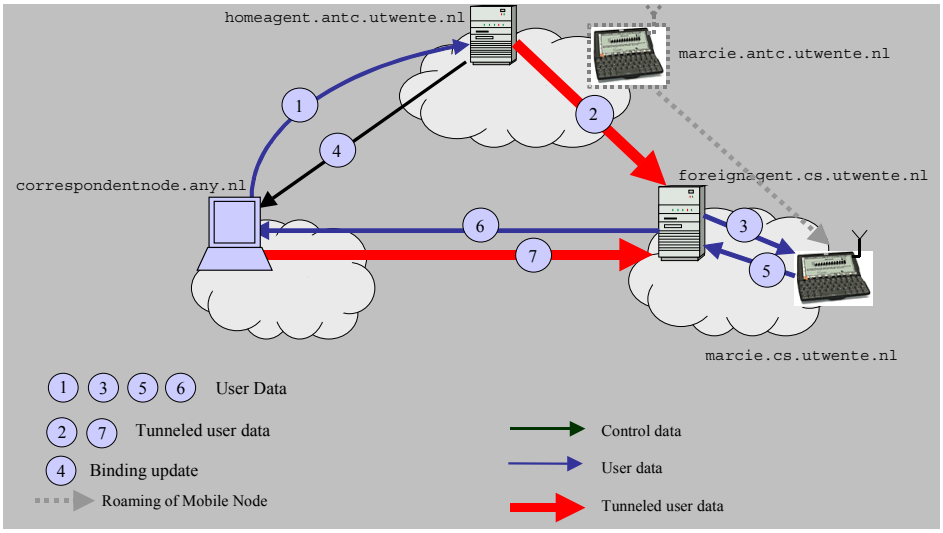


Figure 8.3.1-4: Route optimization in Mobile IP

8.3.2 SIP and mobility support

Unlike Mobile IP, the mobility support using Session Initiation Protocol (SIP) [WeSc99] proposes a mechanism in handling mobility at the higher layer, that is the application (i.e. session) layer whenever that is applicable.

Similar to Mobile IP, the mobile host has a home network that is managed by a physical entity called SIP redirect server. The SIP redirect server as explained in Section 8.2, similar to the HA in Mobile IP, is capable of storing information regarding the location of a mobile host. Every time that a mobile host roams into a new IP sub-network it will inform the SIP redirect server about its new IP address that it will register (Figure 8.3.2-1). When a correspondent host wishes to communicate with a Mobile Host, it will send an invite message to the SIP redirect server. The SIP redirect server will send the IP address of the Mobile Host to the Correspondent host. If the mobile host is moving during the session, it sends an invite message to the correspondent host to inform it about its new IP address. The correspondent host will use this IP address to send all the subsequent IP user data traffic to the Mobile Host.

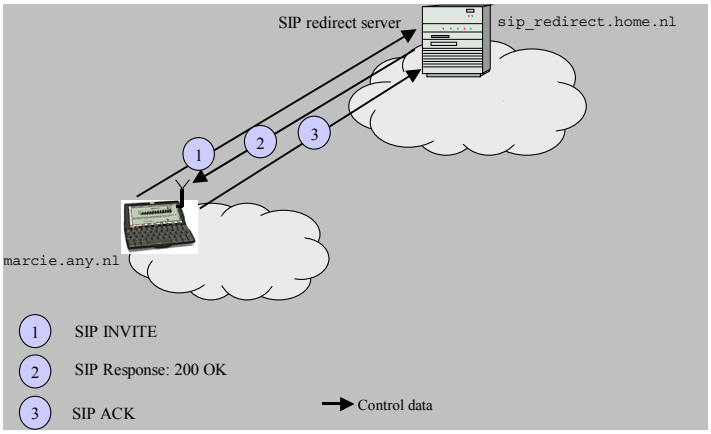


Figure 8.3.2-1: IP address registration at the SIP redirect server

The operation of SIP mobility support is shown in Figure 8.3.2-2 and the explanation of the SIP messages is given in Table 8.2-1. When the mobile host (e.g.marcie.home.nl) is moving from its home network to another location (e.g. marcie.foreign1.nl or marcie.foreign2.nl) then it always registers its new location with the SIP redirect server, similar to home agent registration in Mobile IP (Figure 8.3.2-2 msg. 9,10). When the correspondent host (e.g. correspondent.any.nl) which might be another mobile host or a fixed host begins a session, it sends an INVITE to the redirect server, which will then redirect this message to current location of the mobile node (see Figure 8.3.2-2 msg. 1-5). When the mobile host is moving during the ongoing session then it must send a new INVITE to the correspondent node (Figure 8.3.2-2 msg. 6-8). In the Contact field of the SIP message it inserts a new IP address, which the correspondent node will indicate in the new SDP field as a transport address for redirecting the data flow.

The advantages of mobility support using SIP instead of Mobile IP are: there will be no need for tunnelling data packets, easily applicable to most common applications (that are not using the TCP protocol) and thus no need for changes of the IP protocol stack of the mobile host. However, the SIP mobility cannot support TCP connections, which limits its usage only for real-time communications using UDP.

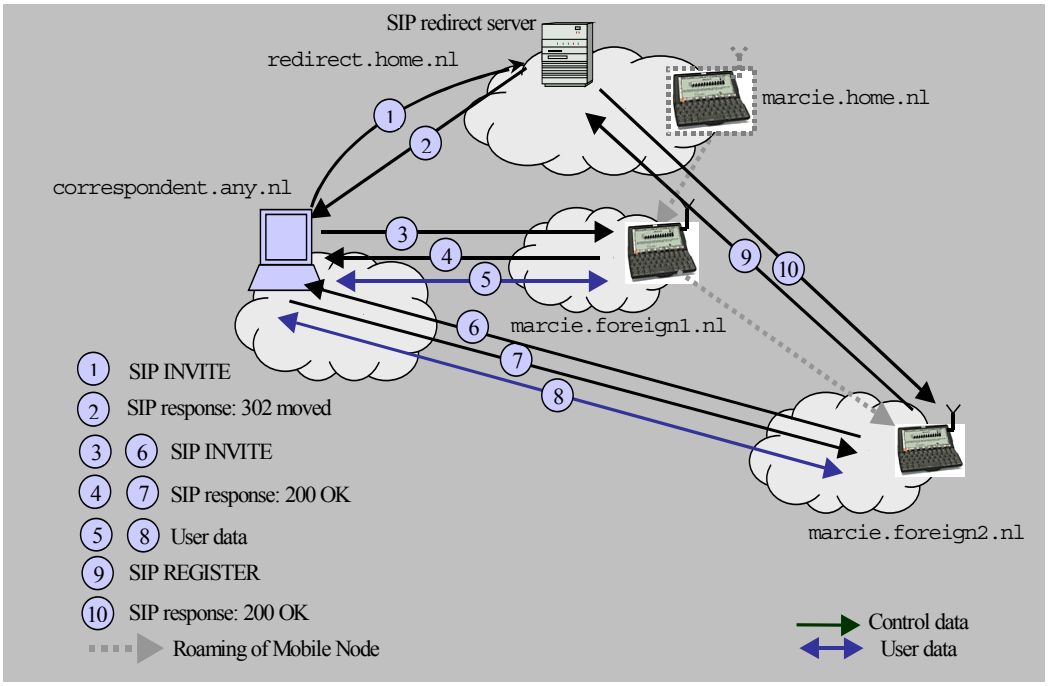


Figure 8.3.2-2: SIP mobility support

8.3.3 Mobile IP and SIP

In Section 8.3.2 a scenario has been presented wherein the SIP protocol is used to provide mobility support to a mobile node that is roaming between different IP subnetworks. However, the SIP mobility cannot support TCP connections, which limits its usage only for real-time communications using UDP. By combining the SIP and Mobile IP protocols this disadvantage is solved (Figure 8.3.3-1). In this combination SIP is used on top of Mobile IP, in which case Mobile IP provides to SIP the same IP addressing transparency as it provides to TCP. The messages in Figure 8.3.3-1 are used as follows:

Messages (1) to (7) describe the SIP session setup procedure. Messages (8) to (12) describe the Mobile IP tunnelling to the Care-of Address procedure.

Note that specific access networks to the Internet, such as cellular networks have their own mobility support. However, at the moment, they do not provide mechanisms for roaming between different types of access networks.

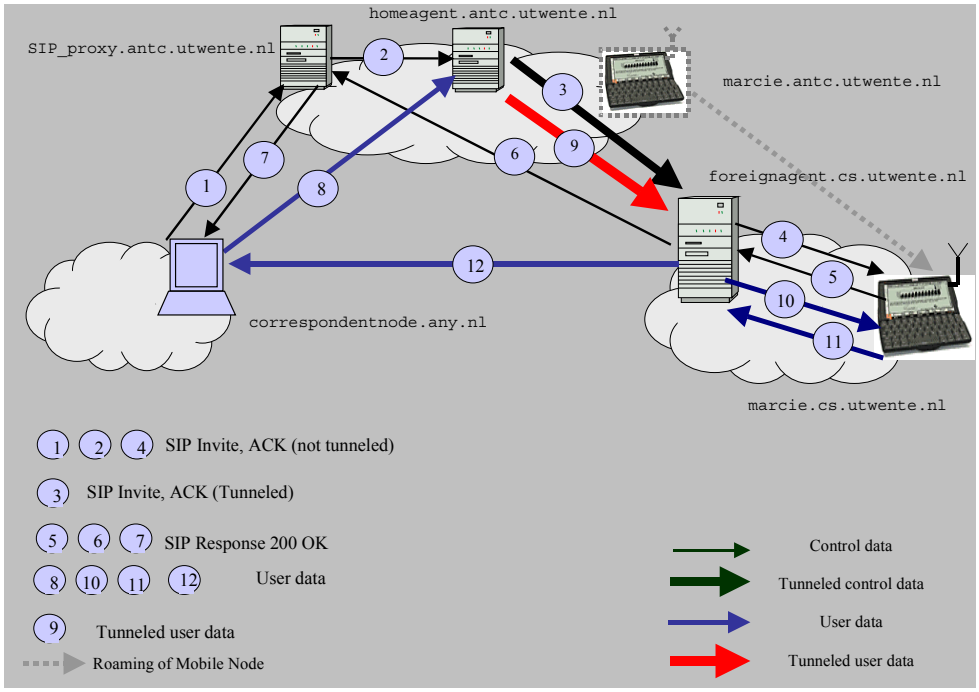


Figure 8.3.3-1: SIP session setup and Mobile IP tunneling to the care of address

8.4 RSVP and protocols for mobility support

It is expected that real time applications running on a mobile node, will need to use the combination of a specific protocol used for IP mobility and another one used for resource reservation, e.g., RSVP. This section presents the usage of RSVP with the protocols for mobility support explained in Section 8.3:

RSVP and Mobile IP

By combining the capabilities of RSVP and Mobile IP it will be ensured that first, the application will not be terminated when the MN will move into another IP subnetwork and second, the required QoS will be satisfied.

Figure 8.4 –1 presents the flow diagram of an interoperability scenario between the Mobile IPv4 and RSVP protocols. In this scenario it is assumed that the Mobile Node is roaming into a foreign domain. The Mobile Node has discovered its Care-of Address via the Foreign Agent. Furthermore, its Care-of Address is registered with the Home Agent.

Messages (1) to (6) are used to reserve the resources on the upstream direction, i.e., from Correspondent Host to Mobile Node. For the downstream direction, i.e., from Mobile Node to Correspondent Host, the resource reservation is accomplished using messages (7) and (8). Subsequently, the IP user data transfer on the upstream and downstream directions takes place. At the moment that one of the end terminals is willing to terminate the application, then the resource release procedures are initiated. Messages (9) to (14) are used to release the resources on the upstream direction, while messages (15) and (16) are used to release the resources on the downstream direction.

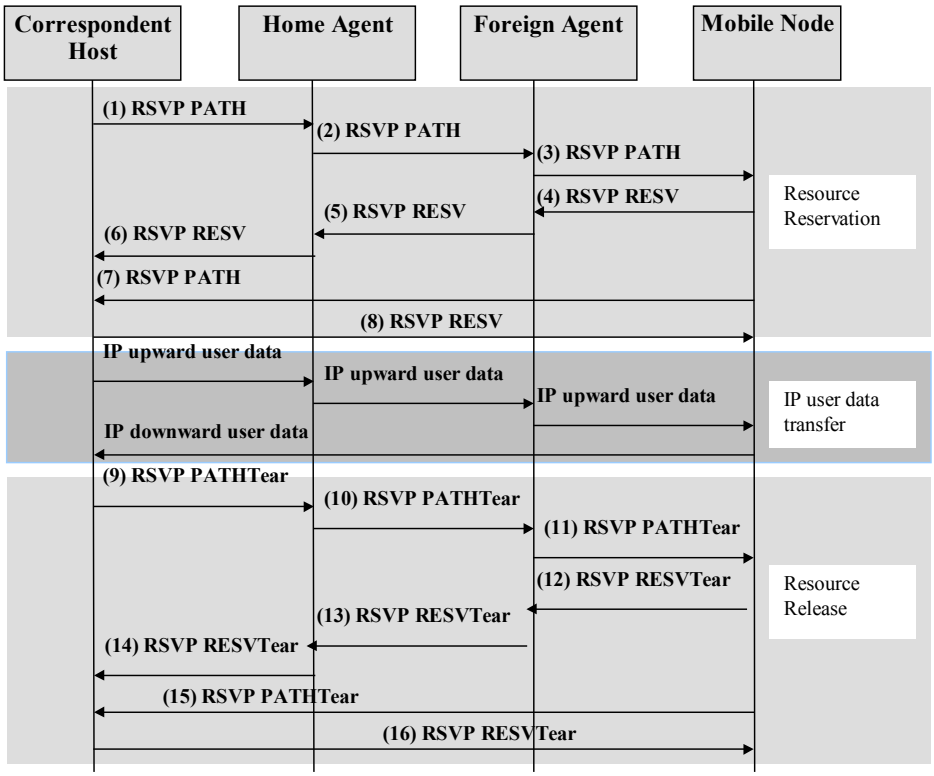


Figure 8.4 -1: Interoperability Mobile IPv4 and RSVP

RSVP and SIP (used for mobility support)

By combining the capabilities of the RSVP and SIP (used for mobility support) protocols it is ensured that first the required QoS is satisfied and second that the session in progress will not be terminated when the MN will move into another IP subnetwork.

Figure 8.4-2 presents a flow diagram of an interoperability scenario between the SIP and RSVP protocols. Messages (1) to (6) are used to setup the session between the Correspondent Host and the Mobile Node. The messages (7) and (8) accomplish the resource reservation for the upstream direction. For the downstream direction the messages (9) and (10) accomplish the resource reservation. Subsequently, the IP user data transfer in both directions, upstream and downstream, takes place. When one of the end terminals wishes to terminate the application then a combination of resource release and a session release procedure will be accomplished. Messages (11) to (14) are used to release the resources on both the upstream and downstream directions. Messages (15) to (17) are used for session release.

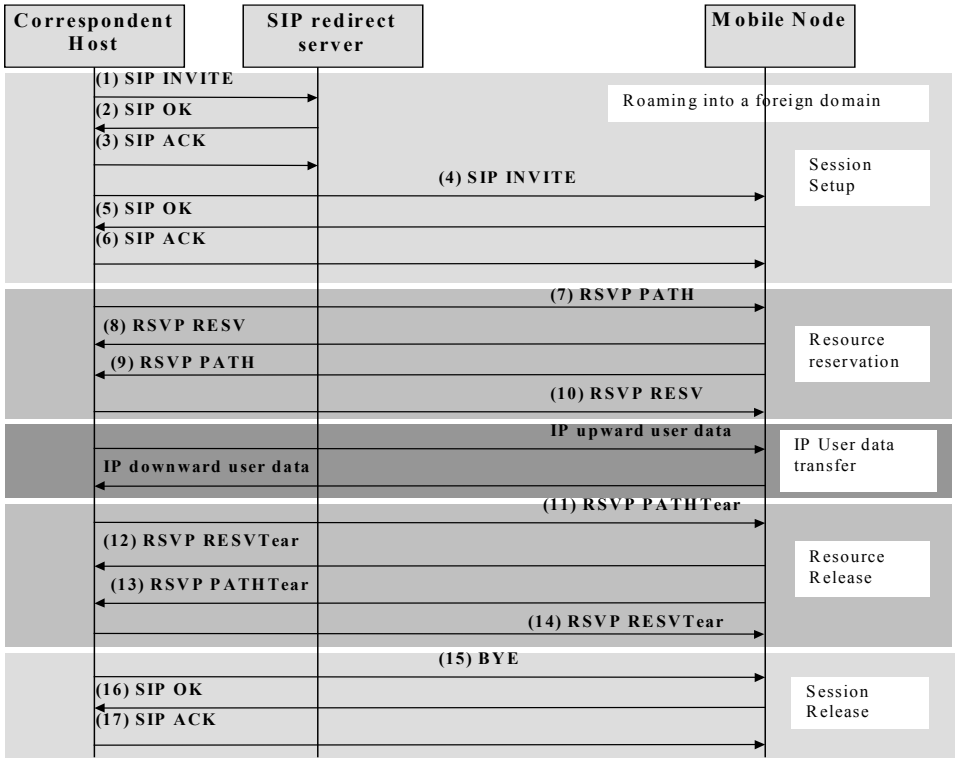


Figure 8.4-2 Interoperability between SIP (used for mobility support) and RSVP

8.5 General Description of QoS and Mobility Framework

This section describes a general framework for QoS and Mobility, the requirements that this framework will have to support, the functional entities and protocols used.

8.5.1 Design Goal and Requirements

The future Internet will have to support a large variety of applications with different QoS demands that are running on different types of wireless or wired terminals connected on various types of networks. This requires that the future Internet architecture will have to be very flexible and open, capable of supporting all these different types of networks, terminals and applications. Furthermore, it can be seen that existing QoS management architectures (such as OSI QoS, QoS-A, OMEGA, etc., see [KaRe00] for a description of these architectures) are optimised to operate efficiently in small access networks. It is therefore reasonable to consider that a framework should provide the opportunity to support efficient local access QoS management architectures. Furthermore, the core network should have a flexible and scalable architecture for providing QoS. In several papers and reports, (e.g. [RFC 2475], [RFC2638]) it is claimed that the Differentiated Services architecture is a flexible and scalable QoS architecture and should be used in the core network of the future Internet architecture.

Based on the above given considerations a list of requirements that should be fulfilled by the proposed framework architecture is composed:

- The IP core network is based on the Diffserv network architecture.

- Both static and dynamic provisioning of resources in the IP Diffserv core network should be supported.
- The access networks may support any of the existing IP QoS management architectures, like Integrated Services Architecture, Differentiated Services Architecture, QoS capabilities of the access technology, overprovisioning of resources, etc. In the situation that an access network operator configures its network in such a way that it becomes overprovisioned, applications may or may not gain the demanded QoS.
- The access networks may support different access technologies, e.g. Bluetooth, General Packet Radio Service (GPRS), Universal Mobile Telecommunication System (UMTS), Wireless Local Area Network (W-LAN).
- Each mobile node that supports multiple access technologies should be able to select the most efficient and cost-effective technology that supports the application QoS requirements.
- Handovers between different access networks and technologies should be supported.
- Global QoS interoperation of local QoS mechanisms should be possible.

8.5.2 Separation of QoS Session Negotiation and Resource Reservation

Especially in a wireless and mobile environment, it is very important to be able to separate the negotiation of a communication session and its QoS from the actual reservation of the communication resources. In wireless networks the available resources are scarce, and therefore, efficient resource reservation mechanisms should be applied. Efficient resource reservation mechanisms should reserve resources only when it is certain that these resources will be used. Furthermore, the scalability of a network and in particular the scalability of a large IP core network will be enhanced if its resources are only reserved when it is certain that they will be used.

- A separation between session layer control (or negotiation) and bearer control (or resource reservation) as proposed in [ErOh00] is justified, since negotiating service at the session level certainly adds value to the QoS and mobility framework for several reasons:
- Session negotiation can establish the session before claiming the resources. This avoids an unnecessary reservation of communication resources due to unavailability of a suitable (e.g. high speed, when a mobile host is "on the move") access network, incompatible session / application layer parameters, shortage of resources in the remote access network, or a remote user not accepting the invitation.
- Separation of session negotiation and resource reservation allows for mobility issues to be sorted out before resources are reserved. For instance, the (Care-of) IP address of a mobile host can be obtained before any resources are reserved.
- In the Internet protocol suite, separate protocols are available for session control and resource reservation. These are SIP and RSVP.

8.5.3 QoS Mobility Service Classes

The service classes defined by the Intserv architecture and Diffserv architecture do not include support for mobility and therefore roaming users will not be able to use applications with a

satisfactory QoS. Therefore, a proposal and a specification of two new QoS service classes is given, which are extensions of the aforementioned classes:

- *Mobility Dependent Locally Guaranteed (MDLG)* is associated with non-adaptive applications. In this service class the QoS requirements can be statistically guaranteed locally in a subnetwork*. The statistically guarantees of a QoS requirement are related to a probabilistic guarantee, e.g., a QoS requirement can be guaranteed with a certain probability, e.g., 95%.

  When the mobile host moves to another IP subnetwork that provides a lower QoS, the application will re-negotiate the QoS parameters by specifying the lowest QoS limit that the application is willing to accept. Note that when a mobile host moves to another IP subnetwork then the handover requests succeed with higher probability than the new user requests. If the negotiated QoS is lower than its limit than the application terminates the session.

- *Mobility Dependent Adaptive (MDA*) is associated with adaptive applications. This class consists of several relative sub-classes assigned to different QoS levels subsequently. The MDA is similar to the AF-PHB [RFC2597] defined by the Diffserv architecture and to the Controlled Load [RFC1633] defined by the Intserv architecture. When the mobile host moves to another IP sub-network and if the sub-network satisfies the QoS requirements then the application continues with the same QoS, otherwise it adapts to another sub-class with lower QoS. Afterwards, all the other hosts that are probably connected to the roaming host have to be informed about the reduction in the QoS. Furthermore, the handovers succeed with higher probability than the new user requests of the same sub-class. If there are no resources available for none of its sub-classes than its traffic is treated as best effort traffic.

- *Best Effort* is associated with applications requiring no QoS like file transfers or e-mail. No special provisions are taken for moving hosts.

8.5.4 Framework Entities and Protocols

The QoS and mobility framework depicted in Figure 8.5.4-1 consists of three major building blocks: Hosts, local Access Networks, and a Diffserv Core Network. Hosts represents the calling and called hosts, i.e. Host X and Host Y, respectively. The local Access Network includes possible efficient local QoS mechanisms. The Diffserv Core Network is represented as one Diffserv domain, but it may consist of more than one Diffserv domain. Each block includes the main active functional entities that have to be used in the QoS and mobility framework. In the following subsections first examples of protocols that may be used to provide the intercommunication between the applied functional entities are described, followed by a description of each functional entity per block.

* The definition of the subnetwork in this case is the same as defined in Mobile IP [Per98].

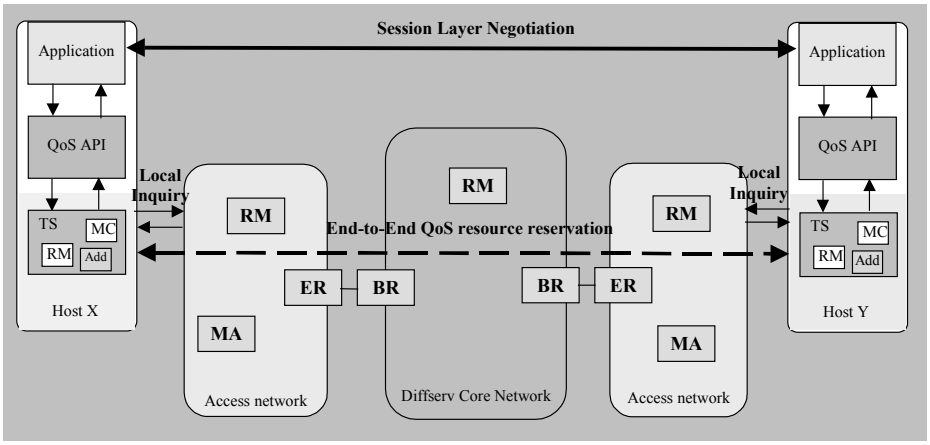


Figure 8.5.4-1 QoS & Mobility Framework building blocks and protocols

8.5.5 Protocols

The following examples of protocols may be used to interconnect the various functional entities in the QoS and Mobility framework.

- Session Layer Negotiation protocol is any protocol that the Application entities will use for initiating a session between hosts. It might be SIP or H323, or it might be an entirely new protocol, as long as it fits within the framework requirements.
- End-to-End QoS Resource Reservation protocol is a protocol that will be used for resource reservation in the end-to-end path. It might be RSVP, RSVP aggregation, or tunnelled RSVP.
- Local Inquiry is a simple protocol, which may be used for local resource inquiry (Figure 8.5.4-1), i.e. communicating with the access network resource manager. This protocol can be implemented using the SNMP [RFC1905] or COPS [RFC2748].

8.5.6 Host Functional Entities

The following host functional entities are required for the QoS and mobility framework:

- Application is an abstraction for a QoS aware application. The QoS aware application is any application that is able to specify its traffic and QoS requirements, based on which the QoS API determines to which QoS Mobility Service Class it belong, i.e. its service profile. It is also required that these applications support the session layer protocols. In case of for instance SIP the application will be a SIP client.
- QoS API is the abstraction for mechanisms that based on application attributes (e.g. audio, video) and QoS requirements determine the application's service profile. It will perform mapping of the application service profile in an understandable form for the underlying host Resource Manager and also the mapping of Resource Manager messages in an understandable form for the application itself to let it know whether the session initiation is going to be performed or not. Of course these mechanisms will be able to detect when the host has entered another access domain, e.g. using the Mobility Client. (See also [ErOh00] for a similar QoS API definition)
- The host Resource Manager (RM) is the abstraction for the entity that is in fact a QoS decision point for the end host. It will provide the mechanism for resource control within

the end host based on request and responses it receives from the QoS API and Local Inquiry protocol messages. The Resource Manager should interpret the QoS Mobility Service class parameters and based on their interpretation and the Local Inquiry protocol messages it should decide on whether there are enough network resources locally for the Application to initiate a session.

- The Mobility Client (MC) is a functional entity that in combination with the Mobility Agent located at the access networks is providing IP mobility management.
- Technology Selector is the entity, which will be part of any mobile host that wishes to select a certain underlying radio technology and/or underlying wired technology supported by an access network. The TS is able to provide this selection by using certain criteria, based on e.g. application's service profile. Depending on the required profile information, the TS will encompass various numbers of functional entities. For example, the TS may encompass the RM, MC but also some other Host entity that will provide the authentication and accounting management (see block ADD in Figure 8.5.4-1).

Figure 8.5.6-1 depicts the situation that the host is able and willing to perform the technology selection. In this situation the host is capable of selecting one of the underlying radio technologies, e.g. Bluetooth and GPRS. The main operation is as follows.

The Host needs to start a real time application, e.g. VoIP. The QoS API will perform the mapping of the application requests to parameters that are understood by the TS. If the TS entity has the required profile information to perform the technology selection it will do so and it will inform the application entity (i.e. session client) about it. Otherwise, the TS will sent one request, i.e. `TS_Inquiry REQUEST` to the Bluetooth access technology and another request to the other access technology, e.g. GPRS. Note that the `TS_Inquiry Request` message may be sent in either one or more than one messages. These requests will include query information regarding for example: (1) the requested QoS parameters, (2) the authentication restrictions, (3) accounting restrictions, (4) the financial and complexity cost of a connection to the core network, etc. This query information will have to be distributed to all functional entities, e.g. RM, MA, in the access technologies that will be able to answer them. The replies of each queried functional entity will be either sent individually in one `TS_Inquiry RESPONSE` message or they will all be combined in one `TS_Inquiry RESPONSE` and sent to the Host TS. The TS by applying the predefined criteria will choose one of the access technologies and it will inform the application entity (i.e. session client) about it.

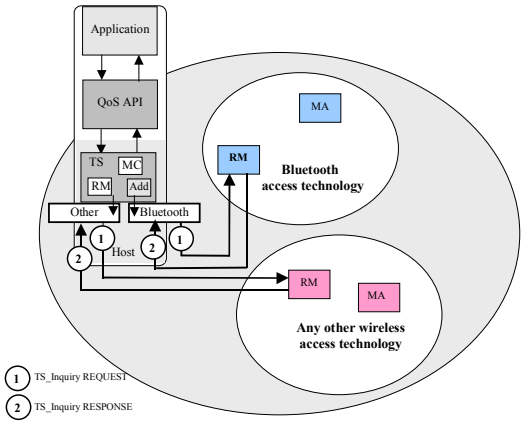


Figure 8.5.6-1 Example of technology selection accomplished by the host

8.5.7 Diffserv Core Network Functional Entities

In the requirements it is noted that the IP core network should be a Diffserv network. Therefore, the functional entities that will be located in this block and are used in the QoS and mobile framework should be in full compliance with the Diffserv network architecture definitions. The functional entities that are located in the Diffserv core network region are:

- The Resource Manager (RM) performs the resource allocation and admission control for the core network either statically or dynamically. It can be centralised (e.g. see [RFC2638] or [TeChi99]) or distributed within the core network (e.g. [RFC2475]).
- The Border Routers (BR) are standard Diffserv border routers, which should be able to treat traffic aggregates from the adjacent domains in compliance with the SLS agreement. In some particular cases they might also perform other tasks for interoperation with other, non-Diffserv domains.

8.5.8 Access Network Functional Entities

The functional entities that are located in a local Access Network and are necessary for the QoS and mobility framework are the following:

- The Resource Manager (RM) in the access network is the same as the role of the Diffserv core network Resource Manager. It is responsible for resource allocation and admission control within the access domain. Its specific realisation depends on the IP QoS architecture that will be used at the access network.
- Edge Router (ER) is an abstraction for any edge device residing at the periphery or boundary of an administrative domain. Its functionality depends on specific IP QoS architecture used at the access network.
- Mobility Agent (MA) is an abstraction for all the mechanisms that are related to the IP mobility protocols, e.g. Mobile IP and SIP. It may for example represent a Home Agent or a Foreign Agent or a SIPS (SIP redirect server).

8.6 QoS & Mobility Framework Architecture Operation

Specific realisations of the framework architecture will depend on the QoS architectures used at the *Access Network*s, resource provisioning mechanisms at the *Core Network,* management of mobility support, and related protocols.

The operation of this framework architecture can be described as consisting of certain procedures, that can be performed either sequentially or simultaneously, depending on the specific realisation of the framework:

- QoS "Session setup": a session is initiated between the end hosts that are willing to start an application.
- QoS "Resource reservation": reservation of the required resources in the access and / or core network.
- "IP user data transfer": the flow of IP user data traffic.
- QoS "Resource release": the reserved resources are released.
- QoS "Session termination": the session is terminated.

- Network attachment: a mobile host attaches to a certain network, using a specific access technology.
- Network detachment: a mobile host detaches from a network

This section describes briefly an example of the operation of the QoS and mobility framework, when Intserv/Diffserv architectural framework is applied. A more detail description of the QoS and mobility framework and in particular of its operation can be found in [KaRe00]. First, an example for the start of the communication is given, i.e. *session setup*, *resource reservation*, and *IP user data transfer*. Thereafter, an example of a hand-over from one access technology to another is also described, using *network attachment*, *network detachment*, *resource release*, and *resource reservation*.

8.6.1 Applicability of the Intserv/Diffserv architectural framework in a wireless environment

For this particular example following assumption have been made: the first five procedures described above are performed sequentially. The calling user, i.e. Host X is already attached to an *Access Network* supporting the RSVP/Intserv architecture. It is attached to this network in two ways: using Bluetooth (on his home subnetwork), and using the GPRS access technology (on another subnetwork). The Host Y is also residing in an *Access Network* supporting the Intserv architecture. The *Application* in the Host X and Host Y is VoIP, i.e. non-adaptive with hard QoS requirements belonging to the Mobility Dependent Locally Guaranteed (MDLG) service class. Mobile IP manages the mobility support and *TS* at the host decides on the access technology. Host X and Host Y use SIP as *Session Layer Negotiation* protocol and RSVP enhancements as *End-to-End QoS Resource Reservation* protocol. *Application* entities can be seen as SIP clients also. The Access Network and Core Network have the static SLA, SLSes respectively in place and the interoperability is performed at the BR, which is in fact an RID border router. Also the RSVP tunnel is used to carry the RSVP messages through the *Diffserv Core network*.

Figure 8.6.1-1 depicts the overall functionality of QoS and Mobility framework, when Intserv/Diffserv architectural framework is applied.

8.6.1.1 Start of Communication

QoS "Session Setup"

(S1): A calling user, *Application* entity in Host X, starts up a VoIP session to communicate with the called user – *Application* entity in Host Y.

(S2): At Host X, the *QoS API,* based on application attributes and QoS requirements, determines the Mobility Depend Locally Guaranteed (MDLG) service profile and translates these parameters in an understandable form for the underlying entities. Since, the Access Network is supporting the RSVP/Intserv architecture the RM at Host X will use RSVP as a *Local Inquiry* protocol to inquire within its access network whether there are enough resources available for its application. If the inquiry is positive than Host X will initiate a session with Host y. Additionally, since Host X is able to support more than one access technology, by using the technology selection procedure described in Section 8.5.6, *TS* will select the access technology that satisfies the predefined technology selection criteria for MDLG. Suppose for now, Bluetooth is selected.

(S3): By means of a SIP message, the calling user invites the called user, i.e. Host Y to start a VoIP session. The session description in the SIP message will contain the session name, purpose,

media and timing information, and additional information regarding the bandwidth to be used by the VoIP application.

(S4): The called user, i.e. Host Y will perform the same procedures as Host X in ***(S2)*** and it will inform the calling user about the successful session setup completion, if the session is acceptable and the resources for the session are available in the remote access network.

QoS "Resource reservation"

(S5): Since both *Access Networks* support the RSVP/Intserv concept, then the calling user, i.e. RM in Host X, must start the "resource reservation" procedure by sending RSVP PATH messages. Thus when the RSVP messages arrive at the border between the *Access Networks* and *Diffserv Core Network* the same interoperability functions will be performed as described in Chapter 3 and Chapter 4 in order to reserve resources negotiated between Host X and Host Y at the session layer. In the *Diffserv Core Network,* RSVP tunnel will be set up to carry RSVP messages through the Diffserv Core Network.

"IP user data traffic phase"

(S6) After successful completion of the QoS "session setup" and the QoS "resource reservation" procedure Host X and Host Y may start sending IP user data traffic, i.e. VoIP speech data.

8.6.1.2 Handover

If Host X gets out of the coverage of its home Bluetooth network, it has to rely on the GPRS network to continue the session. So, the assumption here is that Host X performs a handover from the Bluetooth subnetwork to the GPRS subnetwork during the exchange of data traffic, in order to remain connected to the Intserv-based access network. This will be handled by the *MC* and *MA* entities, in conjunction with the *RM* entities. The specific example given here exemplifies a so-called hard handover; i.e. the old link is broken down before the new link is established.

Network detachment / Resource release

Possibly, the network detachment will be performed automatically, because Host X will loose contact with the Bluetooth network, and the soft state for Mobile IP and RSVP will be removed from the network. Alternatively, the state in the network is removed because it is being replaced by a new state, because of network attachment.

Network attachment

(Si-1): The *MC* entity of the Host X will try to find out from *MA* in the GPRS subnetwork what its new identity, i.e. IP address, is. In Mobile IP this is known as Care-of Address discovery.

(Si-2): The *MC* will send its new IP address to the *MA* in its home subnetwork, i.e. it will register. From now on, all IP packets will be tunnelled to the new IP address.

Resource reservation

(Si-3) Host X and Host Y will keep the current the QoS session setup and will re-initiate the "resource reservation". The resource reservation will be done using the same QoS requirements as before. If the reservation is not successful, the QoS requirements will be renegotiated with the application. This may lead to either successful reservation or session termination.

(Si-4) After successful " resource reservation" data exchanges follow as in ***(S6)***.

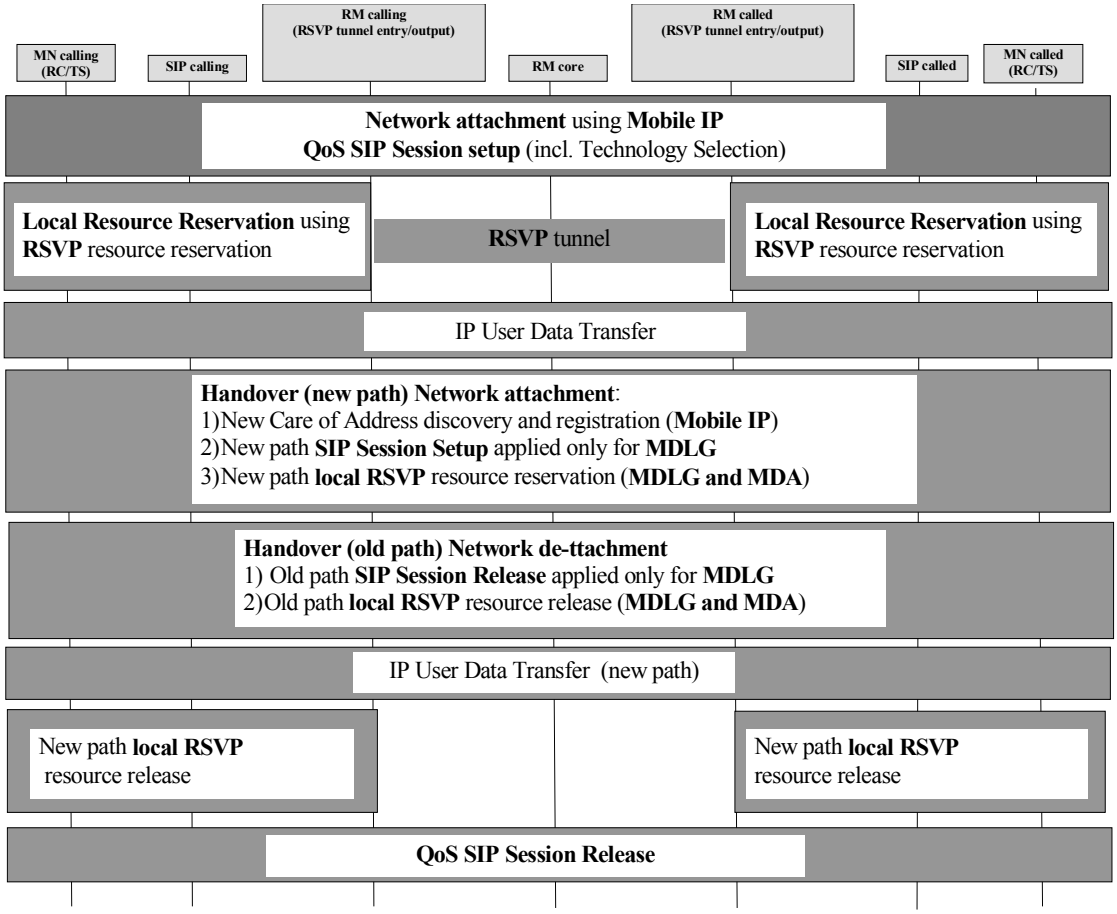


Figure 8.6.1-1 Functionality phases of the QoS and Mobility framework, when Intserv/Diffserv architectural framework is applied

[9] CONCLUSIONS

9.1 Introduction

This thesis was intended as a contribution towards end-to-end QoS deployment on the wired and wireless Internet. A one year research work has proven right what was stated in the beginning: that enabling QoS in the Internet is a difficult task due to the complexity it introduces in the overall wired and wireless Internet architecture influence and that in order to provide flexible feasible solutions applicable to future Internet services it is necessary to take into account the QoS mobility issues.

9.2 The outcome of the defined activities

Recalling the activities identified towards reaching the objective this thesis:

- Literature Study on Internet QoS, Intserv, RSVP, Diffserv
- Continuous monitoring of Intserv/Diffserv integration proposals
- Designing a general Intserv/Diffserv architecture (without reference to network configuration and implementation environment)
- Designing in detail the border router architecture residing at the Diffserv network to support RSVP/Intserv and Diffserv interoperability
- Studying and creating the implementation environment (select existing Diffserv, Intserv, RSVP implementation, plan and install a network configuration (in conjunction with the existing testbed in Ericsson))
- Implementation of the border router prototype based on the detailed design. Possibly using the already existing RSVP/Intserv and Diffserv implementations
- Testing the correct behaviour of the prototype implementation in a planned testing environment
- Studying QoS mobility issues and possibilities for applying the general Intserv/Diffserv architecture in an environment of mobile hosts and wireless access networks.

The first two activities identified required continuos monitoring of the work on Internet Engineering Task Force (IETF) Integrated Services (Intserv), Integrated Services over Specific Link Layers (ISSLL), Resource ReserVation Protocol (RSVP) and Differentiated Services (Diffserv) working groups. As such these activities where fully accomplished. Most of the knowledge gathered is bundled together in Chapter 2, although of course it is also spread all over the thesis.

Regarding the third activity a general Intserv/Diffserv architecture was defined, which is described in Chapter 3. This architecture defines certain requirements, such as the SLA, SLS respectively is assumed to be static. The decision was made due to the still open issues related to the Diffserv dynamic provisioning of the resources. However, this does not mean that the general Intserv/ Diffserv architectural framework is limited only to the static SLA. On the contrary by choosing the Diffserv Border Router as the RSVP/Intserv – Diffserv translator, this architectural framework can be easily extended to support dynamic SLAs also.

Chapter 4 is related to the fourth activity defined. In this chapter a detailed design of a RSVP/Intserv–Diffserv border router was given. This design identifies all the necessary functional blocks for handling the RSVP messages and Diffserv packets. Furthermore Chapter 5 identifies the role of the same router, when used as an end-point of an RSVP tunnel. RSVP-tunnel, was introduced to provide Diffserv transparency to RSVP messages.

In order to prove the feasibility of the design a basic prototype implementation was developed. The outcome of the activities defined in bullets 5, 6 and 7 is documented in chapters 6 and 7. The testbed environment was set up and the appropriate free software packages where installed for Intserv / Diffserv support under Linux. However, due the unexpected runtime errors received with the RSVP daemon (Appendix B) and due to the limited amount of time the activity related to the implementation of the prototype and appropriate testing results has not been fully completed. A basic prototype has been developed and some test has been performed to check whether the behaviour of the prototype is correct.

Finally the outcome of studies on the last activity are given in Chapter 8. QoS mobility issues are addressed in terms of combining the IP QoS architectures and mobility protocols. Further a general framework for QoS and mobility support was developed. Finalising with the applicability of the Intserv/Diffserv architectural framework defined in chapter [3] to the QoS and Mobility framework.

Derived from the work above the Intserv / Diffserv interoperability is a complex topic, which needs further studies and research especially when the Diffserv dynamic resource provisioning mechanisms are to be applied. Intserv over Diffserv is particularly advantageous in those types of networks where the users would like to have control and guaranteed services per-flow or /and where the network resources are scarce. Thus, it is almost perfect to be applied in wireless access environments, where of course the mobility, handovers issues and RSVP would need extensive research for efficient solutions.

Independently, the RID router is a powerful tool for any network – service provider, since regardless of the whether the IP QoS architecture chosen for his domain is Intserv or Diffserv, it can still offer the services offered by both architectures. Thus, by means of RID router it will be able to satisfy its customers needs independently of the type of applications the user is using, i.e. the user might prefer to use RSVP aware applications rather than to just send data that will be handled in Diffserv aggregates.

9.3 Future Work

Although based on the above, the activities identified towards the objective of this thesis are almost fully completed, still in order to achieve end-to-end QoS in a wired and wireless Internet

architecture requires a lot more research. There can be a number of research directions derived from this thesis for future work.

Initially the most important point would be full implementation prototype of the RID router (recall here also 6.5), which would enable a lot valuable testing with different types of experiments in a wireless and wired environment related to the combination of Intserv and Diffserv. These experiments would probably prove the above statement as true. In this direction having RSVP-aware applications running on wireless terminal would be particularly desirable. Extending the Intserv / Diffserv framework to support dynamic SLSes is also part of future work activities. Furthermore, the RID router prototype implementation would be used for experimenting with other protocols than RSVP such as RSVP aggregation or Load Control.

To conclude the future work activities would be directed to proving the feasibility of the QoS mobility architectural framework defined in Chapter 8, since I believe that this framework is a good foundation for supporting future wired and wireless Internet QoS services.

This thesis was intended as a contribution towards end-to-end QoS deployment on the wired and wireless Internet. A one year research work has proven right what was stated in the beginning: that enabling QoS in the Internet is a difficult task due to the complexity it introduces in the overall wired and wireless Internet architecture influence and that in order to provide flexible feasible solutions applicable to future Internet services it is necessary to take into account the QoS mobility issues.

[10] REFERENCES

[AlGi00] Almersberger, W., et al "A Prototype of Implementation for the Intserv Operation over Diffserv Networks", Globecom 2000.

[AlSa99] Almersberger, W., Salim, J., H., Kuznetsov, A., "Differentiated Service on Linux", draft-almersberger-wajhak-diffserv-diffserv-linux-01.txt, May 1999. Available via http://lrcwww.epfl.ch/linux-diffserv/

[Al99] Almersberger, W., "Linux Network Traffic Control – Implementation Overview", April 1999. Available via http://lrcwww.epfl.ch/linux-diffserv/

[BaIt00] Baker, F., Iturralde, C. Le Faucher, F., Davie, B., "Aggregation of RSVP for Ipv4 and Ipv6 Reservations", draft-ietf-issl-rsvp-aggr-02.txt, March 2000 (work in progress)

[BeBi99] Bernet, Y., Binder, J., Blake, S., Carlson, M., Davies, E., Ohlman, B., Werma, D., Wang, Zh., Weiss, W., "A framework for Differentiated Services", draft-ietf-diffserv-framework-02.txt, February 1999. (work in progress)

[BeDa94] Besse, L., Dairaine, L., Fedaoui, L., Tawbi, W., Thai, K., "Towards an Architecture for Distributed Multimedia Application Support", Proc., International Conference on Multimedia Computing and Systems, Boston, May 1994.

[Ber99] Bernet, Y. "Format of the RSVP DCLASS Object", IETF draft, draft-ietf-issll-dclass-01.txt, October 1999. (work in progress)

[BeSm00] Bernet, Y., Smith, A., Blake, S., Grossman, D., "A conceptual model for Diffserv Networks", IETF Draft, draft-ietf-diffserv-model-03.txt, May 2000. (work in progress)

[BeYa00] Bernet, Y., Yavatkar, R., Ford, P., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Felstaine, E. "Framework for Integrated Services operation over Diffserv Networks", draft-ietf-issll-diffserv-rsvp-04.txt, March 2000 (work in progress)

[BoSa00] Bouch, A., Sasse, M. A., DeMeer, H., "Of packets and People: A user-centred Approach to Quality of service", June 2000.

[Chi00] Chimento, P., " Bandwidth Broker – A view of Service Level Agreements, Service Level Specifications and resource requests", V0.1-working document, June 2000.

[ClWr97] Clark, David; Wroclawski, John. An approach to Service Allocation in the Internet", IETF draft, draft-clark-diff-svc-alloc-00.txt, July 1997.

[DuYa99] Durham, D., Yavatkar, R., "Inside the Internet's Resource reSerVation Protocol: Foundations for Quality of Service", John Willey & Sons, Inc., 1999.

[ErOh00] Eriksson, A., Ohlman, B., "A framework for On-demand QoS Bearers over Fixed and Mobile Internet Access", paper submitted to the IWQoS 2000, June 5-7, 2000.

[FeHu98] Fergusson, P., Huston, G., " Quality of Service: delivering QoS on the Internet and in Corporate Networks", Wiley Computer Publishing, 1998.

[FlJa95] Flloyd, S., Jacobson, V., "Link Sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol.3, No. 4, pp. 365 – 386, August 1995.

[FlJa93] Flloyd, S., Jacobson, Van., "Random Early Detection Gateways for Congestion Avoidance", Transactions on Networking, August 1993.

[Gro00] Grossman, D., "New terminology for Diffserv", IETF Draft, draft-ietf-diffserv-new-terms-02.txt, November 1999.

[Kar99] Karagiannis, G., "Mobile IP: State of the Art", Internet Next Generation report, located at http://ing.ctit.utwente.nl/WU4/Documents/mobip_a.pdf, 1999.

[KaRe00] Karagiannis, G., Rexhepi, V., " A Framework for QoS&Mobility in the Internet Next Generation", Internet Next Generation Report, located at http://ing.ctit.utwente.nl/WU4/Documents/frame_a.PDF

[Kil99] Kilkki, K., "Differentiated Services for the Internet", Macmillian Technical Publishing, USA, 1999.

[PeJo00] Perkins, C., Johnson, B., J., "Route Optimisation in Mobile IP", Internet draft, draft-ietf-mobileip-optim-09.txt, Work in progress, February 2000.

[Per97] Perkins, C., E., "Mobile IP", IEEE Communications Magazine, May 1997.

[Per98] Perkins, C., E., "Mobile networking through mobile IP", IEEE Internet Computing, 1998.

[RFC1256] Deering, S., (ed.), "ICMP Router Discovery Messages", RFC 1256, August 1989.

[RFC1633] Braden, R., Clark, D., Shenker, S., "Integrated Services in the Internet Architecture: An Overview", IETF RFC 1633, 1994.

[RFC1853] Simpson, W., "IP in IP tunnelling", RFC 1853

[RFC1905] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, 1996.

[RFC2113] Katz, D., "IP router alert option", February 1997

[RFC2002] Perkins, C., E., "(ed.) "IP Mobility Support", RFC2002, proposed standard. IETF Mobile IP Working Group, Oct., 1996.

[RFC2003] Perkins, C., "IP encapsulation within IP", RFC2003, October 1996.

[RFC2205] Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S., " Resource Reservation Protocol (RSVP) Version 1 Functional Specification", IETF RFC 2205, 1997.

[RFC2210] Wroclawski, J., " The use of RSVP with IETF integrated Services", IETF RFC 2210, 1997.

[RFC2211] Wroclawski, J., "Specification of the Controlled-Load Network Element Service, September 1997.

[RFC2212] Shenker, S., Partridge, C., Guerin, R., "Specification of Guaranteed Quality of Service", September 1997.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Zh., Weiss, W., "An architecture for Differentiated Services", IETF RFC 2475, 1998.

[RFC2327] Handley, M. and V. Jacobson, "SDP: session description protocol", RFC 2327, 1998.

[RFC2481] Ramakrishnan, K., Floys, S., "A proposal to add explicit Congestion Notification (ECN) to IP", January 1999.

[RFC2474] Nichols, K., Blake, S., Baker, F., Black, D., " Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", November 1998.

[RFC2543] Handily, M., Schulzrinne, H., Schooler, E., Rosenberg, J., " SIP: Session Initiation Protocol", IETF RFC 2543, 1999.

[RFC2597] Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., "Assured Forwarding PHB group", IETF RFC 2597, 1999.

[RFC2598] Jacobson, V., Nichols, K., Poduri, K., "An Expedited Forwarding PHB", IETF RFC 2598, 1999.

[RFC2638] Nichols, K., Jacobson, V., Zhang, L., " A two-bit Differentiated Services Architecture for the Internet", IETF RFC 2638, 1999.

[RFC2746] Terzis, A. Krawczyk, J., Wroclawski, J., Zhang, L., "RSVP operation over IP Tunnels", IETF RFC2746, January 2000.

[RFC2748] Durham, D., Boyle, J., Cohen, R., Herzog, S., Raja, R., Sastry, A., "The COPS (Common Open Policy Service) Protocol ", IETF RFC, January 2000.

[Ra99] Radhakrishman, S. "Linux – Advanced Networking Overview Version 1", Information and Communications Technology Centre, Department of Electrical Engineering &Computer Science, The University of Kansas Lawrence, KS 66045-228, August 1999.

[ReKa00] Rexhepi, V., Karagiannis, G., Heijenk, G., " A framework for QoS & Mobility in the Internet Next Generation" EUNICE 2000 proceedings, September 2000.

[RoVi00] Rosen. E., Viswanathan, A., Callon, R., "Multiprotocol Label Switching Architecture", IETF draft, draft-ietf-mpls-arch-06.txt, Work in progress, February 2000.

[Sh99] Shah, N., J., " Exploring TCP Stream rate Control in LAN Environments", a thesis submitted in partial satisfaction of the requirements for the degree of Master of Science in Computer Science, June 1999.

[TeCh99] Teitelbaum, B., Chimento, P. "Qbone Bandwidth Broker Architecture" at "http://qbone.ctit.utwente.nl/deliverables/1999/d2/bboutline2.html" (work in progress)

[WeSc99] Wedlund, E., Schulzrine, H., "Mobility support using SIP", ", Second ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'99), Seattle, Washington, August, 1999

[Will99] Williams, B., "Quality of Service- Differentiated services and Multiprotocol Label Switching", Ericsson Internal White paper, August 1999.

[XiHa00] Xiao, X., Hannan, A., Bailey, B., "Traffic Engineering with MPLS in the Internet", IEEE Network, March/April 2000.

[XiNi99] Xiao, X., Ni, L., M., " Internet QoS: A big picture", IEEE Network, March/April 1999.

[Sof1] University of Southern California - Information Sciences Institute (ISI), the RSVP Project. Available via http://www.isi.edu/div7/rsvp/

[Sof2] Differentiated Service in Linux, Available via http://lrcwww.epfl.ch/linux-diffserv/

[Sof3] Tcpdump Network Tool, Network Research Group (NRG), LBL.

Available via http://www-nrg.ee.lbl.gov/

[Sof4] TTCP traffic generator, available via http://www.caip.rutgers.edu/~arni/linux/tg1.html

[Sof5] TC Traffic control tool, available via, available via ftp://ftp.inr.ac.ru/ip-routing/ version

iproute2-2.2.4-now-ss991023.tar>

[11] ABBREVIATIONS

AF	Assured Forwarding
API	Application Program Interface
BA	Behavior Aggregate
BB	Bandwidth Broker
BE	Best Effort
BR	Border Router
CL	Controlled Load Service
CBQ	Class Based Queuing
Diffserv	Differentiated Services
DS	Differentiated Services
DR	Diffserv Router
DSCP	DS Code Point
E2E	End to End
EF	Expedited Forwarding
ER	Edge Router
FA	Foreign Agent
GPRS	General Packet Radio Service
GS	Guaranteed Service
GRED	Generalized RED
HA	Home Agent
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol

Intserv	Integrated Services
IP	Internet Protocol
IPv6	Internet Protocol version 6
IR	Intserv Router
IPX	Internetwork Packet Exchange
LSP	Label Switching Path
LSR	Label Switching Router
MA	Mobility Agent
MC	Mobility Client
MDA	Mobility Dependent Adaptive
MDLG	Mobility Dependent Locally Guaranteed
MF	MultiField Classifier
MN	Mobile Node
MPLS	MultiProtocol Label Switching
QoS	Quality of Service
PHB	Per Hop Behaviour
PSTN	Public Switched Telephone Network
PSB	PATH state Block
RID	RSVP/Intserv –Diffserv
RED	Random Early Detection
RIO	Random Early detection with In/Out
RM	Resource Manager
RSVP	Resource reSerVation Protocol
RSB	RESV state Block
SDP	Session Description Protocol
SFQ	Stochastic Fairness Queuing
SIP	Service Initiation Protocol
SLA	Service Level Agreement

SLS	Service Level Specification
TBF	Token Bucket Filter
TC	Traffic Control
TCA	Traffic Control Agreement
TCP	Transport Control Protocol
TOS	Type of Service
TS	Technology Selector
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
W-CDMA	Wideband - Code Division Multiple Access
W-LAN	Wireless- Local Area Network
WFQ	Weighted Fair Queuing (WFQ)
WRR	Weighted Round Robin (WRR)

[12] APPENDIX A

This Appendix contains all the tc shell scripts used in the lab for developing the RID router prototype. They implement CBQ on Intserv Interface and DSmark and CBQ combined with GRED for edge and core Diffserv router functionality necessary for implementing AF1, AF2 and BE classes. They are presented here as implemented in the machines depicted in Figure 7.3-1.

12.1 Willow Scripts

Willow Ingress Interface – Intserv script:

```
#! /bin/sh

################################################################################
# Configuration of willow eth0 interface as classical Intserv interface #
################################################################################
TC=/local/QoS/iproute2/tc/tc
DEVICE=eth0
BANDWIDTH="bandwidth 10Mbit"

# Attach CBQ on $DEVICE. It will have handle 1:.
#   $BANDWIDTH is real $DEVICE bandwidth (10Mbit).
#   avpkt is average packet size.
#   mpu is minimal packet size.

$TC qdisc add dev $DEVICE  root  handle 1:  cbq \
$BANDWIDTH avpkt 1000 mpu 64

# Create root class with classid 1:1.

$TC class add dev $DEVICE parent 1:0 classid :1 est 1sec 8sec cbq \
$BANDWIDTH rate 10Mbit allot 1514 maxburst 50 avpkt 1000

# BE Class.
# weight is set to be proportional to "rate". Eeight=1 will work as well.
# defmap and split say that best effort traffic, not classfied by another
# means will fall to this class

$TC class add dev $DEVICE parent 1:1 classid :2 est 1sec 8sec cbq \
$BANDWIDTH rate 5Mbit allot 1514 weight 500Kbit \
prio 6 maxburst 50 avpkt 1000 split 1:0 defmap ff3d

# OPTIONAL.
# Attach "sfq" qdisc to this class, quantum is MTU, perturb
# gives period of hash function perturbation in seconds.
#
$TC qdisc add dev $DEVICE parent 1:2 sfq quantum 1514b perturb 15

# Interactive-burst class
```

```
$TC class add dev $DEVICE parent 1:1 classid :3 est 2sec 16sec cbq \
$BANDWIDTH rate 1Mbit allot 1514 weight 100Kbit \
prio 2 maxburst 100 avpkt 1000 split 1:0 defmap c0

$TC qdisc add dev $DEVICE parent 1:3 sfq quantum 1514b perturb 15


# Realtime class for RSVP

$TC class add dev $DEVICE parent 1:1 classid 1:7FFE cbq \
rate 5Mbit $BANDWIDTH allot 1514b avpkt 1000 \
maxburst 20

# Reclassified realtime traffic
#
# New element: split is not 1:0, but 1:7FFE. It means,
#     that only real-time packets, which violated policing filters
#     or exceeded reshaping buffers will fall to it.

$TC class add dev $DEVICE parent 1:7FFE classid 1:7FFF  est 4sec 32sec cbq \
rate 1Mbit $BANDWIDTH allot 1514b avpkt 1000 weight 10Kbit \
prio 6 maxburst 10 split 1:7FFE defmap ffff
```

Willow Egress Interface –Diffserv edge router script performs remarking:

```
#! /bin/sh

#####################################################################
# Configure willow eth1 device for setting appropriate DSCP     #
#####################################################################

#Path definition:
TC=/local/QoS/iproute2/tc/tc

#Set up of the proper qdisc on eth1
$TC qdisc add dev eth1 handle 1:0 root dsmark indices 64 set_tc_index

# First a DSMARK qdisc is introduced in order to change tc_index

# AF1 with 3 drop levels
$TC class change dev eth1 parent 1:0 classid 1:1 dsmark mask 0x3 value 0x0a
$TC class change dev eth1 parent 1:0 classid 1:2 dsmark mask 0x3 value 0x0c
$TC class change dev eth1 parent 1:0 classid 1:3 dsmark mask 0x3 value 0x0e

#AF2 with 3 drop levels
$TC class change dev eth1 parent 1:0 classid 1:4 dsmark mask 0x3 value 0x12
$TC class change dev eth1 parent 1:0 classid 1:5 dsmark mask 0x3 value 0x14
$TC class change dev eth1 parent 1:0 classid 1:6 dsmark mask 0x3 value 0x16

#Best effort
$TC class change dev eth1 parent 1:0 classid 1:7 dsmark mask 0x3 value 0x0

# Then a CBQ qdisc is used in order to support EF, AF and BE classes
# The following structure is very similar to those used in the core router
$TC qdisc add dev eth1 parent 1:0 handle 2:0 cbq bandwidth 10Mbit cell 8
avpkt 1000 mpu 64

$TC filter add dev eth1 parent 2:0 protocol ip prio 1 tcindex mask 0xf0
shift 4 pass_on

##### Definition of the CBQ leaf classes to support AF1, AF2 and BE

##### AF Class 1 specific setup
```

```
$TC class add dev eth1 parent 2:0 classid 2:1 cbq bandwidth 10Mbit rate
3Mbit \
avpkt 1000 prio 4 allot 1514 weight 250Kbit maxburst 21 defmap 0 borrow

$TC filter add dev eth1 parent 2:0 protocol ip prio 1 handle 1 tcindex
classid 2:1

$TC qdisc add dev eth1 parent 2:1 gred setup DPs 3 default 2 grio

# --- AF Class 1 DP 1---
$TC qdisc add dev eth1 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 10Mbit DP 1 probability 0.02 prio 2
# --- AF Class 1 DP 2---
$TC qdisc add dev eth1 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 10Mbit DP 2 probability 0.04 prio 3
#  -- AF Class 1 DP 3--- #
$TC qdisc add dev eth1 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 10Mbit DP 3 probability 0.06 prio 4

##### AF Class 2 specific setup
$TC class add dev eth1 parent 2:0 classid 2:2 cbq bandwidth 10Mbit rate
2Mbit \
avpkt 1000 prio 5 allot 1514 weight 250Kbit maxburst 21 defmap 0 borrow

$TC filter add dev eth1 parent 2:0 protocol ip prio 1 handle 2 tcindex
classid 2:2

$TC qdisc add dev eth1 parent 2:2 gred setup DPs 3 default 2 grio

# --- AF Class 2 DP 1---  #
$TC qdisc add dev eth1 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 10Mbit DP 1 probability 0.02 prio 2
# --- AF Class 2 DP 2---  #
$TC qdisc add dev eth1 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 10Mbit DP 2 probability 0.04 prio 3
#  -- AF Class 2 DP 3---
$TC qdisc add dev eth1 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 100Mbit DP 3 probability 0.06 prio 4

##### BE class specific setup
$TC class add dev eth1 parent 2:0 classid 2:6 cbq bandwidth 10Mbit rate
5Mbit \
avpkt 1000 prio 7 allot 1514 weight 3Mbit maxburst 21 bounded isolated
$TC qdisc add dev eth1 parent 2:0 red limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 10Mbit probability 0.4
$TC filter add dev eth1 parent 2:0 protocol ip prio 1 handle 6 tcindex
classid 2:6
```

12.2 Spike scripts

Egress Interface - Diffserv Core router script:

```
#! /bin/sh
################################################
#                                              #
# This script implements a DS interior router #
# It adds to eth1 a proper qdisc               #
```

```
# which implements AF1, AF2 PHBs and BE       #
#                                              #
################################################

##### Path definition:
TC=/usr/src/iproute2/tc/tc

##### Set up of the proper qdisc on eth1

# First a DSMARK qdisc is introduced in order to retrieve TOS from IP header
$TC qdisc add dev eth1 handle 1:0 root dsmark indices 64 set_tc_index

$TC filter add dev eth1 parent 1:0 protocol ip prio 4 tcindex mask 0xfc
shift 2 pass_on

# Second a CBQ qdisc is used in order to support AF and BE classes
$TC qdisc add dev eth1 parent 1:0 handle 2:0 cbq bandwidth 10Mbit cell 8
avpkt 1000 mpu 64

$TC filter add dev eth1 parent 2:0 protocol ip prio 1 tcindex mask 0xf0
shift 4 pass_on

##### Definition of the CBQ leaf classes to support AF1, AF2 and BE

##### AF Class 1 specific setup
$TC class add dev eth1 parent 2:0 classid 2:1 cbq bandwidth 10Mbit rate \
3Mbit avpkt 1000 prio 4 allot 1514 weight 250Kbit maxburst 21 defmap 0
borrow

$TC filter add dev eth1 parent 2:0 protocol ip prio 1 handle 1 tcindex
classid 2:1

$TC qdisc add dev eth1 parent 2:1 gred setup DPs 3 default 2 grio

# --- AF Class 1 DP 1---
$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 10 tcindex
classid 1:111

$TC qdisc add dev eth1 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit DP 1 probability 0.02 prio 2
# --- AF Class 1 DP 2---
$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 12 tcindex
classid 1:112
$TC qdisc add dev eth1 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit DP 2 probability 0.04 prio 3
#  -- AF Class 1 DP 3---
$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 14 tcindex
classid 1:113
$TC qdisc add dev eth1 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit DP 3 probability 0.06 prio 4

##### AF Class 2 specific setup
$TC class add dev eth1 parent 2:0 classid 2:2 cbq bandwidth 10Mbit rate
2Mbit \
avpkt 1000 prio 5 allot 1514 weight 250Kbit maxburst 21 defmap 0 borrow
$TC filter add dev eth1 parent 2:0 protocol ip prio 1 handle 2 tcindex
classid 2:2
$TC qdisc add dev eth1 parent 2:2 gred setup DPs 3 default 2 grio
# --- AF Class 2 DP 1---
$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 18 tcindex
classid 1:121
$TC qdisc add dev eth1 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit DP 1 probability 0.02 prio 2
# --- AF Class 2 DP 2---
```

```
$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 20 tcindex
classid 1:122
$TC qdisc add dev eth1 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit DP 2 probability 0.04 prio 3

# -- AF Class 2 DP 3---
$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 22 tcindex
classid 1:123
$TC qdisc add dev eth1 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit DP 3 probability 0.06 prio 4

##### BE class specific setup
$TC class add dev eth1 parent 2:0 classid 2:6 cbq bandwidth 10Mbit rate \
5Mbit avpkt 1000 prio 7 allot 1514 weight 30Kbit maxburst 21 bounded
isolated
$TC qdisc add dev eth1 parent 2:6 red limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 10Mbit probability 0.4

$TC filter add dev eth1 parent 1:0 protocol ip prio 1 handle 0x0 tcindex
classid 1:161

$TC filter add dev eth1 parent 2:0 protocol ip prio 1 handle 6 tcindex
classid 2:6
```

12.3 Tara Scripts

Egress Interface – standard Intserv script:

```
#! /bin/sh

# Path definition:
TC=/QoS/iproute2/tc/tc

# Set up of the proper qdisc on eth1 (classical IntServ interface)
$TC qdisc add root dev eth1 handle 1: cbq bandwidth 100Mbit allot 1514 cell
8 avpkt 1000 mpu 64
# Create root class
$TC class add dev eth1 parent 1:0 classid :1 est 100msec 800msec cbq \
bandwidth 100Mbit rate 100Mbit allot 1514 cell 8 weight 10Mbit prio 8 \
maxburst 250 avpkt 1000
# BE class
$TC class add dev eth1 parent 1:1 classid :2 est 100msec 800msec cbq
bandwidth 100Mbit rate 5Mbit \
allot 1514 cell 8 weight 500kbit prio 6 maxburst 250 avpkt 1000 split 1:0
defmap 0xff

$TC qdisc add dev eth1 parent 1:2 sfq quantum 1514b perturb 15
# Interactive-burst class
$TC class add dev eth1 parent 1:1 classid :3 est 200msec 1600msec cbq
bandwidth 100Mbit rate 1Mbit \
allot 1514 cell 8 weight 1Mbit prio 2 maxburst 100 avpkt 1000 split 1:0
defmap 0xff00
$TC qdisc add dev eth1 parent 1:3 sfq quantum 1514b perturb 15
# RSVP class
$TC class add dev eth1 parent 1:1 classid 1:7FFE cbq bandwidth 100Mbit rate
4Mbit allot 1514b \
avpkt 1000 weight 4Mbit prio 1 maxburst 100 cell 8

# Reclassified RSVP traffic
$TC class add dev eth1 parent 1:7FFE classid 1:7FFF  est 400msec 3200msec
cbq rate 5Mbit bandwidth 100Mbit allot 1514b avpkt 1000 weight 10Kbit \
prio 6 maxburst 10 cell 8 split 1:7FFE defmap 0xffff
```

Ingress Interface – Diffserv Interior router script:

```
#! /bin/sh
################################################
#                                              #
# This script implements a DS interior router #
# It adds to eth0 a proper qdisc               #
# which implements AF1,AF2 PHBs and BE         #
#                                              #
################################################

##### Path definition:
TC=/QoS/iproute2/tc/tc

##### Set up of the proper qdisc on eth0

# First a DSMARK qdisc is introduced in order to retrieve TOS from IP header
$TC qdisc add dev eth0 handle 1:0 root dsmark indices 64 set_tc_index

$TC filter add dev eth0 parent 1:0 protocol ip prio 4 tcindex mask 0xfc
shift 2 pass_on

# Second a CBQ qdisc is used in order to support AF and BE classes
$TC qdisc add dev eth0 parent 1:0 handle 2:0 cbq bandwidth 100Mbit cell 8
avpkt 1000 mpu 64

$TC filter add dev eth0 parent 2:0 protocol ip prio 1 tcindex mask 0xf0
shift 4 pass_on

##### Definition of the CBQ leaf classes to support AF1, AF2 and BE

##### AF Class 1 specific setup
$TC class add dev eth0 parent 2:0 classid 2:1 cbq bandwidth 100Mbit rate
3Mbit \avpkt 1000 prio 4 allot 1514 weight 250Kbit maxburst 21 defmap 0
borrow

$TC filter add dev eth0 parent 2:0 protocol ip prio 1 handle 1 tcindex
classid 2:1

$TC qdisc add dev eth0 parent 2:1 gred setup DPs 3 default 2 grio

# --- AF Class 1 DP 1---
$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 10 tcindex
classid 1:111

$TC qdisc add dev eth0 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 100Mbit DP 1 probability 0.02 prio 2
# --- AF Class 1 DP 2---
$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 12 tcindex
classid 1:112
$TC qdisc add dev eth0 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 100Mbit DP 2 probability 0.04 prio 3
#  -- AF Class 1 DP 3---
$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 14 tcindex
classid 1:113
$TC qdisc add dev eth0 parent 2:1 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 100Mbit DP 3 probability 0.06 prio 4
```

```
##### AF Class 2 specific setup
$TC class add dev eth0 parent 2:0 classid 2:2 cbq bandwidth 100Mbit rate
2Mbit \avpkt 1000 prio 5 allot 1514 weight 250Kbit maxburst 21 defmap 0
borrow

$TC filter add dev eth0 parent 2:0 protocol ip prio 1 handle 2 tcindex
classid 2:2

$TC qdisc add dev eth0 parent 2:2 gred setup DPs 3 default 2 grio
# --- AF Class 2 DP 1---
$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 18 tcindex
classid 1:121

$TC qdisc add dev eth0 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 100Mbit  DP 1 probability 0.02 prio 2

 # --- AF Class 2 DP 2---
$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 20 tcindex
classid 1:122

$TC qdisc add dev eth0 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 100Mbit DP 2 probability 0.04 prio 3

# -- AF Class 2 DP 3---
$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 22 tcindex
classid 1:123
$TC qdisc add dev eth0 parent 2:2 gred limit 60KB min 15KB max 45KB burst 20
avpkt 1000 bandwidth 100Mbit DP 3 probability 0.06 prio 4

##### BE class specific setup
$TC class add dev eth0 parent 2:0 classid 2:6 cbq bandwidth 100Mbit rate
5Mbit \avpkt 1000 prio 7 allot 1514 weight 30Kbit maxburst 21 bounded
isolated

$TC qdisc add dev eth0 parent 2:6 red limit 60KB min 15KB max 45KB burst 20
\
avpkt 1000 bandwidth 100Mbit probability 0.4

$TC filter add dev eth0 parent 1:0 protocol ip prio 1 handle 0x0 tcindex
classid 1:161

$TC filter add dev eth0 parent 2:0 protocol ip prio 1 handle 6 tcindex
classid 2:6
```

[13] APPENDIX B

While testing the ISI RSVP daemon `rel4.2.a4 – 1` some unexpected and quite strange run time errors have been encountered. The result of the rsvpd log file was:

```
rsvpd.log file:
-----------------------------------------------------------
11:42:05.460  Physical, Virtual, and API interfaces are:
11:42:05.461    0 eth0                                          195.169.97.194/30
TCup UDP Police M
11:42:05.462    1 eth1                                          195.169.97.209/28
NoIS M
11:42:05.463    2 API                                                  0.0.0.0/0
NoIS
11:42:17.431|   API  Reg   *L          willow.wmrtb.net/5000[17]  <API pid=973
Asid=1
11:42:57.078|   API  Reg   *L          willow.wmrtb.net/5000[17]  <API pid=973
Asid=1
11:42:57.080|  Rcv API  PATH      willow.wmrtb.net/5000[17] <API ttl=/63
PATH: Sess: willow.wmrtb.net/5000[17]    R: 30000   PHOP: <(API) LIH=0>
         marcie.wmrtb.net/5000  T=[100(200) 200B/s 0 200]
                 Adspec( 0 hop InfBW 0us 65535B, G={br!}, CL={br!})

11:42:57.089|    Snd  Raw    PATH         willow.wmrtb.net/5000[17]  0=>eth0  >
willow.wmrtb.net/62
PATH: Sess: willow.wmrtb.net/5000[17]     R: 30000    PHOP: <marcie.wmrtb.net
LIH=0>
marcie.wmrtb.net/5000  T=[100(200) 200B/s 0 200]
                 Adspec(  1  hop  655KBW  48us  1500B,  G={br!757  35807  757
35807}, CL={br!})

11:43:18.577|   Rcv  Raw   RESV          willow.wmrtb.net/5000[17]  eth0<=0  <
willow1.wmrtb.ne/63
RESV: Sess: willow.wmrtb.net/5000[17]     R: 30000   NHOP: <willow1.wmrtb.net
LIH=0>
FF   marcie.wmrtb.net/5000    [CL T=[100(200) 200B/s 0 200] ]

11:43:18.632  : errno 1071644672
11:43:18.634  : errno 1072168960
11:43:18.634  : errno 1072431104
11:43:18.634  : errno 1072562176
11:43:18.634  : errno 1072627712
11:43:18.635  : errno 1072660480
11:43:18.635  : errno 1072676864
11:43:18.635  : errno 1072685056
11:43:18.635  : errno 1072689152
11:43:18.635  : errno 1072691200
11:43:18.635  : errno 1072692224
11:43:18.635  : errno 1072692736
11:43:18.636  : errno 1072692992
11:43:18.636  : errno 1072693120
11:43:18.636  : errno 1072693184
11:43:18.636  : errno 1072693216
11:43:18.636  : errno 1072693232
```

```
11:43:18.636  : errno 1072693240
11:43:18.636  : errno 1072693244
11:43:18.636  : errno 1072693246
11:43:18.636  : errno 1072693247
----------------------------------------
```

The line I got in the konsole:

```
rsvpd: rsvp_debug.c:460: log: Assertion `severity == 0 || severity >= 3'
failed.Aborted (core dumped)
 ----------------------------------------
```

Later while debugging it was found (with the help of Gabor Retvari) that this error was related to the math.h log() function, because once this function in rsvp_specs.c file:

```
double
math_log(double x)
{
      return(log(x));
//    return(x);
}
```

was replaced with returning x and not log(x)

```
double
math_log(double x)
{
      return(x);
}
```

the errors disappeared.

Bob Lindell from ISI suggested maybe somewhere in the code "the rsvp debug function log() is being called when it really wants the math.h log() function". However, this remains as part of future work activities.