Università degli Studi di Palermo

Facoltà di Ingegneria

Dipartimento di Ingegneria Elettrica

Dottorato di Ricerca in Ingegneria Elettronica, Informatica e delle Telecomunicazioni, XVI Ciclo

Network Edge Support for QoS-aware Applications

Author: Vincenzo Mancuso

Tutor: Prof. Giuseppe Bianchi

A tutti quelli che sanno
regalare un sorriso

Index

Index iii
Introduction 1
Chapter I 5
IP network architectures for QoS 5
§1.1 IP network models 7
§1.1.1 Stateless and stateful QoS approaches 9
§1.2 DiffServ 11
§1.2.1 DiffServ field definition (DSCP) 11
§1.2.2 Differentiated services components 11
§1.2.3 Per Hop Behaviors (PHB) 13
§1.2.3.1 Default PHB 14
§1.2.3.2 Class-Selector PHB 14
§1.2.3.3 Assured Forwarding PHB 14
§1.2.3.4 Expedited Forwarding PHB 15
§1.2.4 DiffServ PDBs 16
§1.2.4.1 Best Effort (BE) Per Domain Behavior 18
§1.2.4.2 The Virtual Wire (VW) PDB 18
§1.2.4.3 An Assurated Rate (AR) 19
§1.2.4.4 An one-to-any Assurated Rate PDB 19
§1.2.4.5 A Lower Effort (LE) PDB 19
§1.3 IntServ 20
§1.3.1 Integrated Service Architecture 21
§1.3.1.1 IntServ/RSVP QoS-related parameters 24
§1.3.2 Guaranteed Service 25
§1.3.3 Controlled Load Service 27
§1.3.4 Resource Reservation Protocol (RSVP) 27
§1.3.4.1 Path messages 28
§1.3.4.1.1 AdSpec 30
§1.3.4.2 Reservation via OPWA 32
§1.3.4.3 Reservation styles and merging 34
§1.3.4.3.1 Fixed filter (FF) 35
§1.3.4.3.2 Wildcard filter (WF) 35
§1.3.4.3.3 Shared explicit (SE) 36
§1.3.4.4 The slack term 37
§1.4 Conclusions 38
Chapter II 40
Traffic control issues 40
§2.1 Admission Control Schemes 41
§2.1.1 Parameter Based Admission Control (PBAC) 43
§2.1.2 Measurement Based Admission Control (MBAC) 44
§2.1.3 Endpoint Admission Control 46
§2.1.3.1 EAC in DiffServ 48
§2.2 The effect of MBAC on Self-similarity 48
§2.2.1 Self-similarity and LRD in network traffic 49
§2.2.2 MBAC vs. PBAC implementations 51
§2.2.3 The simulation scenario 52

§2.2.3.1 Traffic sources 54
§2.2.3.2 Adopted MBAC algorithm 55
§2.2.3.3 Statistical analysis of self-similarity 55
§2.2.4 Performance evaluation 57
§2.3 Conclusions 65
Chapter III 67
Support for multiple-levels QoS-needy unicast applications in DiffServ networks 67
§3.1 GRIP 68
§3.1.1 End Point Operation 70
§3.1.2 Router Operation 70
§3.2 GRIP over AF DiffServ 71
§3.2.1 DiffServ semantic and GRIP 71
§3.2.2 GRIP AF Router Operation 73
§3.2.3 Possible roles of the AFx3 level 75
§3.2.4 Security considerations 76
§3.3 The degree of QoS support in GRIP 76
§3.3.1 GRIP over pure DiffServ (RED/RIO queues) 79
§3.3.1.1 RED/RIO queue management 79
§3.3.1.2 Weighted RED (WRED) 82
§3.3.1.3 RIO (RED with In and Out) 82
§3.3.2 Performance of GRIP over legacy AF PHB (with RED/RIO) 83
§3.3.3 Performance of GRIP over MBAC-aware AF PHB 88
§3.3.3.1 Simulated performance 89
§3.4 API for network traffic regulation 94
§3.4.1 Characteristics of the API 94
§3.4.2 Packet dropping mechanisms for distributed AC support 97
§3.4.3 The packet-level traffic control API 97
§3.4.3.1 Examples of traffic control schemes implementations 101
§3.5 Performance of GRIP over a real test bed 103
§3.5.1 The real test bed and relative GRIP performance 104
§3.6 Conclusions 107
Chapter IV 109
DiffServ/GRIP QoS-aware extensions for signaling interoperability and multicast 109
§4.1 Signaling interoperability 110
§4.1.1 The use of GRIP for signaling interoperation purposes 111
§4.1.2 Edge-to-edge and cross-domain implicit signaling operation 114
§4.1.2.1 Explicit-to-Implicit signaling mapping 116
§4.1.2.2 Implicit-to-Implicit signaling mapping 118
§4.1.3 Deployment considerations 119
§4.1.4 RSVP transparent signaling across GRIP-DiffServ domains 120
§4.1.4.1 RSVP over GRIP 120
§4.1.4.2 RSVP plus GRIP 122
§4.1.4.3 Service mapping 124
§4.2 The multicast case 125
§4.2.1 Multicast applications in DiffServ network and related work 126
§4.2.2 The PIM-SM multicast protocol and QoS extensions 128
§4.2.3 Integration of GRIP-like operation 131
§4.2.4 A real test bed implementation 135
§4.3 Conclusions 140

Chapter V 142
Streaming and interactive traffic in Vehicular Area Networks 142
§5.1 VAN networking 143
§5.1.1 Proxies and elastic buffers 145
§5.2 Services 148
§5.2.1 VoD Services 150
§5.2.2 DRT Services 151
§5.2.3 Adaptation of elastic buffering to moving networks 152
§5.2.3.1 The All-To-Min resource sharing algorithm 153
§5.2.4 Adaptation to Delayed Real Time streaming services 157
§5.3 Simulative approach 161
§5.4 Performance evaluation 163
§5.4.1 Video-on-Demand services 165
§5.4.2 Further results for VoD-like services 167
§5.4.3 Delayed Real Time services 173
§5.4.4 FIFTH Train scenario: Italian railways 175
§5.4.5 Mixed VoD and DRT services evaluation 177
§5.5 Conclusions 181
Chapter VI 182
QoS using interactive DVB satellite systems 182
§6.1 DiffServ and IntServ in a satellite DVB-RCS system 183
§6.1.1 IntServ service model 187
§6.1.2 DiffServ service model 188
§6.1.3 DVB-RCS service model 188
§6.2 QoS-aware interworking of DiffServ, IntServ and DVB system 189
§6.2.1 QoS-related mapping between IntServ/RSVP and DVB-RCS 190
§6.2.2 QoS-related mapping between DiffServ/GRIP and DVB-RCS 192
§6.2.3 Interoperation of DiffServ and DVB-RCS satellite networks 195
§6.2.3.1 Gateway/hub/NCC side 196
§6.2.3.2 Satellite Terminal side 202
§6.2.3.3 Control-plane interworking 211
§6.3 A concrete example of cross-layer interoperation 213
§6.4 Conclusions 217
Conclusive remarks 219
Bibliographic References 222
List of Acronyms 228

Introduction

The interconnection of small, medium and large packet data networks in a world wide Internet represents today's faster and more broadly polyvalent service factory. Internet, based on IP networking technologies, offers the possibility to interconnect communication areas, where multiple technologies are adopted for the local connectivity and for the backbone networking elements. The IP protocol is the *Solution* for heterogeneous networks to interoperate, no matter if some users adopt wirelines and some others use wireless technologies, no matter if point-to-point communication mechanisms are devised to serve small or wide areas and rely on cable, optical fiber or radio transmission media. IP has revealed to be suitable also for satellite communications, both in the case of geostationary orbits and when satellites move due to low or medium earth orbits adopted, where satellites often fly from pole to pole. IP has been also proposed for interplanetary data transmission. Thus, every category of users and service providers could potentially exploit Internet features. The point is that services require network resources, and network resources are limited. So, in order to actually deploy services over the Internet, it rises the necessity to have resource enough, and, in turn, to make the best with any available resource. In particular, as the load offered to the network grows up to a moderately level, it arises the problem to protect some applications whose data are more sensitive to impairments like delay and packet loss due to the statistical multiplexing adopted in each network element. In practice, the overall Internet can be seen as an enormous network of queues where packets are served with different rates and policies.

Network nodes deal with a variety of applications which generate a multiform traffic. The basic Internet applications were meant to support slow rate data transfer as in the case of e-mail and remote terminal control. Web browsing was limited to very simple on-line pages, with some text and few or not images. The file transfer was considered a heavy load application requiring long time to be completed, so that it was recommended to operate it out of the working hours, i.e., during the night. Luckily, today's network and user resources are enough to transfer tens of gigabytes in a few minutes or in tens of minutes also for private users paying for DSL or cable connections. Unluckily, the amount of traffic generated by users is not limited to e-mail and file transfer applications. Mostly relevant, the degree of interaction with the network is well above the remote terminal and web browsing applications. In fact, the evolution of the concept of interaction between users and between users and network is now proposing a full immersive multimedia exchange based on several interrelated data flows. Example of market relevant applications are the delivery of multimedia on-demand, the multicasting and broadcasting of real time audio and video to complement, enhance or also replace the traditional mass media operation. Modern telecommunication systems also converges to IP services in order to extend the almost traditional cellular telephony to a general-purpose wireless multimedia, including packet network facilities. A great number of these applications is resource consuming and, mostly important, they generate traffic not reactive to network congestions. This is the case of UDP-based streaming protocols that try to send data without adapting to the network load, no matter if every packets are lost because of network queues overflow. It is clear that any not reactive flow cannot be under the network control *after* it starts, but it should be stopped *before* to enter the network, if available resource are not enough to sustain the additional load offered by the new flow.

In the framework depicted above, it is worth considering the risk that the impressive amount of available network resources could result insufficient to satisfy today's services requirements. It seems that the more room is allotted to host applications in a network, the more heavy is the load requested by users and the more stringent are the requisites of quality associated with new applications. It is like a fractal process, where a figure recursively replicates its own parts while expanding in the space and complicating the resulting drawing. Hence it is needed to provide the network with some tools that can help to avoid to definitely lose the control of traffic dynamic behavior. Obviously, a full control of the Internet is out of reach for everyone. This is due to the multiple technologies adopted, and also to the broad variety of network providers that can be found all over the Internet. However, even though Internet were managed by a single entity, it would be impossible to manage all the information related to every users, every flows and to do it all over the time. Conversely, some partial solutions are possible, and in this thesis we will focus on a networking design which allows network providers to offer services with a suitable level of quality, without requiring a tight control of the overall network, but enforcing the majority of quality-enabling mechanisms at the edge of the network. Here, the term edge is related to the border between areas managed by separate providers, and also to the borderline between users and networks, i.e., in that part of data networks which is called access network. The complexity of our approach is quite limited in space: we look for same scalable mechanisms that can be enforced in the overall network without requiring a substantial and sudden change in devices and technologies already adopted. Instead, the complexity of the proposed approaches is concentrate in edge nodes, which could be more easily changed or updated without excessive costs.

Since we want to support different kind of services with deeply different requirements, we need to know something about the application and the traffic generated by users, without forcing inefficient and heavy signaling exchanges all over the network. Heavy signaling exchanges should be limited to access networks, where the number of concurrent connections is almost limited, while the core of the network should be as much as possible free of stateful per-flow mechanisms that requires the instantiation of a finite state machine for each flow. We feel that quality of service aware applications should be supported by opportune network services, whose intelligence is bound as much as possible to the network edge nodes. However, the role of core network nodes is not negligible at all. In fact, we will see that edge nodes need at least some implicit form of signaling conveyed from the core routers along a communication path. Thus we will show that the mechanisms that enables quality support can be placed at network edge, while the effectiveness of such mechanisms is strongly dependent on the degree of interoperation between edge and core nodes. We will also demonstrate how an implicit form of collaboration between edge and core of the networks is possible without introducing any complex tools and without causing a revolution in the network infrastructures. Hence, we will deal with several kind of networks and technologies, aiming at proposing a sort of paradigm for the edge-based control of network resources as well as of the access to network resources and their utilization.

In the first chapter of the thesis, an overview of network models is proposed. In particular we present the already implemented network model with its “best effort” behavior for every kind of applications. In the same chapter we present two opposite visions that was proposed in the last years in order to enhance the degree of service that networks can support. They are the IP Differentiated Service architecture, and the IP Integrated Service architecture. Pros and cons of these approaches will be presented, together with a comparative performance study. In Chapter II to Chapter IV we present some solutions to the problem of resource sharing in a network with finite resources. The aim of those chapters is to deal with the support that network can offer to applications by means of shrewd admission control mechanisms. In particular, Chapter II introduces to the admission control theory and proposes some insights in the family of parameter based and measurement based admission control. The measurement based features are deeper investigated, and quite unexpected capabilities are shown that make measurement based admission control preferable to parameter based schemes, especially when heavy burstiness arises in the offered traffic, as in the case of traffic aggregates exhibiting a self-similar behavior. Chapter III proposes the GRIP paradigm for end-to-end, or edge-to-edge, admission control support, which capabilities will be enhanced if core routers are able to convey implicit signaling by means of local discarding of packets that carry admission requests. GRIP is a probe based tool that can be deployed in already existent DiffServ routers implementing the “assured forwarding per hop behavior”, even though some modifications are required inside routers to enforce a sort of local measurement based admission control. Simulated performance results and real implementation schemes are eventually proposed for GRIP over DiffServ. Chapter IV proposes two classes of GRIP extensions to multicast applications and to signaling interoperations between stateless and stateful network domains. The case of RSVP signaling extension over DiffServ domain is presented as a particular and concrete example. Chapter V uses the example of vehicular area networks, where a proxy operates as the edge node of a mobile network, thus allowing vehicular customer to access Internet facilities and real time broadcast services. In that scenario, we will focus on the features that edge nodes need to implement in order to control the service level of the already admitted connections. The point is that once a network provider wants to provide users with services, it has to honor some service requirements upon a user accesses the service. That chapter mostly concerns the advantages offered by the adoption of proxy servers able to elastically buffer user data while decoupling data retrieval and data delivery operations. The effectiveness of the approach also depends on the way a proxy enforces a bandwidth sharing scheme for the management of the active connections. The last chapter of this thesis (Chapter VI) finally introduces to issues related to the optimization of quality of service support that can be obtained by harmonizing network layer protocols with underlying MAC protocols. Dealing with the concrete example of a DVB-RCS satellite network, in Chapter VI it is shown how IntServ and DiffServ architectures can collaborate with each other and with the MAC layer capabilities offered by the DVB-RCS system. A conclusive example will show the possibility to map IntServ services and RSVP messages to DiffServ “per hop behaviors” by means of GRIP. Furthermore, GRIP is proposed to run in a DiffServ framework with measurement based admission control facilities, and a smart mapping of DiffServ classes and GRIP messages on DVB-RCS Profile Classes is drawn. Finally, some short conclusive remarks complete the thesis.

Chapter I

IP network architectures for QoS

Modern data networks have to support manifold types of traffic over network links shared by multiple kind of applications. Each different application demands different treatments from the network, and some customers are ready to pay extra for preferential treatment of their traffic. Unfortunately, they cannot have separate network connections for each of the customers. Therefore much of the bulk of network traffic has to flow through lines where "first class" traffic and other classes of traffic have to share the bandwidth. The network provider can only differentiate the service at active network elements where the traffic flows through. Hence, there is the necessity to design networks which can deliver multiple classes of service, so that networks should be aware of Quality of Service (QoS) issues. New networks should also be scalable, so that user traffic can increase without affecting network performance. Moreover, data networks should support emerging intensive and mission critical applications which will be the key determinant of a service providers success.

QoS is the capability of a network to provide better service to selected network traffic over various underlying technologies like ATM, IP, MPLS, and also Layer 2 standards like ethernet, WiFi, WiMax, DVB and many others. In other words, it is that feature of the network by which it can differentiate between different classes of traffic and treat them differently. QoS in a entire network involves capabilities in the endpoint software running on a user terminal, and in the networks that carry the data being sent back and forth from one endpoint to another. Several parameters can be metered in order to quantify the QoS level, such as:

- delay: the time interval experienced by a packet to cross a network path;
- delay jitter: the variation in the delay encountered by similar packets following the same route through the network;
- throughput: the neat rate at which packets go through the network; it can be considered from different points of view, according to the protocols which the measure is referred to[1];

[1] For instance, IP throughput is computed on the basis of the total number of bit per second transmitted in IP datagrams, while the TCP throughput solely takes into account data segments correctly delivered, without considering packet retransmitted due to data loss or transmission errors.

- loss rate: the rate at which packets are dropped, get lost or become corrupted during the transmission through the network route;
- service availability: the reliability of the offered connection to network services.

Any network design should try to maximize the service availability and, mostly important, the throughput, while minimizing the transmission delay and reducing as much as possible the jitter and the loss occurrences.

In this first chapter we shall be mostly discussing about network capabilities required to support QoS, rather than endpoint features devoted to augment network performance. A thorough analysis of endpoint-driven network operations for QoS purposes will be treated in following chapters. Hence, this first chapter offers an introduction to IP network architectures that have been proposed in order to enhance and differentiate the QoS experienced by network users. These architectures also offer to network administrators and service providers a set of tools well suited to administrate and to manage the network and its resources in a QoS-aware fashion. In particular two approaches are presented in following sections: the Differentiated Service architecture (DiffServ) and the Integrated Services architecture (IntServ). They were both proposed by the IETF, and their specifications are given in public domain RFC documents and IETF standards, that contain the guideline for a correct architectural implementation. It is worth noting that a number of implementation details still remain available for customized deployments, and each network provider has to take care of the appropriate design of such parameters, also tacking into account the set of services that it wants to provide and traffic engineering related factors.

The new architectures are meant to cope with QoS issues while the existing IP networks, and specially today's Internet, are founded on the Best Effort principle: hence, they are not prepared to deal with Quality of Service considerations. Conversely, the Best Effort model maximizes network resource utilization, while keeping at a minimum the complexity of network devices.

In what follows, section §1.1 introduces to today's IP network models and possible enhancement strategies, that are deeply treated in the two following sections: section §1.2 gives an insight in the Differentiated Services architecture, while section n §1.3 deals with the Integrated Services model. Eventually, section §1.4 very briefly concludes the chapter.

§1.1 IP network models

IP is now the foundation for a universal networking of multiple technologies and standards. It is a widely spread - almost ubiquitously diffused - protocol, because it is simple, flexible and powerful. However, as a consequence of its simplicity, especially as regards management issues, IP has not native means to control the quality of the service that can be offered through its networks. As such a broad IP diffusion has grown, it also has arisen the necessity to endow IP with QoS-aware mechanisms, in order to applications receive adequate treatment across the network. The problem with IP is that, like currently adopted forms of ethernet, it is a connectionless technology and does not guarantee bandwidth. Specifically, the IP protocol will not differentiate network traffic based on the type of flow to ensure that the proper amount of bandwidth and prioritization level are defined for a particular type of application. For example, network customers might need a way to ask to the network to provide a low level of latency and packet loss to ensure some performance bound to data flows of particular interest. They may need to ensure that a real time communications such as voice or video over IP be not degraded and look choppy and annoying. Unluckily, currently deployed IP networks are mostly compliant with the Best Effort model. In this model, an application sends data whenever it decides to do it, as much as it feels like and without requiring permission to anyone. On the other hand, the network elements try their mechanisms best to deliver the packets to their destination without any packet discriminations or any bounds on delay, jitter, loss rate and other QoS parameters. As a consequence, a network element could stop attempting to deliver packets if it cannot, and it do it without informing either the sender or the receiver. Network endpoint are ultimately in charge of surveying the packet flow and its correct behavior.

Best Effort is the default service associated with Internet, and it is characterized by a variable service response. In fact, the network makes no attempt to actively differentiate its service response between the traffic generated by concurrent users of the network. Hence, as the load generated by the active traffic flows within the network varies, the Best Effort service response will also vary. Various Internet QoS efforts have been spent with the aim to augmenting this base service with a number of optional service responses. Some form of superior service level has to be provided, or a somewhat predictable service response which is scarcely or not affected at all by the number of concurrent users and active services. Since any network service response is an outcome of the resources available to service a load, and the level of the load itself, service responses can be distinguished by means of a form of loose or tight control of the load admitted into a service, so that the resources allocated by the network to support a particular service are capable of providing the desired response for the imposed load.

QoS provisioning generally encompasses bandwidth allocation, prioritization, and control over network latency. There are several ways to reach the desired performance for a service response, in terms of QoS experienced by network applications. The easiest, and perhaps more expensive, way to augment the QoS level is to increase the available bandwidth until service quality becomes acceptable all over the time. In turn, this approach generally drives to inefficiencies and not needed resources overprovision. However, augmenting bandwidth resources involves the upgrade of physical devices and communication media with higher-speed technologies. Thus, network dimensioning is often performed in accordance to statistical parameters that characterize the user traffic behaviors, and they guarantee a certain QoS in the level of the service response, but only in a statistical fashion. Moreover, this simplistic strategy collapses if a network is even moderately busy, since the variability of the traffic can easily turn in bottlenecks and congestions in some network links and paths. Additionally, simply adding bandwidth does not furnish the sometime needed capability to distinguish high priority traffic flows from lower priority ones. In other words, in a Best Effort environment, all traffic is treated the same. In any case, additional bandwidth can solve some of short-term problems, but it is not a viable long-term solution, especially if the network is expected to accommodate intensive real time application and multimedia streams.

Multiple queues with different priorities could be implemented at network interfaces, so that flows could be accommodated in distinct classes of services. The mechanisms to select the right queue to be adopted may use a flow identification code embedded in the IP header, or based on information like source and destination addresses conveyed by each datagram. This approach requires a shared protocol devised to suitably add a label to each IP datagram, or to diffuse information about flow requirements to allow every routers to filter flows based on a learned flow database. The presence of multiple queue does not imply any QoS guarantees, but it is enough to satisfy some service differentiation requirements. In fact, the traffic pertaining to higher level queues is rather scarcely affected by traffic intended for lower level queues. In any case, not differently from any other queue, the higher priority queue could be overwhelmed by incoming traffic, and network performance remain still poor in severe load conditions.

A smarter solution to QoS related problems, but with higher associated operational load, consists of a reservation style adopted for distinguishing network flows and to bind part of available resources to single data flows. Reservation protocols are meant to let end stations request specific QoS levels for QoS-enabled applications. Request could include information that defines the maximum transmission rate, the maximum delay and an acceptable level of jitter, as well as the maximum loss rate, and so on. When an user makes a reservation request for an application, each router along the data path annotates the QoS request and attempts to honor it. If a router cannot comply with the request, the requesting station receives an error message, possibly conveying the reason of the impossibility to setup a reservation in the network. The user can re-attempt to obtain a reservation after having modified the request conditions or after a suitably long time interval. Obviously, the reservation approach is quite expensive in terms of network management and signaling overhead. Furthermore, as to the service availability, using a reservation approach could also lead to a perceived degradation of some network services. In fact, in a Best Effort scenario, even though an application might run slowly or badly, a somewhat reduced services is provided. Conversely, using reservation, sometime applications might not run at all if there is not enough bandwidth to support them. Note that some applications, like e-mail, are better supported by low rate and variable network services, rather than not constantly available high level services. Similarly, upon high priority flows commit all the available resources, a router have to drop packets belonging to non prioritized flows. Finally, consider the difficulty expressed by reservation protocols in scaling as soon as the network dimensions and the number of potential users grows.

Based on the consideration carried out in this section, and focusing on the capability of the network and of the network protocols to convey and retain information about traffic and flows, the following subparagraph introduces to stateful and stateless networking strategies that will be respectively adopted in the two major IP network architectures proposed to overcome the Best Effort model problems: DiffServ (section §1.2) and IntServ (section §1.3).

§1.1.1 Stateless and stateful QoS approaches

Basically, there are two approaches to network load control, and they can be characterized as stateful and stateless approaches respectively. An example of stateless architecture is represented by the Differentiated Services model, where the packets are marked with a code to trigger the appropriate service response from the network elements that take care of packets. Thus, there is no strict requirement to install a per-reservation state on these network elements. Similarly, the endpoint application or the service requestor is not required to provide the network with advance notice relating to the destination of the traffic, nor any indication of the intended traffic profile or the associated service profile. In the absence of such information any form of per-application or per-path resource reservation is not feasible. In this model there is no maintained per-flow state within the network.

Conversely, the architecture of the Integrated Services model represents an example of stateful approach. In this model, the cumulative sum of honored service requests is equated to the current reserved resource levels of the network. In order for a resource reservation to be honored by the network, the network must maintain some form of permanent state to describe the resources that have been reserved, and the network path over which the reserved service will operate. This is to ensure integrity of the reservation. In addition, each active network element within the network path must maintain a local state that allows incoming IP packets to be correctly classified into a reservation class. This classification allows the packet to be placed into a packet flow context that is associated with an appropriate service response consistent with the original end-to-end service reservation. This local state also extends to the function of metering packets for conformance to a flow-by-flow basis, and the additional overheads associated with maintenance of the state of each of these meters.

The stateless approach to service management is rather approximate in the nature of its outcomes. There is no explicit negotiation between the application signaling of the service request and the network capability to deliver a particular service response. If the network is incapable of meeting the service request, then the request simply will not be honored. In such a situation there is no requirement for the network to inform the application that the request cannot be honored, and it is left to the application to determine if the service has not been delivered. The major attribute of this approach is that it can excellently scale with flows number. If the network is capable of supporting a limited number of discrete service responses, and the routers uses per-packet marking to trigger the service response, then the processor and memory requirements in each router do not increase in proportion to the level of traffic passed through the router. Of course, this approach does introduce some degree of compromise in that the service response is more approximate as seen by the end client, and scaling the number of clients and applications in such an environment may not necessarily result in a highly accurate service response to every application.

The state-based Integrated Services architectural model admits the potential to support greater level of accuracy, and a finer level of granularity on the part of the network to respond to service requests. Each individual service request can be used to generate a reservation state within the network that is intended to prevent the resources associated with the reservation to be reassigned or otherwise preempted to service other reservations or to service Best Effort traffic loads. The state-based model is intended to be exclusionary, where other traffic is displaced in order to meet the target of the reservation.

§1.2 DiffServ

The Differentiated Service (DiffServ) IP architecture was proposed in order to introduce a scalable and flexible service differentiation between IP flows. DiffServ IP networks operate a flow aggregation at domain edges and then resulting macro-flows (or *traffic aggregates*) receive separate treatment inside the DiffServ domain. Each DiffServ node is provided a set of different "behaviors" (Per Hop Behaviors, PHB) to operate in correspondence with different traffic aggregates, thus obtaining a service differentiation in terms of bandwidth, delay performance, latency and packet loss. DiffServ has no knowledge of network state (i.e., it is a stateless network architecture) and service guarantees are not assured. Nonetheless, DiffServ architecture results in a very scalar approach to traffic engineering, and, most importantly, we will show how DiffServ allows network providers to introduce traffic control procedures as well as connection admission control and congestion control procedures. Finally, implementing DiffServ includes the following advantages: reducing the burden on network devices and easily scaling as the network grows; allowing customers to keep any existing Layer 3 prioritization scheme; allowing customers to mix DiffServ-compliant devices with any existing equipment in use; alleviating bottlenecks through efficient management of current corporate network resources.

§1.2.1 DiffServ field definition (DSCP)

A replacement header field, called the DiffServ Code Point (DSCP) field, is defined by Differentiated Services. The DSCP replaces the existing definitions of the IP version 4 type of service (TOS) octet and the IPv6 traffic class octet. More precisely, only six bits are used as the DSCP to select the Per Hop Behavior (PHB) at each interface. A currently unused two-bit field is reserved for explicit congestion notification.

§1.2.2 Differentiated services components

The following components are used to implement DiffServ in real systems:

- Traffic conditioning (policing and shaping). Traffic conditioners perform traffic shaping and policing functions to ensure that traffic entering the DiffServ domain conforms to the rules specified by the Traffic Conditioning Agreement (TCA), and complies with the service provisioning policy of the domain. Traffic conditioning may range from simple code point re-marking to complex policing and shaping operations. Traffic conditioning is performed at the edges of a DiffServ domain. Shapers buffer the traffic stream and increase the delay of a stream to make it compliant with a particular traffic profile. Packets might be discarded if there is lack of buffer resources.
- Packet classification, used to categorize a packet within a specific group. After the packet has been defined (that is, classified), the packet is then accessible for QoS handling on the network. Using packet classification, one can partition network traffic into multiple priority levels or classes of service. Traffic policers and traffic shapers use the traffic descriptor of the packet to ensure adherence to TCA agreement.
- Packet marking, allowing to classify a packet based on a specific traffic descriptor (such as the DSCP value). Markers add a label to packets by setting the DSCP value to a correct code point in its IP header. This classification can then be used to apply user-defined differentiated services to the packet and to associate a packet with a local QoS group. Thus a packet is categorized into a particular behavior aggregate. When a marker changes the label of a packet, then it is said to re-mark the packet. The marker may be configured by various policies.
- Scheduling, or Congestion management, is achieved through traffic scheduling and traffic queueing. When there is network congestion, a scheduling mechanism such as CBWFQ is used to provide guaranteed bandwidth to the different classes of traffic.
- Congestion avoidance. Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through packet dropping. Among the more commonly used congestion avoidance mechanisms is WRED. Specific droppers are adopted, and they also drop packets of a stream to make the stream profile compliant to TCAs.
- Metering. The DiffServ conditioner module receives packets from the classifier and uses a meter to measure the temporal properties of the stream against the appropriate traffic profile from the TCA. Further processing is done by the markers, shapers and policers based on whether the packet is in- or out-of-profile. The meter passes this information to the other components along with the packet.

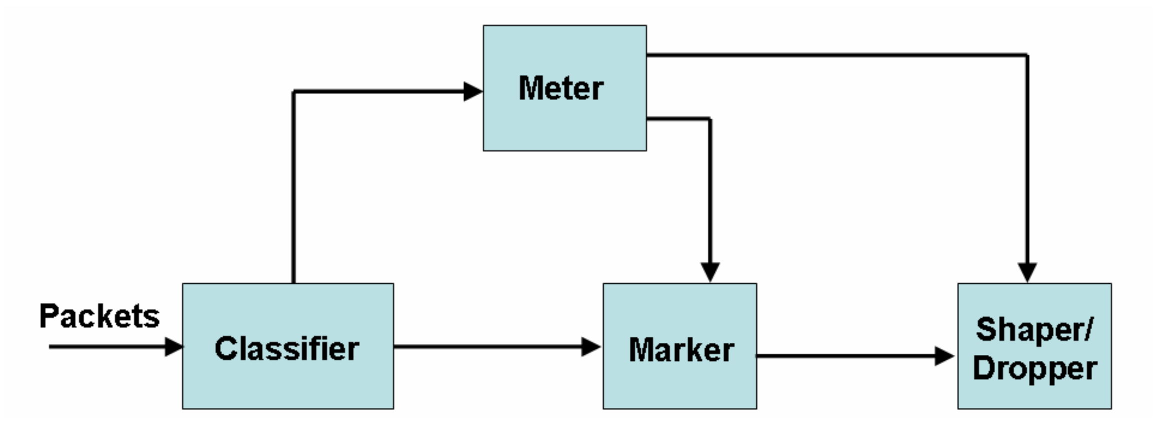


figure 1 – DiffServ components interoperation

§1.2.3 Per Hop Behaviors (PHB)

Out of the existing QoS approaches, the DiffServ requires low additional complexity to be implemented in legacy network node, resulting in a modular, scalable, and incrementally deployable architecture. From the end user point of view, QoS should be supported from end-to-end between any pair of hosts but this goal is not immediately reachable because it requires inter-domain QoS support. Many steps remain on the road to achieve this. The first is to support edge-to-edge or intra-domain QoS between the ingress and the egress of a single network. The DiffServ Working Group concluded a first standardization phase defining the behaviors required in the forwarding path of all network nodes, the Per Hop Behaviors or PHBs. A generalization of this concept to an homogeneous domain (PDB) was defined in a subsequent phase.

The DiffServ philosophy is based on the concept of traffic aggregation. That is, traffic flows are not independently forwarded, but they are aggregate in a certain number of macro-flows, according to their QoS requirements. In order to constitute traffic flow aggregates, packet marking is used. Every IP packet, in a DiffServ Internet, is labeled with a code point, named DSCP that corresponds to the TOS field in legacy IP, but it is regarded in conformance to a new meaning. By this way, a router is able to distinguish between several macro-flows (i.e., traffic aggregates), and to treat each of them according to specified rules. The way to handle a macro-flow (i.e., network behavior), is referred to PHB.

RFC 2475 [14] defines PHB as the externally observable forwarding behavior applied at a DiffServ-compliant node to a DiffServ Behavior Aggregate (BA). With the ability of the system to mark packets according to DSCP setting, collections of packets with the same DSCP setting and sent in a particular direction can be grouped into a BA. Packets from multiple sources or applications can belong to the same BA. In other words, a PHB refers to the packet scheduling, queueing, policing, or shaping behavior of a node on any given packet belonging to a BA, as configured by a service level agreement (SLA) or a policy map. The following sections describe the four available standard PHBs:

- Default PHB (as defined in [75])
- Class-Selector PHB (as defined in [75])
- Assured Forwarding (AFny) PHB (as defined in [50])
- Expedited Forwarding (EF) PHB (as defined in [63])

§1.2.3.1 Default PHB

The default PHB essentially specifies that a packet marked with a DSCP value of '000000' (recommended) receives the traditional Best Effort service from a DiffServ-compliant node (that is, a network node that complies with all of the core DiffServ requirements). Also, if a packet arrives at a DiffServ-compliant node, and the DSCP value is not mapped to any other PHB, the packet will get mapped to the default PHB.

§1.2.3.2 Class-Selector PHB

To preserve backward-compatibility with any IP precedence scheme currently in use on the network, DiffServ has defined a DSCP value in the form xxx000, where x is either 0 or 1. These DSCP values are called Class-Selector Code Points. (The DSCP value for a packet with default PHB 000000 is also called the Class-Selector Code Point). The PHB associated with a Class-Selector Code Point is a Class-Selector PHB. These Class-Selector PHBs retain most of the forwarding behavior as nodes that implement IP Precedence-based classification and forwarding. For example, packets with a DSCP value of 11000 (the equivalent of the IP Precedence-based value of 110) have preferential forwarding treatment (for scheduling, queueing, and so on), as compared to packets with a DSCP value of 100000 (the equivalent of the IP Precedence-based value of 100). These Class-Selector PHBs ensure that DiffServ-compliant nodes can coexist with IP Precedence-based nodes.

§1.2.3.3 Assured Forwarding PHB

AF PHB defines a method by which BAs can be given different forwarding assurances. For example, network traffic can be divided into the following classes:

- Gold: Traffic in this category is allocated P_1 percent of the available bandwidth;

- Silver: Traffic in this category is allocated P_2 percent of the available bandwidth;
- Bronze: Traffic in this category is allocated P_3 percent of the available bandwidth;

where $P_1 + P_2 + P_3 = 100$ and $P_1 > P_2 > P_3$.

Further, the AF PHB defines four AF classes: AF1, AF2, AF3, and AF4. Each class is assigned a specific amount of buffer space and interface bandwidth, according to the SLA with the service provider or policy map. Within each AF class, one can specify three drop precedence values: 1, 2, and 3. Assured Forwarding PHB can be expressed as AFxy: x represents the AF class number (1 to 4) and y represents the drop precedence value (1 to 3) within the AFx class. In instances of network traffic congestion, if packets in a particular AF class (say class x) need to be dropped, packets in the AFx class will be dropped according to the following guideline:

$$(1) \qquad P_d(AFx3) \geq P_d(AFx2) \geq P_d(AFx1)$$

where $P_d(AFxy)$ is the probability that packets of the AFxy class will be dropped. The dropping precedence method penalizes traffic flows within a particular BA that exceed the assigned bandwidth. Packets on these offending flows could be re-marked by a policer to a higher drop precedence.

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

table 1 - Recommended DSCP value and corresponding dropping precedence value for each AF PHB class.

§1.2.3.4 Expedited Forwarding PHB

The EF PHB, is meant to supply a robust service by providing low loss, low latency, low jitter, and assured bandwidth service. EF can be implemented using Priority Queueing, along with rate-limiting on the class (or BA). When implemented in a DiffServ network, EF PHB provides a virtual leased line, or premium service. For optimal efficiency, however, EF PHB should be reserved for only the most critical applications because, in instances of traffic congestion, it is not feasible to treat all or most traffic as high priority. EF PHB is ideally suited for applications such as VoIP that require guaranteed bandwidth, low delay, and low jitter. The recommended DSCP value for EF PHB is '101110'.

§1.2.4 DiffServ PDBs

In order to support QoS in Internet in a scalable fashion the behavior for a group of packets has to be identified and quantified and than to be preserved when these packets are aggregated with other packets, while traversing the Internet. The DiffServ Work Group has decided to use the term Per-Domain Behavior or PDB, to describe the behavior experienced by a particular set of packets as they cross a DiffServ domain. A PDB is characterized by specific metrics that quantify the treatment a set of packets with a particular DSCP, or set of DSCPs, will receive as it crosses a DiffServ domain. By definition, a PDB is "*the expected treatment that an identifiable or target group of packets will receive from edge-to-edge of a DiffServ domain. A particular PHB (or, if applicable, a list of PHBs) and traffic conditioning requirements are associated with each PDB"*[76].

A PDB specifies a forwarding path treatment for a traffic aggregate and, due to the role that particular choices of edge and PHB configuration play in its resulting attributes, it is where the forwarding path and the control-plane interact. Thus a PDB can be considered as the extension of PHB over an entire DiffServ domain.

Each PDB has measurable, quantifiable, attributes that can be used to describe what happens to its packets as they enter and cross the DiffServ domain. These attributes derive from the characteristics of the traffic aggregate that results from application of classification and traffic conditioning during the entry of packets into the DiffServ domain and the forwarding treatment (PHB) the packets get inside the domain. PDB attributes may be absolute or statistical and they may be parameterized by network properties. A wide range of metrics is possible but in general they will be expressed as bounds or percentiles rather than absolute values.

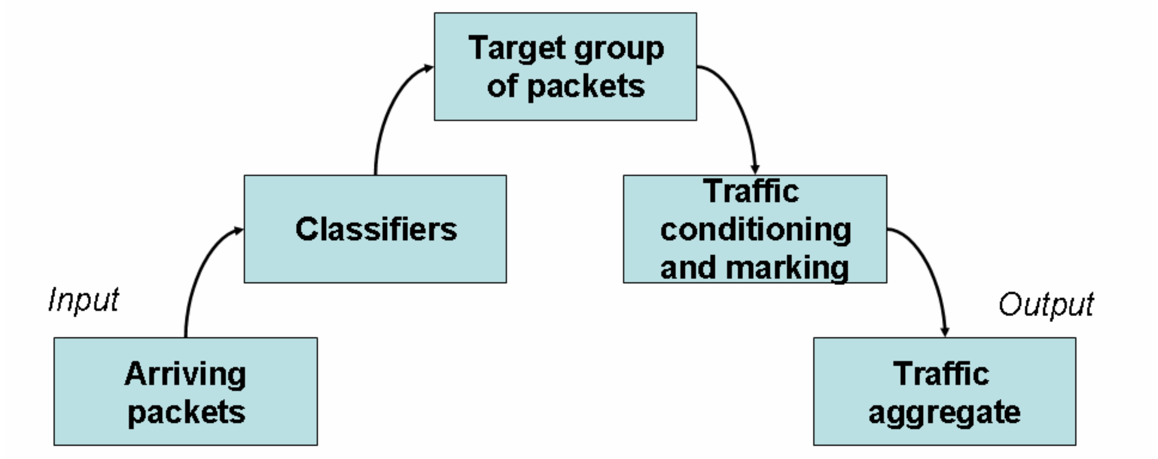


figure 2 - How PDB acts on the arriving packets

The measurable parameters of a PDB should be suitable for use in Service Level Specifications (SLS) at the network edge, thus providing an external description of the edge-to-edge quality of service that can be expected by packets of that PDB within that network.

A PDB is applied to a target group of packets arriving at the edge of the DiffServ domain. This group is distinguished from all arriving packets by use of packet classifiers. The PDB acts on the target group in two parts. The first one is the use of traffic conditioning to create a traffic aggregate. During traffic conditioning, conformant packets are marked with a DSCP for the PHB associated with the PDB. The second part is the treatment experienced by packets from the same traffic aggregate transiting the interior of a DiffServ domain. The following figure describes these two effects.

PDB performance is characterized by metrics relative to the transit of packets between any two edges of the DiffServ domain, since the traffic aggregate is generally formed in the boundary router before the packets are queued and scheduled for output. A PDB does not have to be consistent for arbitrary network topologies, but the limits on the range of applicability for a specific PDB must be clearly specified. PHBs give explicit expressions of the treatment a traffic aggregate can expect at each hop. For a PDB, this behavior must apply to merging and diverging traffic aggregates; thus characterization of a PDB requires understanding what happens to a PHB under aggregation. That is, PHBs recursively applied must result in a known behavior. An advantage of constructing behaviors that aggregate is the ease of concatenating PDBs so that the associated traffic aggregate has known attributes. The use of PHB groups to construct PDBs can be done in several ways. A single PHB member of a PHB group might be used to construct a single PDB. A single PDB can be realized using more than one PHB from the same PHB group. A set of related PDBs might be defined using a PHB group. This is appropriate when the traffic conditioners that build traffic aggregates associated with each PDB have some relationships and interdependencies such that the traffic aggregates for these PDBs should be described and characterized together.

There are long-term (average) and short-term (bursty) PDB attributes. Long term attributes hold in normal situation. Unexpected situation will come out, if the topology of the DiffServ domain suddenly changes and new route must be used. Long-term attributes should be told as the rates or throughput seen over some specified period of time. Short-term attributes are usually expressed as the allowable burstiness in a traffic aggregate. Burst might also accumulate in a large DiffServ domain with many tails. Conditions should be imposed in PDB to avoid this to ensure concatenating of PDBs. A relationship between PHB and PDB can be made using many different ways. Multiple PDBs may use the same PHB. The specification of a PDB can contain a list of PHBs and their required configuration, all of which would result in the same PDB. In operation, it is expected that a single domain will use a single PHB to implement a particular PDB, though different domains may select different PHBs. A single PHB might be selected within a domain by a list of DSCPs. Multiple PDBs might use the same PHB in which case the transit performance of traffic aggregates of these PDBs will be the same. Yet, the particular characteristics that the PDB designer wishes to claim as attributes may vary, so two PDBs that use the same PHB might not be specified with the same list of attributes.

Five PDBs have been defined. One is for Best Effort traffic. Opposite to Best Effort traffic is Low Effort PDB, which is meant for traffic having lower priority than Best Effort. There are two PDBs for Assurated Rate traffic. Virtual Wire PDB is meant for traffic, which needs a behavior of hard-wired circuit.

§1.2.4.1 Best Effort (BE) Per Domain Behavior

A Best Effort (BE) PDB is for sending common Internet traffic across a DiffServ network. That is, the definition and use of this PDB is to preserve the delivery expectation for packets in a DiffServ network that do not require any special differentiation. Packets of this PDB will not be completely starved and when there are resources available this PDB is let to use them. The PDB does not include bounds on availability, latency, and packet loss. Yet providers can use bounds to give the network the quality the want to sell their customers.

§1.2.4.2 The Virtual Wire (VW) PDB

The VW PDB describes a service between edge-to-edge in the domain just identical to dedicated wires between the endpoints. The domain using VW PDB must support DiffServ EF PHB. It is important that there will be almost empty queues. The VW document is meant for ISPs and router builders. The VW PDB is suitable for any packet based on traffic that uses fixed circuits (e.g., telephone, leased data lines) and packet traffic that has similar delivery requirements (e.g., IP telephone or video conferencing). VW PDB can replace some part or all of the physical wire between two points. The DiffServ domain using VW PDB is totally invisible to sender and receiver.

§1.2.4.3 An Assurated Rate (AR)

This AR PDB is meant for traffic that requires rate assurance, but do not require delay and jitter bounds. It is possible that AR PDB traffic will get excess bandwidth beyond the committed information rate, but it is no guarantee of that extra bandwidth. AR PDB is created using the AF PHB with suitable policers at the domain ingress nodes. AR PDB can be used to create services using only TCP or UDP micro-flows. Attributes used for this PDB are a rate that is assured and low drop probability. There is also determined that some parameters must be included to the AR PDB. These parameters are a committed information rate (CIR) that is assured with high probability, traffic parameters that are needed to measure CIR and maximum packet size for the aggregate. The idea is that policers mark the arriving packets with colors to describe the drop precedence (green, yellow and red). The packets with assurance are marked green and these are the one to be dropped the last. The assumption has been made that the network is well provisioned so possibility to drop packets marked with green color is very low. This PDB is applicable for a one-to-one or one-to-few VPN-like services and one-to-any general services.

§1.2.4.4 An one-to-any Assurated Rate PDB

An one-to-any AR PDB is meant for traffic that requires rate assurance, but does not require delay and jitter bounds. This PDB is similar to AR PDB, but determines only the one-to-any case. It is very expensive to assure CIR with almost no drops in this one-to-any case, so this PDB considers the possibility that assurated rate will no be met with a certain probability. This PDB is used in situation, where one ingress point is sending data to any egress point of the DiffServ domain. The example user could be web site that wants to provide its users with high-speed access to its web pages.

§1.2.4.5 A Lower Effort (LE) PDB

A LE PDB is meant for sending non-critical traffic across DiffServ domains. The idea is to delay or drop LE PDB packets, if there are other traffic present. So, LE PDB has lower priority compared to other traffic. It is meant as a tool for operator to protect their networks for selected types of traffics. This LE PDB is meant e.g. for multimedia applications using UDP, Netnews, peer-to-peer traffic and for traffic caused by worldwide web search engines while they gather information from web servers. It is not meant as a PDB to use with customer traffic; but only as a tool to help operator to organize or shape the traffic during the rush hour.

§1.3 IntServ

In response to the growing demand for an integrated services Internet, the Internet Engineering Task Force (IETF) [57] setup an Integrated Services (IntServ) Working Group [58], which has defined several service classes for any offered data flow.

In a IntServ network, a data flow identifies the set of packets to receive special QoS, and it is defined by a *session* comprising the IP address, transport-layer protocol type, and port number of the destination along with a list of specific senders to that session that are entitled to receive the special QoS. Each sender is identified by source address and port number, while its protocol type must be the same as for the session. If supported by the routers traversed by a data flow, different classes for particular QoS commitments can be provided. In contrast, best effort traffic entering a router will receive no such service commitment and will have to make do with whatever resources are available.

The level of QoS provided by these enhanced QoS classes is programmable on a per-flow basis according to requests from the endpoints. These requests can be passed to the routers by network management procedures or, more commonly, using a reservation protocol such as RSVP. The requests describe the level of resources (e.g., bandwidth, buffer space) that must be reserved along with the transmission scheduling behavior that must be installed in the routers to provide the desired end-to-end QoS commitment for the data flow. In determining the resource allocations necessary to satisfy a request, the router needs to take into account the QoS support provided by the link layer in the data forwarding path. Furthermore, in the case of a QoS-active link layer such as asynchronous transfer mode (ATM) or certain types of local area network (LAN), the router is responsible for negotiations with the link layer to ensure that the link layer installs appropriate QoS support should the request be accepted. Each router must apply admission control to requests to ensure that they are only accepted if sufficient local resources are available. In making this check, admission control must consider information supplied by end applications regarding the traffic envelope their data flow will fall within. One of the parameters in the traffic envelope that must be supplied is the maximum datagram size of the data flow, and should this be greater than the maximum transmission unit (MTU) of the link, admission control will reject the request since the integrated services models rely on the assumption that datagrams receiving an enhanced QoS class are never fragmented. Once an appropriate reservation has been installed in each router along the path, the data flow can expect to receive an end-to-end QoS commitment provided no path changes or router failures occur during the lifetime of the flow, and provided the data flow conforms to the traffic envelope supplied in the request. Service-specific policing and traffic reshaping actions are employed within the network to ensure that non-conforming data flows do not affect the QoS commitments for behaving data flows. The IETF has considered various QoS classes such as in [4], although to date only two of these, Guaranteed Service and Controlled-Load Service, have been formally specified for use with RSVP.

§1.3.1 Integrated Service Architecture

As is usual in the Internet, there are a number of options for implementing a function. For IntServ, some information has to be communicated between the end systems and the network elements (routers and subnets). A logical choice for this communication is RSVP, but that is not the only possibility. Some other setup protocol (equivalent in functionality) or management procedures may be used to perform the same functions. figure 3 shows part of the architecture of an integrated services system. Note that the RSVP is only a protocol that carries (among other things) the invocation of some QoS functions. This is communicated to the functions within the router/subnet which actually deliver the QoS. RSVP itself is not aware of the semantics of the objects that it carries. Note that other relationships exist between these types of information than are shown in this picture. For example, the admission control function would use not only policy objects carried by RSVP, but also the objects used to invoke the QoS functions and possibly even the authentication and accounting objects.

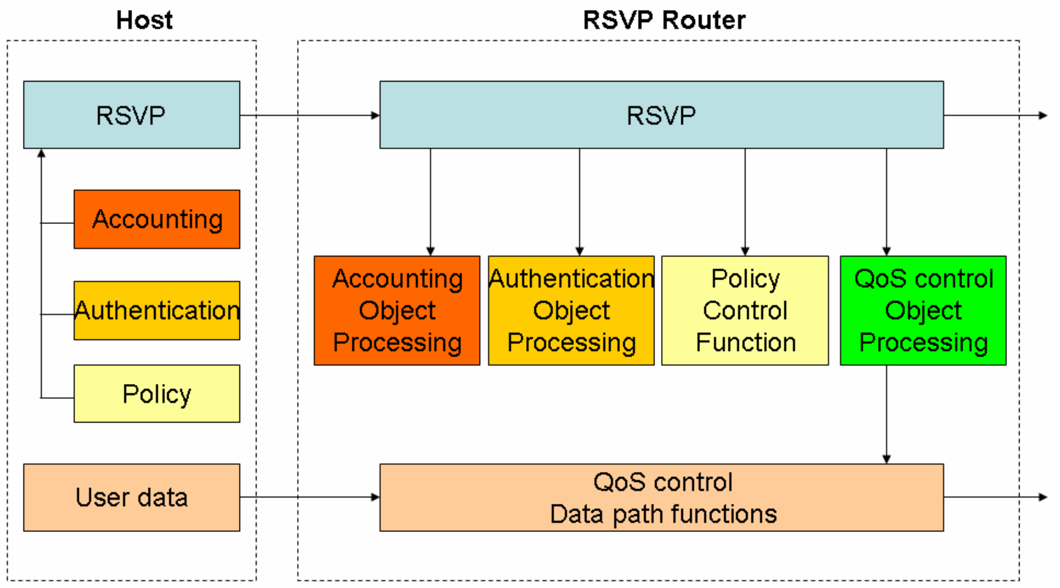

Host
RSVP Router
RSVP
RSVP
Accounting
Authentication
Policy
Accounting Object Processing
Authentication Object Processing
Policy Control Function
QoS control Object Processing
User data
QoS control Data path functions

figure 3 - High level architecture for integrated services

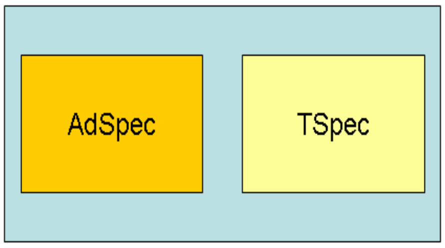


figure 4 - Object in the Path message

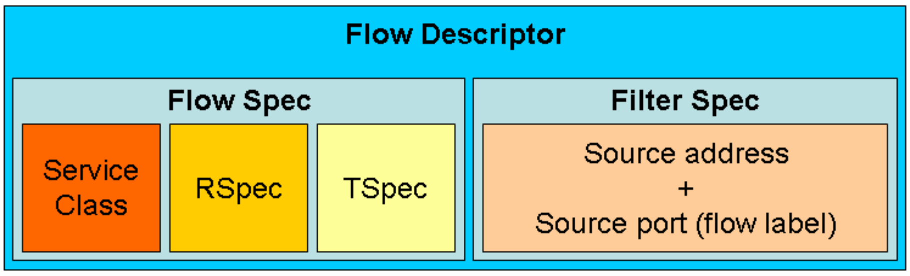


figure 5 - Object in a Flow descriptor

The main point is that the invocation and provision of the QoS functions are different, and further, that RSVP itself does not provide any functions, but only communication services. So, the provision of integrated services is independent of RSVP itself. While acknowledging that integrated services can be setup in various ways, it is nonetheless handy to describe the information needed for the various kinds of service in terms of sets of parameters that (possibly) will flow in RSVP messages. We begin by looking at figure 4 which show some of the data objects found in the Path message. In RSVP, the Path message flows from the sender to the receiver (downstream) and contains information about the traffic stream coming from the sender. The traffic stream itself is described by the TSpec of the sender. The TSpec does not change, but signals to all intermediate nodes and the destination what the characteristics of the stream are. The AdSpec can be originated by the sending host or the first router in the path, and contains a number of general parameters characterizing the path which are determined by hop-by-hop computations, and also parameters characterizing particular QoS services supported by the network elements. Examples of the general parameters are: PathMTU size, path bandwidth, path minimum latency, etc.

figure 5 shows the format of a Flow Descriptor in RSVP. A Flow Descriptor is made up of two parts: the Flow Spec and the Filter Spec. The Flow Descriptor is found in the Resv message which flows from the receiver to the sender (upstream). The Filter Spec is input to the packet classifier which selects sub-streams of the packet stream seen by a particular link which have requested a particular kind of service. The Flow Spec is made up of three parts which state: the type of service requested, the parameters of the service (the RSpec) and the parameters of the flow requesting the service (the TSpec). The Filter Spec serves a purely logical packet discrimination function. Based either on the flow label (in an IPv6 packet) or the combination source address and source port (in an IPv4 packet) a sub-stream of packets can be selected to which the service is applied. The Flow Spec contains the specific values of the parameters that make up the traffic characteristics of the flow and the specific values of the parameters of the service being requested.

There is more detail in the QoS control data path functions, as shown in figure 6. Here we see the information from RSVP being passed to the various functions that must exist in the router in order for QoS to be provided. We see, first of all, packet classifier and packet scheduler functions. The packet classifier determines how the packets will be handled. The packet scheduler applies the particular mechanisms to the packets in order to provide the quality of service requested. Other important functional blocks play a role. The policy control and admission control functions work together to determine whether the particular flow is permitted to request this particular service at this particular time (policy) and whether there exist enough resources in the network element to support the service requested by this flow (admission). The results of the application of these functions will of course influence the operation of the packet classifier. In addition, the RSVP information can be used to influence the routing algorithms that are running in the network element.

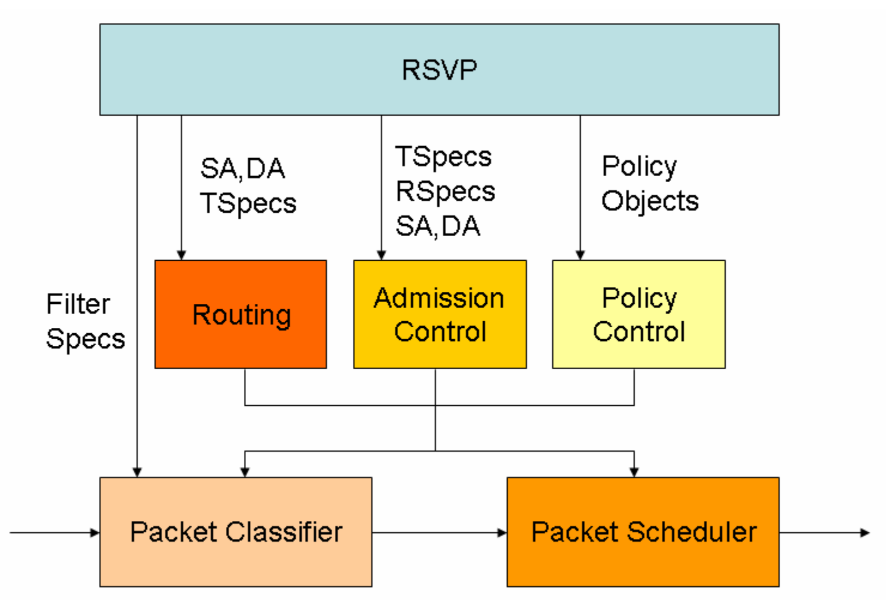


figure 6 - Relationship of RSVP information to router packet handling

§1.3.1.1 IntServ/RSVP QoS-related parameters

The IntServ/RSVP parameters which define the reservation request are the Flow Spec contained in the RSVP Resv message. The Flow Spec includes: i) the requested service class (Guaranteed Service or Controlled Load Service); ii) two set of numeric parameters: a TSpec that describes the data flow and a RSpec which defines the desired QoS when a guaranteed service is requested.

The TSpec parameters include:

- Token Bucket Rate or Sustainable Rate [r]
- Peak Data Rate [p]
- Token Bucket Size [b]
- Minimum Policed Unit [m]
- Maximum Packet Size [M]

The token rate is measured in bytes of IP datagrams per second. Values of this parameter may range from 1 byte per second to 40 terabytes per second. The bucket depth is measured in bytes. Values of this parameter may range from 1 byte to 250 gigabytes. The peak traffic rate is measured in bytes of IP datagrams per second. Values of this parameter may range from 1 byte per second to 40 terabytes per second. The minimum policed unit is an integer measured in bytes. This size includes the application data and all protocol headers at or above the IP level (IP, TCP, UDP, RTP, etc.). The maximum packet size is the biggest packet that will conform to the traffic specification; it is also measured in bytes. Any packets of larger size sent into the network may not receive QoS-controlled service, since they are considered to not meet the traffic specification. Both *m* and *M* must be positive, and $m \le M$.

The RSpec parameters include:

- The Rate *R* selected to obtain bandwidth and delay guarantees. *R* is greater than or equal to *r*. It can be bigger than the TSpec rate because higher rates will reduce queuing delay and it is measured in bytes of IP datagrams per second and has the same range and suggested representation as the bucket and the peak rates.
- The optional Slack Term *S*, which signifies the difference between the desired delay and the delay obtained by using a reservation level *R*. This parameter, when specified, can be used by a network element to reduce its resource reservation for the relevant flow. When a network element chooses to utilize some of the slack in the RSpec, it must follow specific rules in updating the *R* and *S* fields of the RSpec. If at the time of service invocation no

slack is specified, the slack term, *S*, is set to zero. *S* is nonnegative and it is measured in microseconds.

TSpec parameters			**RSpec Parameters**	
p (bytes/s)	Peack rate of flow		R (bytes/s)	Bandwidth
b (bytes)	Bucket depth		S (ms)	Slack term
r (bytes/s)	Token bucket rate			
m (bytes)	Minimum policed unit[2]			
M (bytes)	Maximum datagram size			

table 2 - TSpec and Rspec parameters

§1.3.2 Guaranteed Service

Guaranteed Service provides an assured level of bandwidth, a firm end-to-end delay bound, and no queuing loss for conforming packets of a data flow. It is intended for applications with stringent real time delivery requirements, such as certain audio and video applications that use playout buffers and are intolerant of any datagram arriving after their playout time. Each router characterizes the guaranteed service for a specific flow by allocating a bandwidth, *R*, and buffer space, *B*, that the flow may consume. This is done by approximating the *fluid model* of service [77], [78] so that the flow effectively sees a dedicated wire of bandwidth *R* between source and receiver. In a perfect fluid model, a flow conforming to a token bucket of rate *r* and depth *b* will have its delay bound by *b*/*R* provided $R \geq r$. To allow for deviations from this perfect fluid model in the router approximation[3], two error terms, *C* and *D*, are introduced; consequently, the delay bound now becomes *b/R* + *C/R* + *D*. However, with guaranteed service a limit is imposed on the peak rate, *p*, of the flow, which results in a reduction of the delay bound. In addition, the packetization effect of the flow needs to be taken into account by considering the maximum packet size, *M*. These additional factors result in a more precise bound on the end-to-end queuing delay as follows:

$$\textit{(2)} \qquad Q_{delayE2E} = \frac{(b-M)(p-R)}{R(p-r)} + \frac{M + C_{tot}}{R} + D_{tot} \qquad \textit{if} \;\; p > R \geq r$$

$$\textit{(3)} \qquad Q_{delayE2E} = \frac{M + C_{tot}}{R} + D_{tot} \qquad \textit{if} \;\; R \geq p \geq r$$

[2] Policing will treat any IP datagram less than size *m* as being size *m*.
[3] Among other things, the router approximation must take account of the medium-dependent behavior of the link layer of the data forwarding path.

where C_{tot} and D_{tot} represent the summation of the C and D error terms, respectively, for each router along the end-to-end data path. In order for a router to invoke guaranteed service for a specific data flow, it needs to be informed of the traffic characteristics, TSpec, of the flow along with the reservation characteristics, RSpec. Furthermore, to enable the router to calculate sufficient local resources to guarantee a lossless service requires the terms C_{sum} and D_{sum}, which represent the summation of the C and D error terms, respectively, for each router along the path since the last reshaping point.

Guaranteed service traffic must be policed at the network access points to ensure conformance to the TSpec. The usual enforcement policy is to forward non-conforming packets as best effort datagrams[4]; if and when a marking facility becomes available, these non-conforming datagrams should be marked to ensure that they are treated as best effort datagrams at all subsequent routers. In addition to policing of data flows at the edge of the network, guaranteed service also requires reshaping of traffic to the token bucket of the reserved TSpec at certain points on the distribution tree. Any packets failing the reshaping are treated as best effort and marked accordingly if such a facility is available. Reshaping must be applied at any points where it is possible for a data flow to exceed the reserved TSpec even when all senders associated with the data flow conform to their individual TSpecs. Such an occurrence is possible in the following two cases.

First, at branch points in the distribution tree where the reserved TSpecs of the outgoing branches are not the same, the reserved TSpec of the incoming branch is given by the "maximum" of the reserved TSpecs on each of the outgoing branches. Consequently, some of the outgoing branches will have a reserved TSpec which is less than the reserved TSpec of the incoming branch; so it is possible that, in the absence of reshaping, traffic which conforms to the TSpec of the incoming branch might not conform when routed through to an outgoing branch with a smaller reserved TSpec. As a result, reshaping must be performed at each such outgoing branch to ensure that the traffic is within this smaller reserved TSpec. Second, at merge points in the distribution tree for sources sharing the same reservation, the sum of the TSpecs relating to the incoming branches will be greater than the TSpec reserved on the outgoing branch. Consequently, when multiple incoming branches are each simultaneously active with traffic conforming to their respective TSpecs, it is possible that when this traffic is

[4] Action with regard to non-conforming datagrams should be configurable to allow for situations such as traffic sharing where the preferred action might be to discard nonconforming datagrams. This configuration requirement also applies to reshaping.

merged onto the outgoing branch it will violate the reserved TSpec of the outgoing branch. Hence, reshaping to the reserved TSpec of the outgoing branch is necessary.

§1.3.3 Controlled Load Service

Unlike Guaranteed Service (GS), Controlled Load Service (CLS) provides no firm quantitative guarantees. A TSpec for the flow desiring CLS must be submitted to the router as for the case of GS, although it is not necessary to include the peak rate parameter. If the flow is accepted for CLS, the router makes a commitment to offer the flow a service equivalent to that seen by a best effort flow on a lightly loaded network. The important difference is that the controlled load flow does not noticeably deteriorate as the network load increases. This will be true regardless of the level of load increase. By contrast, a best effort flow would experience progressively worse service (higher delay and loss) as the network load increased. CLS is intended for those classes of applications that can tolerate a certain amount of loss and delay provided it is kept to a reasonable level. Examples of applications in this category include adaptive real time applications. Routers implementing the CLS must check for conformance of controlled load data flows to their appropriate reserved TSpecs. Any non-conforming controlled load data flows must not be allowed to affect the QoS offered to conforming controlled load data flows or to unfairly affect the handling of best-effort traffic. Within these constraints the router should attempt to forward as many of the packets of the non-conforming controlled load data flow as possible. This might be done by dividing the packets into conforming and non-conforming groups and forwarding the non-conforming group on a best effort basis. Alternatively, the router may choose to degrade the QoS of all packets of a non-conforming controlled load data flow equally.

§1.3.4 Resource Reservation Protocol (RSVP)

The Resource Reservation Protocol (RSVP) [18] was designed to enable the senders, receivers, and routers of communication sessions (either multicast or unicast) to communicate with each other in order to setup the necessary router state to support the services described previously. RSVP identifies a communication session by the combination of destination address, transport-layer protocol type, and destination port number. It is important to note that each RSVP operation only applies to packets of a particular session; therefore, every RSVP message must include details of the session to which it applies. In addition, although RSVP is applicable to both unicast and multicast sessions, we concentrate on the more complicated multicast case.

RSVP is not a routing protocol; it is merely used to reserve resources along the existing route setup by whichever underlying routing protocol is in place. figure 7 shows an example of RSVP for a multicast session involving one sender, S1, and three receivers, RCV1 to RCV3. The primary messages used by RSVP are the Path message, which originates from the traffic sender, and the Resv message, which originates from the traffic receivers. The primary roles of the Path message are first to install reverse routing state in each router along the path, and second to provide receivers with information about the characteristics of the sender traffic and end-to-end path so that they can make appropriate reservation requests. The primary role of the Resv message is to carry reservation requests to the routers along the distribution tree between receivers and senders. Returning now to figure 7, as soon as S1 has data to send it begins periodically forwarding RSVP Path messages to the next hop, R1, down the distribution tree. RSVP messages can be transported "raw" within IP datagrams using protocol number 46, although hosts without this raw input/output capability may first encapsulate the RSVP messages within a UDP header.

§1.3.4.1 Path messages

Each Path message includes the following information:

- Phop, the address of the last RSVP-capable node to forward this Path message. This address is updated at every RSVP-capable router along the path.

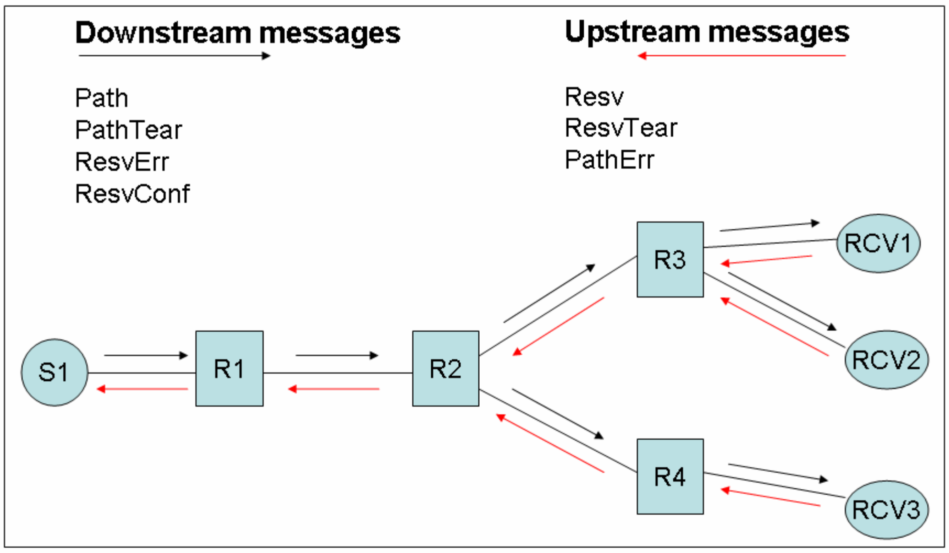


figure 7 - direction of RSVP messages

- The Sender Template, a filter specification identifying the sender. It contains the IP address of the sender and optionally the sender port (in the case of IPv6 a flow label may be used in place of the sender port).
- The Sender TSpec defining the sender traffic characteristics.
- An optional AdSpec containing One Pass With Advertising (OPWA) information which is updated at every RSVP-capable router along the path to attain end-to-end significance before being presented to receivers to enable them to calculate the level of resources that must be reserved to obtain a given end-to-end QoS.

Each intermediate RSVP-capable router along the distribution tree intercepts Path messages and checks them for validity. If an error is detected, the router will drop the Path message and send a PathErr message upstream to inform the sender who can then take appropriate action. Assuming the Path message is valid, the router does the following:

- update the path state entry for the sender identified by the Sender Template. If no path state exists, create it. Path state includes the Sender TSpec, the address, Phop of the previous hop upstream router, and optionally an AdSpec. The Phop address needs to be stored in order to route Resv messages in the reverse direction up the tree. The Sender TSpec provides a ceiling to clip any inadvertently over specified TSpec subsequently received in Resv messages.
- Set cleanup timer equal to cleanup timeout interval and restart timer.

Associated with each path state entry is a cleanup timer, the expiration of which triggers deletion of the path state. Expiration of the timer will be prevented if a Path message for the entry is received at least once every cleanup timeout interval. This is the so-called RSVP soft-state mechanism, which ensures that state automatically timeouts if routing changes while subsequent Path messages install state along the new routing path. In this way, the use of soft-state rather than hard-state helps to maintain much of the robustness of the initial Internet design concepts whereby all flow-related state was restricted to the end systems [28]. The router is also responsible for generating Path messages based on the stored path state and forwarding them down the routing tree, making sure that for each outgoing interface the AdSpec (see the next subsection) and Phop objects are updated accordingly. Path messages will be generated and forwarded whenever RSVP detects any changes to stored path state or is informed by the underlying routing protocol of a change in the set of outgoing interfaces in the data forwarding path. Otherwise, a Path message for each specific path state entry is created and forwarded every refresh period timeout interval in order to refresh downstream path state. The refresh period timeout interval is several times smaller than the cleanup timeout interval so that occasional lost Path messages can be tolerated without triggering unnecessary deletion of path state. However, it is still recommended that a minimum network bandwidth be configured for RSVP messages to protect them from congestion losses. Although all path state would eventually timeout in the absence of any refreshes via Path messages, RSVP includes an additional message, PathTear, to expedite the process. PathTear messages travel across the same path as Path messages and are used to explicitly tear down path state. PathTear messages are generated whenever a path state entry is deleted, so a PathTear message generated by a sender will result in deletion of all downstream path state for that sender. It is recommended that senders do this as soon as they leave the communications session. Also, deletion of any path state entry triggers deletion of any dependent reservation state.

§1.3.4.1.1 AdSpec

The AdSpec is an optional object that the sender may include in its generated Path messages in order to advertise to receivers the characteristics of the end-to-end communications path. This information can be used by receivers to determine the level of reservation required in order to achieve their desired end-to end-QoS. The AdSpec consists of a message header, a Default General Parameters fragment, and at least one of a Guaranteed Service fragment and Controlled Load Service fragment. Omission of either the Guaranteed or Controlled Load Service fragment is an indication to receivers that the omitted service is not available. This feature can be used in a multicast session to force all receivers to select the same service. (At present RSVP does not accommodate heterogeneity of services between receivers within a given multicast session).

The Default General Parameters fragment includes the following fields, which are updated at each RSVP-capable router along the path in order to present end-to-end values to the receivers:

- Minimum path latency (summation of individual link latencies) - This parameter represents the end-to-end latency in the absence of any queuing delay. In the case of guaranteed service, receivers can add this value to the bounded end-to-end queuing delay to obtain the overall bounded end-to-end delay.
- Path bandwidth (minimum of individual link bandwidths along the path)

- Global break bit — This bit is cleared when the AdSpec is created by the sender. Encountering any routers that do not support RSVP will result in this bit being set to one in order to inform the receiver that the AdSpec may be invalid.
- Integrated services hop count — incremented by one at every RSVP/IntServ-capable router along the path.
- PathMTU — path maximum transmission unit (minimum of MTUs of individual links along the path).

Correct functioning of IETF integrated services requires that packets of a data flow to receive the special QoS are never fragmented. This also means that the value of M in the TSpec of a reservation request must never exceed the MTU of any link to which the reservation request applies. A receiver can ensure that this requirement is met by setting the value of M in the TSpec of its reservation request to the minimum of the PathMTU values received in *relevant* Path messages. A Path message is relevant if it originated from a sender that is captured in the intended reservation request in accordance with the reservation styles described later. The value of M in each generated reservation request may be further reduced on the way to each sender if merging of Resv messages occurs.

The minimum value of M from the TSpec of each Resv message[5] received by the sender should then be used by the sending application as the upper limit on the size of packets to receive special QoS. In this way fragmentation of these packets will never occur.

The Guaranteed Service fragment of the AdSpec includes the following fields, which are updated at each RSVP-capable router along the path in order to present end-to-end values to the receivers:

- C_{tot} — end-to-end composed value for C
- D_{tot} — end-to-end composed value for D
- C_{Sum} — composed value for C since last reshaping point
- D_{Sum} —composed value for D since last reshaping point (C_{Sum} and D_{Sum} values are used by reshaping processes at certain points along the distribution tree)
- Guaranteed Service Break bit — This bit is cleared when the AdSpec is created by the sender. Encountering any routers that support RSVP/IntServ but do not support guaranteed service will result in this bit being set to one in order to inform the receiver that the AdSpec may be invalid and the service cannot be guaranteed.

[5] In cases where the last hop to a sender is a shared-medium LAN, the sender may receive Resv messages across the same interface from multiple next-hop router.

- Guaranteed Service General Parameters Headers/Values — These are optional, but if any are included, each one overrides the corresponding value given in the Default General Parameters fragment as far as a receiver wishing to make a guaranteed service reservation is concerned. These override parameters could, for example, be added by routers along the path that have certain service-specific requirements. For example, a router may have been configured by network management so that guaranteed service reservations can only take up a certain amount, *Bgs*, of the outgoing link bandwidth. Consequently, if the Default Path bandwidth value in the AdSpec to be sent out of this interface is greater than *Bgs*, then a Guaranteed Service Specific Path bandwidth header and value equal to *Bgs* may be included in the AdSpec. As for Default General Parameters, any Service-Specific General Parameters must be updated at each RSVP hop.

The Controlled-Load Service fragment of the AdSpec includes the following fields which are updated at each RSVP-capable router along the path in order to present end-to-end values to the receivers.

- Controlled Load Service Break Bit — This bit is cleared when the AdSpec is created by the sender. Encountering any routers that support RSVP/IntServ but do not support CLS will result in this bit being set to one in order to inform the receiver that the AdSpec may be invalid and the service cannot be guaranteed.
- Controlled Load Service General Parameters Headers/Values — As for the Guaranteed Service fragment, override Service-Specific General Parameters may be added to the Controlled Load Service fragment.

§1.3.4.2 Reservation via OPWA

OPWA (One Pass With Advertising) refers to the reservation model for the case where the sender includes an AdSpec in its Path messages to enable the receiver to determine the end-to-end service that will result from a given reservation request. If the sender omits the AdSpec from its Path messages, the reservation model is referred to simply as “One Pass”, in which case there is no easy way for the receiver to determine the resulting end-to-end service. Here we consider the OPWA case. Let us assume that the sender omits the CLS data fragment from the AdSpec, thereby restricting each receiver to reservation of guaranteed service only. Upon receiving Path messages the receiver extracts the following parameters from the Sender TSpec contained therein: r, b, p, m. In addition, the following are extracted from the AdSpec: minimum path latency, C_{tot}, D_{tot}, PathMTU, and path bandwidth. The required bound on end-to-end queuing delay, Q_{delreq}, is now calculated by subtracting the minimum path latency from the value of end-to-end delay required by the receiver application.

Typically, the receiver would then perform an initial check by evaluating (3) for R equal to peak rate p. If the resultant delay was greater than or equal to Q_{delreq}, (3) would be used for calculation of the minimum value of R necessary to satisfy Q_{delreq}; otherwise, (2) would be used for this purpose. This minimum value of R is then obtained by inserting Q_{delreq} into either (2) or (3) along with M (given by PathMTU), C_{tot}, D_{tot}, r, b, and p, as appropriate. If the obtained value of R exceeds the path bandwidth value as obtained from the AdSpec of the received Path message, it must be reduced accordingly. The receiver can now create a reservation specification, RSpec, comprising firstly the calculated value R of bandwidth to be reserved in each router, and second a slack term that is initialized to zero[6]. The RSpec can now be used in the creation of a Resv message, which also includes the following:

- An indication of the reservation style, which can be FF, SE or WF (see the next subsection).
- A filter specification, Filter Spec (omitted for the WF reservation style). This is used to identify the sender(s), and the format is identical to that of the Sender Template in a Path message.
- A flow specification, Flow Spec, comprising the RSpec and a traffic specification, TSpec. TSpec is usually set equal to the Sender TSpec, except M will be given by PathMTU obtained from the received AdSpec.
- Optionally, a reservation confirm object, ResvConf, containing the IP address of the receiver. If present, this object indicates that the node accepting this reservation request, at which propagation of the message up the distribution tree finishes, should return a ResvConf message to the receiver to indicate that there is a high probability[7] that the end-to-end reservation has been successfully installed.

The Resv message is now sent to the previous hop upstream as obtained from the stored path state. Upon reaching the next upstream router, the Resv message can be merged with other Resv messages arriving on the same interface, according to certain rules as described in the next subsection, to obtain an effective Flow Spec and Filter Spec. The following actions are then taken:

[6] Sometime, even with R set to the minimum value of r, the resultant end-to-end queuing delay as given by (2) and (3) will still be less than Q_{delreq}, in which case the difference can be represented in a nonzero slack term.

[7] In practice there are certain scenarios in which a ResvConf message might be received by a receiver, only for the request to be rejected shortly afterwards.

- The effective Flow Spec is passed to the traffic control module within the router, which applies both admission control and policy control to determine whether the reservation can be accepted. Admission control is concerned solely about whether enough capacity exists to satisfy the request, while policy control also takes into account any additional factors that need to be considered (e.g., certain policies may limit a user reserved bandwidth even if spare bandwidth exists).
- If the reservation attempt is denied, any existing reservations are left unaltered, and the router must send a ResvErr message downstream.
- If the reservation request is accepted, reservation state is setup in accordance with the effective Flow Spec and Filter Spec as described in the next subsection. In accepting the request it may be permissible to alter the Rspec associated with the reservation from (R_{in}, S_{in}) to (R_{out}, S_{out}) in accordance with the rules described later. The resultant reservation may then be merged with other reservations in accordance with the rules in the next subsection to obtain a new Resv message which is sent to the next router upstream, the address of which is obtained from the stored path state.

§1.3.4.3 Reservation styles and merging

Associated with each reservation made at a router interface is a Filter Spec describing the packets to which the reservation applies along with an effective Flow Spec. Both the Filter Spec and effective Flow Spec are obtained from a merging process applied to selected Resv messages arriving on the router interface. The rules for merging are dependent on the reservation style of each Resv message, as described below. In addition, the router calculates the Filter Spec and Flow Spec of Resv messages to be sent to the previous hop(s) upstream by applying style-dependent merging of stored reservation state. Any changes to stored reservation state that result in changes to the Resv messages to be sent upstream will cause an updated Resv message to be sent upstream immediately. Otherwise, Resv messages are created based on stored reservation state and sent upstream periodically. As for path state, all reservation state is stored in routers using soft-state and consequently relies on periodic refreshes via Resv messages to prevent state timeout. In addition, just as a PathTear message exists to explicitly tear down path state, a ResvTear message exists to explicitly tear down reservation state. Currently three reservation styles are permissible, as described below and illustrated in figure 8 to figure 10 where the convention style (Filterspec{Flowspec}) is used to summarize the requests made by the Resv messages. It should be noted that the merging processes described below apply only to packets of the same session (this is true of any RSVP process). Also, merging can only occur between messages with the same reservation style. Details of the reservation styles are as follows, where it is assumed that each interface *I* in figure 8 to figure 10 is routable to any other router interfaces.

§1.3.4.3.1 Fixed filter (FF)

The Filter Spec of each FF reservation installed at an interface consists of a single sender only. The effective Flow Spec of the reservation installed is the maximum of all FF reservation requests received[8] on that interface for that particular sender. The Flow Spec of the FF Resv message unicast to the previous hop of a particular sender is given by the maximum Flow Spec of all reservations installed in the router for that particular sender.

§1.3.4.3.2 Wildcard filter (WF)

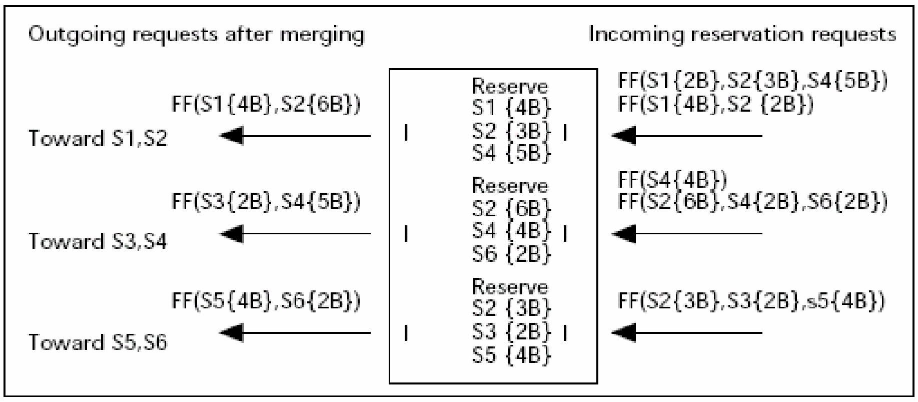


figure 8 - Fixed Filter reservation example

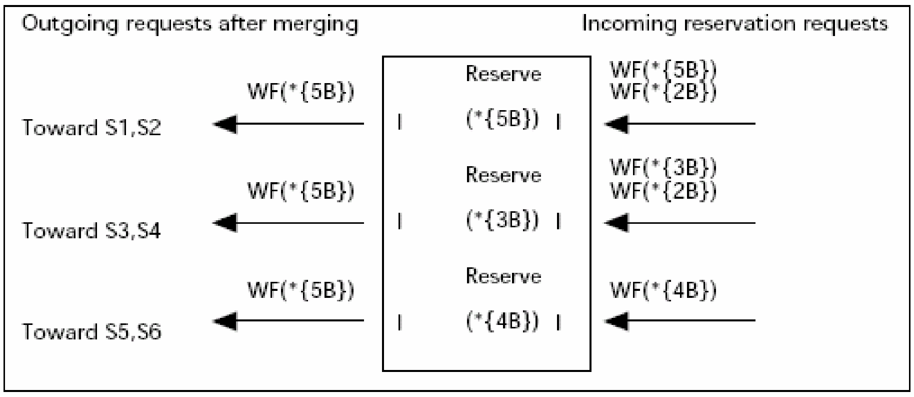


figure 9 - Wildcard filter reservation example

[8] In cases where the interface connects to a shared-medium LAN, Resv messages from multiple next hops may be received.

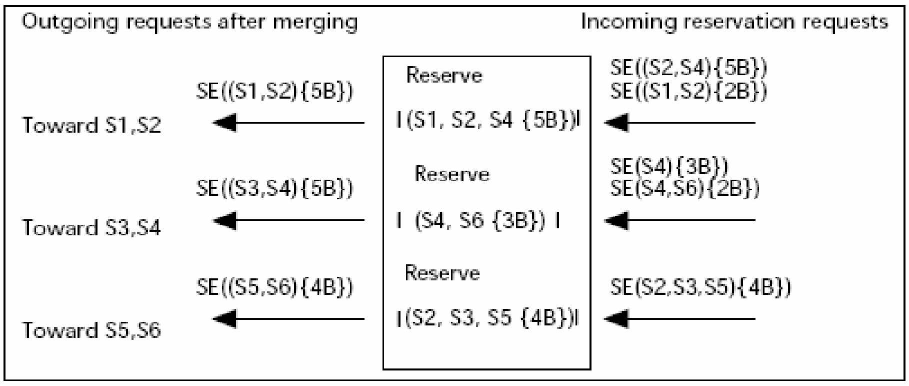


figure 10 - Shared Explicit reservation example

The Filter Spec of each WF reservation installed at an interface is wildcard and matches on any sender from upstream. The effective Flow Spec installed is the maximum from all WF reservation requests received on that particular interface. The Flow Spec of each WF Resv message unicast to a previous hop upstream is given by the maximum Flow Spec of all WF reservations installed in the router[9].

§1.3.4.3.3 Shared explicit (SE)

The Filter Spec of each SE reservation installed at an interface contains a specific set of senders from upstream and is obtained by taking the union of the individual Filter Spec from each SE reservation request received on that interface. The effective Flow Spec installed is the maximum from all SE reservation requests received on that particular interface. The Filter Spec of an SE Resv message unicast out of an interface to a previous hop upstream is the union of all senders whose previous hop is via that interface and who are contained in the Filter Spec of at least one SE reservation in the router. Likewise, the Flow Spec of this SE Resv message is given by the maximum Flow Spec of all SE reservations whose Filter Spec contains at least one sender whose previous hop is via that interface. SE and WF styles are useful for conferencing applications where only one sender is likely to be active at once, in which case reservation requests for, say, twice the sender bandwidth could be reserved in order to allow an amount of over-speaking. Although RSVP is unaware of to which service (controlled load or guaranteed) reservations refer, RSVP is able to identify those points in the distribution tree that require reshaping in the event that the reservations are for guaranteed service, as described previously. Consequently, at all such points RSVP informs the traffic

[9] Strictly speaking, only WF reservations whose scope applies to the interface out of which the Resv message is sent are considered for this second merging process. Scope details are required for WF reservations on non-shared trees to prevent looping.

control mechanisms within the appropriate router accordingly, although such action will only result in reshaping if the reservation is actually for guaranteed service.

§1.3.4.4 The slack term

When a receiver generates an RSpec for a Resv message to be sent for a guaranteed service reservation request, it must include a slack term, *S*(ms), as well as the amount of bandwidth *R* to be installed in each router along the path. *S* represents the amount by which the end-to-end delay bound will be below the end-to-end delay required by the application, assuming each router along the path reserves *R* bandwidth according to the guaranteed service fluid approximation. Inclusion of a non-zero slack term offers the individual routers greater flexibility in making their local reservations. In certain circumstances this greater flexibility could increase the chance of an end-to-end reservation being successful. Some routers have deadline-based schedulers that decouple rate and delay guarantees. Such a scheduler may sometimes be unable to meet its deadline requirement for guaranteed service, in which case it might still be able to accept the reservation, provided the slack term is at least as large as the excess delay. The excess delay would then be subtracted from the slack term before unicasting the Resv message to the previous hop upstream. Similarly, a rate-based scheduler might be able to admit a reservation request by reserving less than the requested bandwidth and unicasting the reduced reservation request to a previous hop upstream, provided it could extract enough slack. Any router using available slack to reduce its reservation must conform to the rules in (4) to ensure that the end-to-end delay bound remains satisfied:

$$(4) \qquad S_{out} + \frac{b + C^i_{tot}}{R_{out}} \le S_{in} + \frac{b + C^i_{tot}}{R_{in}} \qquad r \le R_{out} \le R_{in}$$

where C^i_{tot} is the cumulative sum of the error terms *C* for all the routers that are upstream of, and including, the current element *i*. (R_{in}, S_{in}) is the reservation request received by router *i*. (R_{out}, S_{out}) is the modified reservation request unicast to the previous hop router upstream. An example of how intelligent use of the slack term can increase the probability of an end-to-end reservation request being accepted is illustrated in figure 11 and figure 12. Suppose the token bucket rate of the data to be sent is 1.5 *Mbps*, and the receiver has calculated from the TSpec and AdSpec parameters in received Path messages that the desired end-to-end delay can be achieved by a reservation of (*R* = 2.5 *Mbps*, *S* = 0), which is then requested in figure 11. However, because *R*3 only has 2 *Mbps* of unused bandwidth and there is no slack available, the reservation is denied. In the figure the reservation is increased to *R* = 3 *Mbps*, and the amount by which such a reservation would be within the required delay bound is put in the slack term ($S > 0$). $R5$ and $R6$ reserve the requested 3 *Mbps*. $R3$ can only reserve a value of 2 *Mbps*, which, if used as the new reservation value in the propagated Resv message, will cause an increase in the end-to-end delay bound. $R3$ can calculate this increase, *di*, and if it is less than the value of the slack term, $S1$, in the received Resv message, the request can be accepted and a reservation of 2 *Mbps* installed in $R3$. $R3$ will then set the RSpec in the Resv message to (R = 2 *Mbps*, $S2 = S1 - di$) before unicasting it to the next hop upstream, which results in $R2$ and $R1$ also reserving 2 *Mbps*. The end-to-end delay bound of the reserved path is now no greater than for a reservation of 2.5 *Mbps* in every router if that were possible.

§1.4 Conclusions

The DiffServ and IntServ architectures presented above represent the mostly diffused philosophies adopted by researchers and network managers to face QoS issues. DiffServ and IntServ approach the QoS provisioning problem in two orthogonal modes. In fact, while DiffServ proposes to differentiate the network behavior by means of aggregate flows in a very limited number of classes, symbolized by PHBs and PDBs, IntServ proposes to manage per-flow resource allocation in all network nodes, taking as reference strictly guaranteed and loosely guaranteed services. The DiffServ is a network stateless vision, where QoS can be enhanced with respect to the legacy Best Effort Internet, but no guarantees can be assured to any user or flow. This is a light and promising approach, because of its flexibility, scalability and the possibility to be customized. On the contrary, IntServ envisions a stateful networking, resulting in the possibility to provide QoS guarantees in change of sensible network signaling overhead and relevant augmented complexity in the routers.

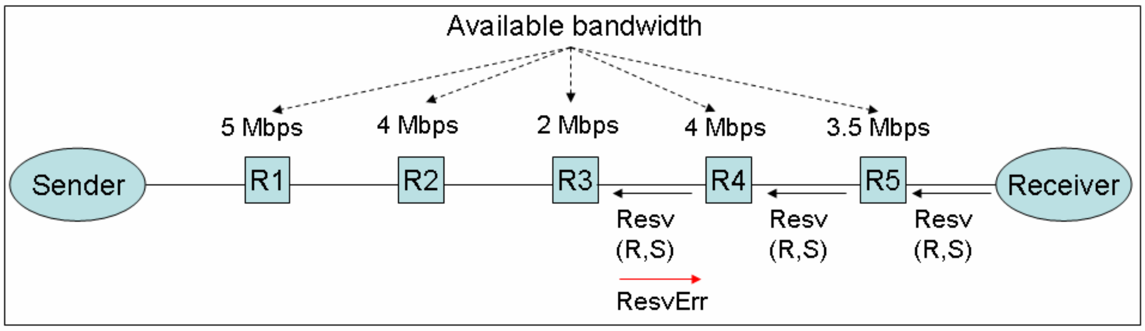


figure 11 - R = 2.5 Mbps, S = 0. Reservation request denied.

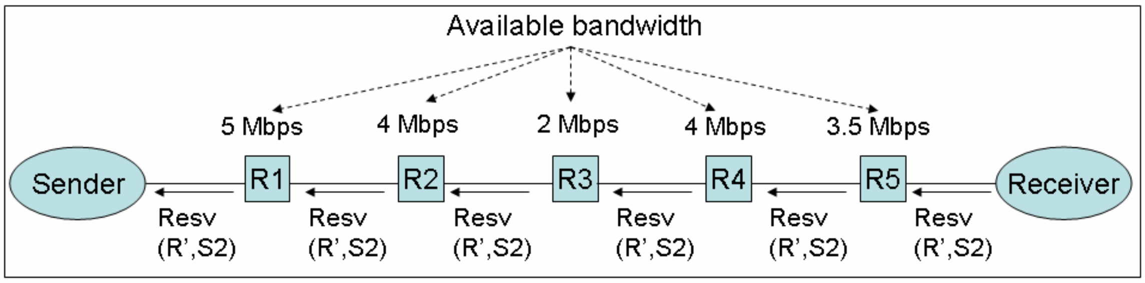


figure 12 - R = 3 Mbps, S1 > 0, R' = 2 Mbps, S2 < S1. Reservation accepted.

As a matter of fact, DiffServ and IntServ represent two opposites in the field of network architecture design strategies, even though they show complementary features, suitable to be deployed in access networks (IntServ) and core networks (DiffServ). However, in the following of this thesis, we will show that a DiffServ IP network can be endowed with some particular QoS-aware tools that permit to add effective QoS capabilities and enhance the gettable service level without hurting DiffServ principles.

Chapter II

Traffic control issues

In the previous chapter we introduced some possible extensions to the IP architecture, with the aim of enhancing the QoS level that a network can be made able to support. We have also shown that any modified IP network architectures need some extra tools that exploit the capabilities of the architectures and enforce QoS differentiation among users and flows. In this field, a fundamental role is played by the admission control tools adopted in the network. In fact, the tuning of admission control parameters determines the characteristics of the traffic that traverses the network and, in turn, it shapes any kind of performance figures. This is particularly relevant for that class of applications which do not adapt to the network congestion state on their own. These applications send their data as long as their transmission buffer is not empty. They do not exploit any tool that monitor the network state and change the transmission rate accordingly. Not reactive data sources could severely degrade the congestion state of a network, but they are needed for market appealing services, like voice over IP, video on-demand, news broadcast and so on. However, congestion reactive application, e.g., TCP-based applications, could unsuitably reduce their transmission rate due to the overbooking of network facilities, so that a preventive traffic control is also needed. Thus, various aspects of QoS-oriented traffic control and engineering are addressed in the present and in the next chapters and admission control issues are primarily considered. Here we will take into consideration the case of not reactive flows which adopt UDP-like transport protocols and we will introduce to the families of endpoint admission control, parameter based admission control and measurement based admission control. Eventually we will show the different impact of such schemes on different network traffic scenarios. In particular we will afford a solution able to significantly reduce the burstiness of data traffic in a network that manages flow aggregates (e.g., DiffServ), even when self-similarity arises in the originating traffic sources. In doing so, we will show how the measurement based admission control schemes result superior *in principle* to parameter based admission control paradigms. Further measurement based and DiffServ-oriented admission control mechanisms will be also presented in the following chapters.

This chapter in subdivided into three main sections: the first (§2.1) introduces to the admission control schemes and in particular to parameter based admission control (PBAC -

§2.1.1), measurement based admission control (MBAC - §2.1.2) and endpoint admission control (EAC - §2.1.3). Next section §2.2 thoroughly investigates the impact of MBAC and PBAC schemes on traffic burstiness, with particular reference to the "long range dependence" that arises in real network traffic (§2.2.1) and its negative impact on network performance. A comparison between the implementations of MBAC and PBAC is given in subsection §2.2.2, and a simulative framework is defined in subsection §2.2.3. Performance results are then proposed in subsection §2.2.4. Eventually, section §2.3 concludes the chapter.

§2.1 Admission Control Schemes

The amount or resources a network has at its disposal is, in general, not enough to satisfy all possible users in the same instant. In fact, users do not need to access the same resources at every time instant; on the contrary, the majority of customers tries to access network resource for a very minor percentage of its life time. This behavior makes network resources suited to be shared in a statistical multiplexing fashion with a large multiplexing factor. However, it is possible that particular network segments or particular sets of resources can be timely over-requested, and in turn overloaded if no countermeasures are adopted. This is the role of admission control protocols: to check for the availability of resources before to accommodate a new flow, a new user or a new service. In fact, the same argumentation can be repeated if we consider a scenario in which a given amount of resources is reserved to a particular class of service that can be accessed by multiple users. Furthermore, some applications, such as video conferencing or streaming audio, produce network traffic which requires a guaranteed level of QoS to work properly. These QoS requirements may be in terms of a minimum bandwidth, bounded end-to-end delays, or maximum packet loss rates suffered by a flow. Network nodes that support such flows must be able to allocate and maintain their finite network resources to uphold their guarantees. Thus, these nodes may also have to reject new traffic flows that would cause a violation of promises. The admission control decision may be as straightforward as a simple inequality calculation: if the sum of the bandwidth usage of the current flows and a new flow is greater than a given threshold (the target network utilization), then reject the flow. These QoS guarantees, which have no tolerance for violations, are called *hard* guarantees, and some flows demand this guaranteed service. Other flows, however, may accept some amount of QoS guarantee violation, usually bounded by some probability values. This is called predictive service, and such statistical, or *soft*, guarantees provide more flexibility for the admission control algorithm, leading to increased network utilization.

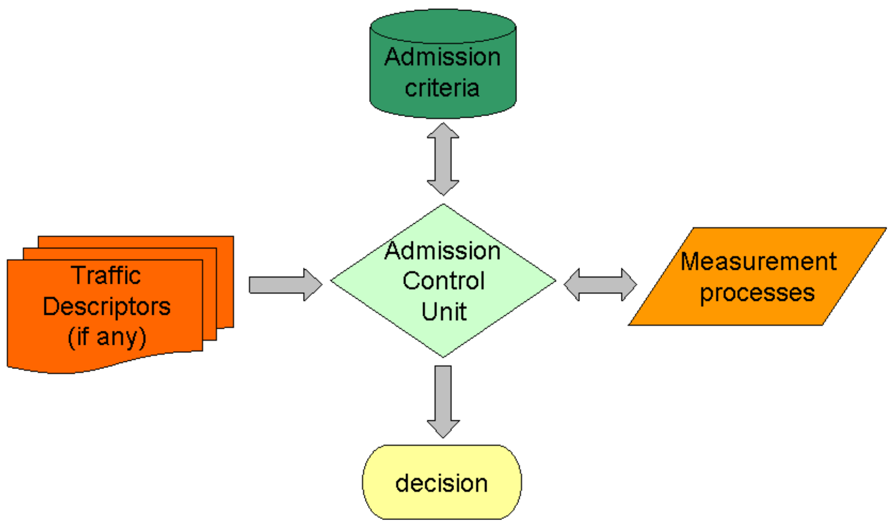


figure 13 - The general admission control scheme

There are many possible admission control schemes that can be classified according to the network entities involved or according to the modalities adopted in the resource management. Thus it is possible to distinguish between centralized and distributed admission control systems, depending on the location of the admission control functions. A particular case of distributed admission control is the endpoint admission control, where decisions are taken by connection endpoints after having probed the network state. A further classification can be done in accordance with the resource consumption estimate that is performed by the admission control unit: a parameter based approach relies on the knowledge of certain traffic flow parameters, while a measurement based admission control relies on run time estimates of the traffic behavior. Parameter and measurement based admission control schemes can either be centralized or distributed.

Eventually, the three basic components of admission control schemes are traffic descriptors, measurement processes and admission decision criteria, as depicted in figure 13 together with the admission control unit. The figure illustrates the role of traffic descriptors, admission criteria and measurement processes and their relationship with the overall admission process:

- Traffic descriptors, if present, convey a set of parameters that characterize traffic sources. A typical traffic descriptor is a token bucket, which is comprised of a token fill rate *r* and a token bucket size *b*. A source described by a token bucket will send at most $b+ t*r$ amount of traffic over any period of *t* larger than one packet transmission time. Sometimes a token bucket also contains a peak rate *p*, which constrains the smallest packet inter-arrival time to be 1/p. In any case, traffic descriptors allow flows to present their usage requirements to an admission control.

- Measurement processes offer run time computed network parameters to the admission control unit. These process are adopted since the parameters reported in the traffic descriptors are often difficult to calculate, they roughly summarize the average traffic behavior and its extreme deviations, but do not suit for bursty traffic. A frequently used parameter is the average arrival rate of the aggregate traffic.
- Admission Criteria are the rules by which an admission control scheme accepts or rejects a flow. Since the network resources allocated to a traffic class are shared by all the flows of that class, the decision to accept a new flow may affect the QoS commitments made to the admitted flows of that particular class. The new flow can also affect the QoS of flows in lower priority classes. Therefore, an admission control decision is usually performed based on an estimation of the effect the new flow will have on other flows and the utilization target of the network.

§2.1.1 Parameter Based Admission Control (PBAC)

A category of admission control schemes is based on the network knowledge of flow parameters. This is the case of RSVP signaled services in IntServ and, in general, the parameter based admission control applies in any network scenarios where each source disposes of means to declare the traffic behavior that want to establish. This is, an explicit message is sent to ask for a specific set of resources. Alternatively, this approach is also suited for services where users has established an a-priori agreement for a given amount of network resources, or again when all data flows behave identically, in a known way. Another feature that makes it possible to deploy a PBAC is the availability of a signaling pipe that allows users and network nodes to exchange information about the service the user requires or the service that the network is able to offer.

PBAC schemes exploit the knowledge of parameters like the data rate a flow will offer if admitted, provided in terms of average value and/or in terms of maximum transmission rate (peak rate), the desired delay and delay jitter that each data packet can tolerate, the loss rate of packets injected by the source, and, in general, some statistic bound to delay, jitter and loss performance. Thus the number of possible PBAC implementations ranges in a very wide field, even though the larger is the number of parameters to be considered, the more heavy the signaling overhead and the network computation overload, and, in turn, the more difficult is a real effective implementation.

As an example, the simplest parameter based admission control algorithm is the so called “simple sum algorithm”: it works by simply ensuring that the sum of requested resources does not exceed link capacity. For instances, resources can be summarized by the bandwidth consumption. Thus, let *S* be the sum of reserved rates, *B* the link bandwidth, *F* the name of a flow requesting admission, and *R* the rate requested by flow. This algorithm accepts the new flow if and only if the following inequality applies:

$$(5) \qquad S+R(F) < B$$

This is the simplest admission control algorithm and hence is being most widely implemented by switch and router vendors adopting PBAC schemes. However, when applicable, the efficiency of the simple sum algorithm is dependent on the definition of the bandwidth consumption parameter *R*. In fact, *R* does not take into account the possible variability of data rate at sender side. For instance, if *R* is the average data rate of a variable bit rate transmitter, relation (5) could be violated in terms of instantaneous values, while remaining still valid in terms of average rates. A solution is to define *R* as the peak rate, or, in alternative, relation (5) should be rewritten in terms of statistic bounds, e.g., by defining a (limited) probability that the relation is not respected.

In general, the implementation of a PBAC requires the support of the network architecture and of opportune signaling protocols. Eventually, it is worth noting that PBAC represent a suitable solution in the circumstances that all previously reported features apply and traffic characteristics are modeled into details. In this conditions, the network administrator can regulate the PBAC in order to attain the higher possible network and service performance. The point is that a deeply knowledge of every parameters and traffic variability not easily applies to real heterogeneous networks. Furthermore, sophisticated elaborations based on several a-priori traffic parameters are often unfeasible and sometime simply lead to meet QoS requirements by significantly reducing the admitted traffic, in a proactive fashion.

§2.1.2 Measurement Based Admission Control (MBAC)

While PBAC methods rely on the a-priori knowledge of the statistical characterization of the offered traffic, other algorithms base the decision whether to accept or reject an incoming connection on run-time measurements of the traffic aggregate process. If we assume sources can characterize their traffic accurately using traffic descriptors, the admission control unit can simply use parameters in the traffic descriptors. However, it is observed that real traffic source, such as real time and multimedia entertainment applications, are very difficult to characterize and the parameters in the traffic descriptor may only provide a very loose upper bound of the traffic rate. When the real traffic becomes very bursty, the network utilization can get very low if admission control is solely based on the parameters provided at connection setup time. Therefore, the admission control unit should monitor the network dynamics and use measurements such as instantaneous network load and packet delay to make its admission decisions. MBAC schemes take advantage of statistical multiplexing of traffic to maximize network utilization, even though to make an intelligent admission decision, MBAC must have an accurate measure of the amount of congestion and the amount of resources used in the network. There are a number of mechanisms to obtain these measurements, and they range from very simple traffic sampling to average estimates of load and performance figures by means of sliding windows filtering or exponential averaging filtering or also more complicated numeric filters (such as autoregressive and ARMA models, Kalman filters, etc.). Each has a rather significant effect on the behavior of the admission control schemes. A large number of MBAC algorithms have been proposed in literature (see for example [45], [47], [65]), but it appears that the measurement process, and in particular the length of the averaging periods and the way in which new flows are taken into account, are much more important than the specific admission criteria (either heuristic or theoretically founded) in determining the MBAC performance, in terms of throughput and packet loss (see [21]).

MBAC schemes have been traditionally proposed in order to solve the problem of admission control when no information can be retrieved about traffic sources or even when the load associated to PBAC schemes is not affordable. For instance, in a stateless network (e.g., DiffServ) a stateful parameter based per flow admission control cannot be implemented due to the impossibility to store in the network nodes any pieces of information about the number of already admitted flows or their specifications. Hence, MBAC was meant to merely approximate the PBAC behavior, without the aim of competing with performance offered by the original PBAC scheme. However, the evolution of MBAC techniques allows to obtain performance comparable with the adoption of a PBAC scheme. Moreover, in particularly bursty traffic condition, we will show that any MBAC scheme is in principle able to outperform the traditional parameter based approach (as a matter of fact, PBAC are not meant to deal with bursty traffic, and do not perform well when burstiness arises).

§2.1.3 Endpoint Admission Control

Endpoint Admission Control (EAC) is a recent research trend in QoS provisioning over IP [20]. EAC builds upon the idea that admission control can be managed by pure end-to-end operation, involving only the source and destination hosts. According to [20], EACs share the common idea that accept/reject decisions are taken by the network endpoints and are based on the processing of “probing” packets, possibly carrying control information, injected in the network at setup to verify the congestion state of the source to destination path. This approach is a radical overhaul of existing admission control schemes, in which all the routers in the source to destination path are involved in connection admission and make an accept/reject decision based on their occupancy state.

At connection setup, each sender-receiver pair starts a probing phase whose goal is to determine whether the considered connection can be admitted to the network. In some EAC proposals [17], [20], [34], during the probing phase, the source node sends packets that reproduce the characteristics (or a subset of them) of the traffic that the source wants to emit through the network. Upon reception of the first probing packet, the destination host starts monitoring probing packets statistics (e.g., loss ratio, probes inter-arrival times) for a given period of time. At the end of the measurement period and on the basis of suitable criteria, the receiver takes the decision whether to admit or reject the connection and notifies back this decision to the source node.

Although the described scheme looks elegant and promising (it is scalable, it does not involve core routers), a number of subtle issues come out when we look for QoS performance. A scheme purely based on endpoint measurements suffers of performance drawbacks mostly related to the necessarily limited (few hundreds of ms, for reasonably bounded connection setup times) measurement time spent at the destination. Measurements taken over such a short time cannot capture stationary network states, and thus the decision whether to admit or reject a connection is taken over a snapshot of the network state, which can be quite an unrealistic picture of the network congestion level. The simplest solution to the above issue is to attempt to convey more reliable network state information to the edge of the network. Several solutions have been proposed in the literature. [24] proposes to drive EAC decisions from measurements performed on a longer time scale among each ingress/egress pair of nodes within a domain. [45], [67] and [93] use packet marking to convey explicit congestion information to the relevant network nodes in charge of taking admission control decisions. [74] performs admission control at layers above IP (i.e., TCP), by imposing each core router to parse and capture TCP SYN and SYN/ACK segments, and forward such packets only if local congestion conditions allow admission of a new TCP flow.

To summarize the above discussion, and to proceed further, we can state that an EAC is, ultimately, the combination of three logically distinct components (although, in some specific solutions the following issues are not clearly distinct):

- edge nodes in charge of taking explicit per flow accept/reject decisions;
- physical principles and measures on which decisions are based (e.g., congestion state of an internal link or an ingress/egress path, and particular measurement technique - if any - adopted to detect such state);
- the specific mechanisms adopted to convey internal network information to edge nodes (e.g., received probing bandwidth measurement, IP packet marking, exploitation of layers above IP with a well-defined notion of connection or even explicit signaling).

In such a view, EAC can be re-interpreted as a MBAC that runs internally to the network (i.e., in a whole domain or, much simpler, in each internal router). This MBAC scheme locally determines, according to some specific criteria (which can be as simple as not performing any measure at all, and taking a snapshot of the link state, or as complex as some of the techniques proposed in [21], [47]), whether a new connection can be *locally* admitted (i.e., as far as the local router is concerned). This set of information (one per each distinct network router) is implicitly (or explicitly) collected and aggregated at the edge nodes of the network; these nodes are ultimately in charge of performing the acceptance/rejection decision.

A group of EAC proposals aims at complete end-to-end EAC schemes (EEAC). Core routers only need to be properly configured to support head-of-line priority based forwarding penalizing probing packets over data packets, and to possibly limit the buffer sizes. No modification or cooperation from the internal network routers is required. The EEAC schemes only require modifications in the end terminals or the edge routers which will initiate the probing phase, take measurements on the arriving probing flow, make accept/reject decision based on the measured statistics on such flow, communicate the decision to the end user. The basic feature of end-to-end endpoint admission control is then that no additional functionalities are required to the core routers, which only have to be configured in order to implement priority based forwarding, or to limit the size of given buffers. Such solutions can therefore be immediately implemented in the Internet, and represent the first step of the evolution toward more refined endpoint admission control schemes. Indeed, we envision that EAC can represent an evolutionary approach in the sense that its philosophy is to progressively provide increasing level of QoS support, starting with statistical guarantees that can be provided by the simplest end-to-end EAC scheme, and progressively moving toward more complex EAC solutions with better QoS support.

§2.1.3.1 EAC in DiffServ

Of the three components outlined above, we argue that, in a DiffServ framework, the least critical issue is how to estimate the congestion state of a router without resorting to per flow operation. In fact, recent papers [21] and [47] have shown that *aggregate* load measurements are extremely robust and efficient. These schemes do not exploit per-flow state information and related traffic specifications. Instead, they operate on the basis of per-node aggregate traffic measurements carried out at the packet level. The robustness of these schemes stays in the fact that, in suitable conditions (e.g., flow peak rates small with respect to link capacities), they are barely sensitive to uncertainties on traffic profile parameters. As a consequence, it seems that scalable estimations can be independently carried out by the routers.

The real admission control problem is how to convey the state of core routers (evaluated by means of aggregate measurements) to the endpoints so that the latter devices can take *learned* admission control decisions, without violating the DiffServ paradigm. For obvious reasons, we cannot use explicit per flow signaling. Similarly, we do not want to modify the basic router operation, by introducing packet marking schemes or forcing routers to parse and interpret higher layer information. What we want to do is to *implicitly* convey the state of core routers to the end points, by means of scalable, DiffServ compliant procedures. A solution will be proposed in the next chapter, with a suitable evolution path to follow in order to gradually enhance network performance without enforcing a network revolution (i.e., without the need of immediately replace any network device already deployed).

§2.2 The effect of MBAC on Self-similarity

In this section we consider an admission-controlled traffic scenario, in which non-adaptive connections are offered to the network. As observed in wide area Internet traffic (see [31]and [80]), self-similarity is shown by the traffic aggregate, no matter the transport protocol or mechanism is adopted. In particular our work focuses on traffic generated by non-reactive sources, i.e., sources that do not react when congestion arises as for real time transmissions over UDP. Even though this kind of flows is not numerically predominant, it relates to a class of market strategically relevant applications, as real time multimedia streams, that is speedily growing, and a great effort has been recently expressed in order to offer to these applications a suitable QoS level.

We show the superiority of MBAC algorithms over PBAC, in particular when Long Range Dependence (LRD) or asymptotic second order self-similarity arises in the offered traffic. We show that a pure MBAC approach is capable of significantly reduce the LRD of the accepted traffic aggregate, thus yielding a considerable performance improvement, in terms of QoS experienced by each flow sharing the media. We focus on LRD and self-similarity because of the severe detrimental impact that they cause in network performance. This negative effect is due to the huge burstiness that is intrinsically shown by LRD and self-similar flows. This turns in a significant reduction of the number of flows that can be conjunctly admitted to the network. In fact, when QoS constrains are given, they have to be guaranteed at least in a minimum number of occurrence (e.g., only a limited percentage of out of QoS profile events can be tolerated); due to the elevate self-correlation of LRD flows, the probability to experience a very large variation from average behavior is very consistent and not well predictable, so that resource over-provisioning is needed without any other countermeasure is adopted. Contrary to PBAC, the MBAC approach results in a powerful tool, robust to traffic correlations properties, specially as to burstiness, and able to overcome the horizon of PBAC scheme, despite of the traditional meaning of MBAC role, commonly intended as an even coarse approximation for PBAC.

§2.2.1 Self-similarity and LRD in network traffic

Packet networks traffic is self-similar. The first experimental evidence was first given in [103], with reference to traffic generated by ethernet LANs. That paper stimulated the research community to explore the various taste of self-similarity. This phenomenon has been also observed in wide area Internet traffic, and many of the causes that contribute to self-similarity for both TCP, [82], [103], and UDP [5] traffic aggregates have been now more fully understood. In what follows, we focus our attention on traffic generated by sources non-reactive to network congestion such as real time streams and multimedia entertainment application. These sources are often the cause of a considerable degree of LRD in the traffic aggregation. At the same time, these are the applications that mostly suffer for the LRD implications, because of the unpredictable delay arising in the congested links, due to very high delay jitter typically generated when network queues are loaded in a bursty way.

In particular, it has been shown that Long Range Dependence, also known as asymptotic second order self-similarity, arises when flows has heavy-tailed (HT) periods of activity/inactivity (see[48]). The random variable assumes a HT behavior when its cumulative distribution function converges to:

$$(6) \qquad F(t) \sim 1 - at^{-c} \textit{ as } t \to \infty\textit{, with } 1<c<2 \textit{ and } a= \textit{constant.}$$

Thus we assume, for convenience, a traffic aggregate scenario resulting from the superposition of homogeneous heavy-tailed flows. In particular, [81], [68] and [103] give insights into the relation between LRD of aggregate traffic and heavy-tailedness. Since a widespread evidence that humans as well as computer sources tend to behave as heavy-tailed *On/Off* sources ([31], [103]), the result should be considered a physical explanation of the traffic self-similarity, independently of network or protocol traits, rather than a mere way to generate self-similar traces. Self-similarity has a global negative impact on network performance ([46], [74], [81], [87]). In fact, for a given link capacity, a buffer size, and a fixed number N of superposed offered flows, the QoS (e.g., loss ratio, delays, etc) experienced by heavy-tailed flows results much worse than that experienced by flows whose activity/inactivity periods are drawn from exponential distributions. By repeating this study for various values of N, it is straightforward to design a traditional (parameter based) Connection Admission Control (CAC) scheme. The PBAC scheme consists in checking that the number of flows admitted to the considered link never exceeds a threshold N_t, computed as the maximum number of connections that can be admitted while still satisfying predetermined QoS requirements. The "parameter-based" stays in the fact that the threshold N_t depends on the flow statistic parameters. As a consequence of self-similarity, the maximum number N_t of heavy-tailed flows that can be admitted to a link may be much lower than in the case of Markovian flows.

The aim of our research is to understand what happens when a PBAC rule is replaced by a MBAC scheme as in [21], [42], [45], [47], [65],where a number of quite unexpected beneficial effects can be highlighted. Our simulation test will allow to draw several conclusions that we anticipate here. Firstly, MBAC schemes provide superior performance than PBACs when LRD flows are considered. Secondly, the traffic aggregate resulting from the accepted flows shows very little LRD - in other words, unlike PBACs, MBAC appears capable of smoothing the self-similarity of the accepted traffic aggregate. Secondly, but not less important, we argue that MBAC approaches are not mere approximations of ideal PBAC schemes, useful in situations where the statistical traffic source characterization is not fully known. On the contrary, they appear to be a promising, powerful and practical way to compensate the high variability of LRD traffic, and therefore improve the network efficiency.

In the following sections we discuss and intuitively justify the important role of MBAC in the presence of self-similar traffic. Upon that, we describe the specific MBAC algorithm that we have adopted, and the methods to evaluate self-similarity. Eventually, simulation results are given and discussed.

§2.2.2 MBAC vs. PBAC implementations

It is frequently considered *obvious* that the ultimate goal of any MBAC scheme is to reach the *ideal* performance of a PBAC scheme. In fact, MBAC schemes are traditionally meant to approximate the operation of a parameter based CAC by estimating the state of the system e.g. in terms of number *n(t)* of admitted flows at time *t*. On the opposite, in this section we want to highlight that MBAC algorithms should have a broader theoretical target. In fact, MBAC schemes cannot rely on the a-priori knowledge of the traffic characteristics, and therefore use an estimate of the accepted traffic aggregate, but they can outperform traditional PBAC approaches, in particularly burst traffic conditions.

This superiority of MBAC can be intuitively justified by comparing the simulation traces presented in figures figure 14 and figure 15, in which two typical traffic behaviors, respectively related to a PBAC scenario and to an MBAC one, are depicted. The figures report both the normalized number of accommodated connections, and the smoothed link load, as measured by the autoregressive filter adopted in the MBAC, whose time constant is of the order of 10 seconds (see section §2.2.3).

figure 14 reports results for an ideal PBAC. In the simulation run a very high offered load (~ 600%) was adopted, so that the number of flows admitted to the link sticks, in practice, to the upper limit (i.e., N_t = 129 flows, in the example, corresponding to about *88%* in link utilization). The leftmost 200 simulation seconds, represented in figure 14, show that, owing to LRD of the accepted traffic, the load offered by the admitted sources is consistently well above the nominal average load. Traffic bursts even greater than the link capacity are very frequent. On the other hand, as shown by the rightmost 200 seconds, there are long periods of time in which the system remains under-utilized. The criticality of self-similarity lies in the fact that the described situation occurs at time scales which dramatically affect the loss/delay performance. Aggregate traffic behaves very differently when an MBAC scheme is adopted.

figure 15 reports results for the simple MBAC scheme described in section §2.2.3.2. In this case, new connections are blocked as long as the offered load measurement is higher than a given threshold[10]. Specifically, we see from both leftmost and rightmost plots that the offered load measurement fluctuates slightly around the threshold. However, long term traffic bursts are dynamically compensated by a significant decrease in the number of admitted connections (leftmost plot). Instead, if admitted connections continually emit below their nominal average rate, the number of admitted connections significantly increases. This compensation capability of MBAC schemes leads us to conclude that MBAC is very suited to operate with LRD traffic: the quantitative analysis carried out in section §2.2.4, in fact, confirms this insight.

§2.2.3 The simulation scenario

To attain evidence for our thesis, we choose to apply a simulative approach. In fact, due to the long term characteristics to be observed in data traffic, and considering the amount of packets to be involved in the measurement campaign, the only alternative to simulation was the long lasting and direct supervision of traffic offered to a high capacity network node. Thus we preferred to develop a C++ event-driven simulator, based on a batch simulation approach. The simulation time was divided into 101 intervals, each lasting 300 simulated minutes, and results collected in the first "warm-up" time interval were discarded. As in many other admission control works ([21], [47]), the network model consisted of a single bottleneck link. In fact, the basic performance aspects of MBAC are most easily revealed in this simple network configuration rather than in a multi-link scenario. Unless otherwise specified, the link capacity was set equal to 2 *Mbps*, and an infinite buffer size was considered. Thus, QoS is characterized by the delay experienced by data packets rather than packet loss as in [65]. The rationale for using delay instead of loss is twofold. Firstly, loss performance depends on the buffer size adopted in the simulation, while delay performance do not require a choice of buffer size. Secondly, the loss performance magnitude may be easily inferred, for a given buffer size, from the analysis of the distribution of the delay. Furthermore, in a very large buffer scenario, the system is forced to keep memory of non-smoothed traffic bursts and therefore performance are strongly degraded in the presence of high traffic variability. However, a comparison with results dealing with a finite buffer scenario is also presented.

[10] In correspondence with the choice of $\boldsymbol{N_t}$ = 129, in PBAC, we select 89% in MBAC, since these selected value results in the same average throughputs in both PBAC and MBAC scenarios.

As performance figures, we evaluated link utilization and delay distribution, summarized by the 99th delay percentile. The 95% confidence intervals were evaluated. In all cases, throughput results show a confidence interval consistently lower than 0.3%. Instead, despite the very long simulation time, higher confidence intervals occur for 99th delay percentiles: less than 5% for MBAC results, and as much as 25% for PBAC (this is an obvious consequence of the self-similarity of the PBAC traffic aggregate). Nonetheless, even accounting for such uncertainty in the results, the PBAC and MBAC delay performance are clearly very different.

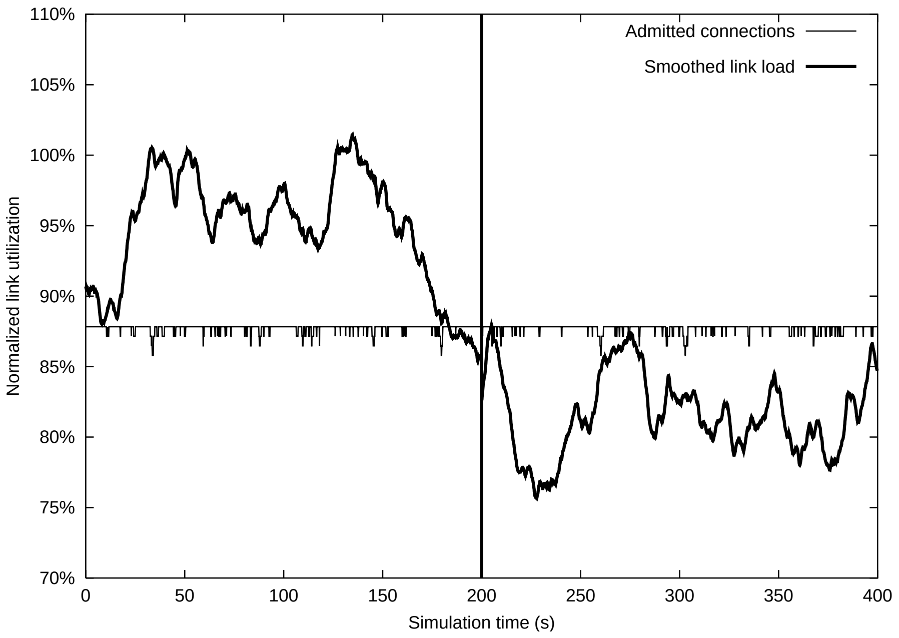


figure 14 - Traditional Admission Control operation

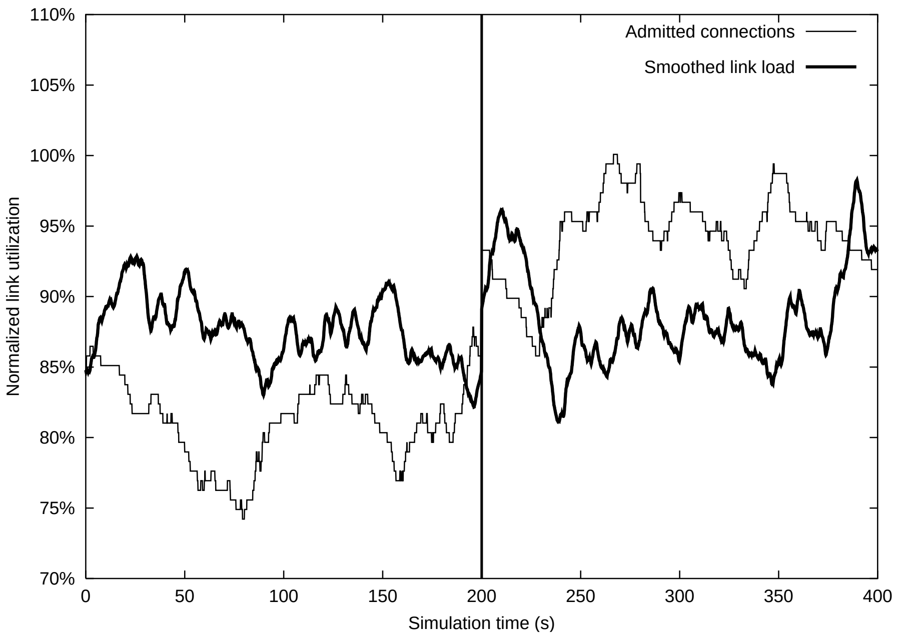


figure 15 – Measurement Based Admission Control operation

§2.2.3.1 Traffic sources

For simplicity, we have considered a scenario composed of homogeneous *On/Off* flows. While in the *ON* state, a source transmits 1000 bit fixed size packets at a Peak Constant Rate (*PCR*) ranging in the small interval 31 to 33 *Kbps*, in order to avoid source synchronization effects at packet level. Conversely, while in the *Off* state, it remains idle. The mean value of the *On* and *Off* periods have been set, respectively, equal to 1 *s* and 1.35 *s*, following a Brady model for voice traffic as in [19]. This results in an average source rate

$$(7) \qquad r = \frac{T_{on}}{T_{on} + T_{off}} E[PCR] \approx 13.6 \ Kbps$$

On and *Off* periods were drawn from two Pareto distributions with the same shaping parameter c = 1.5 (infinite variance), which exhibit heavy tails, hence the traffic aggregate is self-similar. In fact the cumulative distribution function of a Pareto Random Variable is

$$(8) \qquad F(t) = 1 - \left(\frac{t+s}{s} \right)^{-c} \ for \ t \geq 0$$

where s is a scale parameter, so that it suits with the definition of heavy-tailedness in (6).

Simulation experiments were obtained in a dynamic scenario consisting of randomly arriving flows. Each flow requests service from the network, and the decision whether to admit or reject the flow is taken by the specific simulated CAC. A rejected flow departs from the network without sending any data, and does not retry its service request again. The duration of an accepted flow is taken from a lognormal distribution [16] with mean 300 *s* and standard deviation 676 *s* (we adopted unitary variance for the corresponding normal distribution as reported in [16]), but connection duration is extended to the end of the last *On* period. Because of this, the real connection lifetime exhibits longer mean (320 *s*) and infinite variance. If the last burst were cut off, the process variance would become finite.

The flow arrival process is Poisson with arrival rate λ connections per second. For convenience, we refer to the normalized offered load ρ:

$$(9) \qquad \rho = \lambda \frac{r T_{hold}}{C_{link}}$$

being r the mean source rate, T_{hold} the average connection duration and C_{link} the link capacity. Depending on the simulation experiment, the arrival rate ranges from underload conditions (less than 50% of the link capacity) to severe overload conditions (up to 650%).

§2.2.3.2 Adopted MBAC algorithm

We have adopted and tested a very basic MBAC approach. In fact, The results in [21] show that different MBAC schemes present similar performance, and, more importantly, our goal is to show that the introduction of measurement in the admission control decision is the key to obtaining performance advantages in comparison to the PBAC approach, rather than the careful design of the MBAC algorithm. In this perspective the simpler the MBAC scheme is, the more general the conclusions are. Our MBAC implementation can be described as follows: a discrete time scale is adopted, with sample time T=100 *ms*. Let *X(k)* be the load, entering the link buffer during the time slot *k*, and *B(k)* its smoothed version, the bandwidth estimate, obtained by using a first order autoregressive filter:

$$(10) \qquad B(k) = \alpha B(k-1) + (1-\alpha)X(k)$$

We chose α*=0.99*, corresponding to about 10 *s* time constant in the filter memory. If a connection requests admission during the slot *k+1*, the connection is admitted only if the estimated bandwidth *B(k)* is less than a predetermined percentage of the link bandwidth. By tuning this percentage, performance figures can be obtained for various accepted load conditions.

Moreover, when a new flow is admitted, a step input is offered to the system, and a transient phase occurs, in which the load is underestimated, thus not reflecting the presence of a new flow ([42], [65]). A solution to prevent this performance-impairing situation is to artificially increase the load estimate to account for the new flow. Specifically, in our implementation, the actual bandwidth estimate *B(k)* is updated by adding the average rate of the flow:

$$(11) \qquad B(k) := B(k) + r$$

§2.2.3.3 Statistical analysis of self-similarity

The Hurst parameter *H* is able to quantify the self-similarity of the accepted traffic aggregate. For a wide range of stochastic processes *H*=0.5 corresponds to uncorrelated observations, *H*>0.5 to LRD processes and *H*<0.5 to Short Range Dependence processes. In order to evaluate *H*, we used three different methods. All methods receive in input a realization *X(i)* of the discrete-time stochastic process representing the load offered, during a 100 *ms* time window, to the link buffer by the accepted traffic aggregate.

The first method is the *Aggregate Variance* method, where the original series *X(i)* is divided into blocks of size *m* and the aggregated series $X^{(m)}(k)$ is calculated as:

$$(12)\qquad X^{(m)}(k)=\frac{1}{m}\sum_{i=(k-1)m+1}^{km}X(i)\qquad for\ k=1,2,...$$

The sample variance of $X^{(m)}(k)$ is an estimator of $Var[X^{(m)}]$; asymptotically:

$$(13)\qquad Var[X^{(m)}]\sim\frac{Var[X]}{m^{2(1-H)}}$$

In the second method, called *R/S*, for a time series *X(i)* with partial sum

$$(14)\qquad Y(n)=\sum_{i=1}^{n}X(i),$$

and sample variance $S^2(n)$, the R/S statistics or the rescaled adjusted range, is given by:

$$(15)\qquad \frac{R}{S}(n)=\frac{1}{S(n)}\left[\max_{0\le p\le n}\left(Y(p)-\frac{p}{n}Y(n)\right)-\min_{0\le p\le n}\left(Y(p)-\frac{p}{n}Y(n)\right)\right]$$

Asymptotically:

$$(16)\qquad E\left[\frac{R}{S}(n)\right]\sim Cn^{H}$$

The third approach adopted is the *Wavelet Estimator*. The spectrum of a LRD process *X(t)* exhibits power-law divergence at the origin:

$$(17)\qquad W_X(f)\sim c_f|f|^{1-2H}$$

The wavelet estimator method, proposed in [1], recovers the power-law exponent *1-2H* and the coefficient c_f turning to account the following relation:

$$(18)\qquad E\{d_X^2(j,l)\}=2^{j(1-2H)}c_fC$$

where $d_X(j,l)=<X,\psi_{l,j}>$ are the coefficients of the discrete wavelet transform of the signal *X(t)*, i.e., its projections on the basis functions $\psi_{l,j}$, constructed by the mother wavelet through scaling and translation (2^j and *l* are respectively the scaling and the translation factor). A MatLab implementation is freely distributed [51].

The three methods described allow to compute the Hurst parameter through a simple linear regression in a log-log diagram. A comprehensive overview of LRD statistical-parameters estimation, including the first two methods, can be found in [5]. The first two methods have the advantage of being simple and practical to implement, but often exhibit poor statistical properties [96]. The wavelet-based joint estimator, according to [1], displays reliable and robust statistical performance. Besides, if the process *X(i)* is Gaussian, this estimator is able to provide confidence intervals for the Hurst parameter.

A common problem is to determine over which scales LRD property exists, or equivalently the alignment region in the log-scale diagrams. Using the fit test of the MatLab™ tool [51] we determined for our traces the range from 2000 s -11th octave- to 250000 s -18th octave- (the two last octaves were discarded because there were too few values). All the three methods were applied over this scale.

§2.2.4 Performance evaluation

Every CAC scheme has some adjustable parameters that allow the network operator to set a suitable *utilization target* and a consequent QoS provisioning. In the present case, both for ideal PBAC and MBAC algorithms, a higher setting of the threshold value results in an increased system throughput, at the expense of delay performance. By adjusting this parameter, the proposed CAC rules can be designed to be more aggressive or conservative with regard to the number of flows admitted. Results presented in figure 16 were obtained by setting the PBAC and MBAC tuning parameters so that a target 90% link-utilization performance is achieved in offered traffic overload conditions. The figure compares the throughput/delay performance (99th delay percentiles, measured in *ms*, are numerically reported) of MBAC and PBAC, versus the normalized offered load.

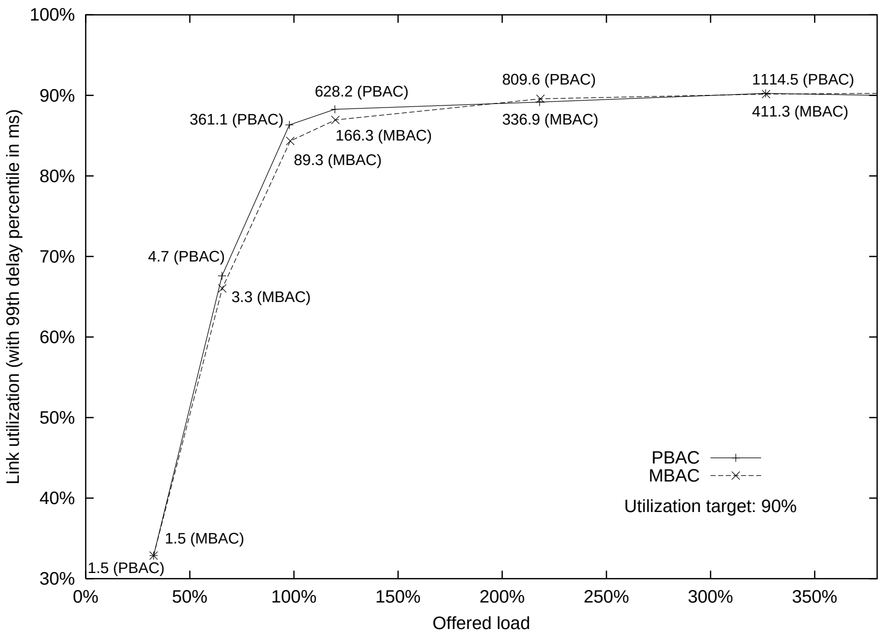


figure 16 - Link utilization vs. offered load

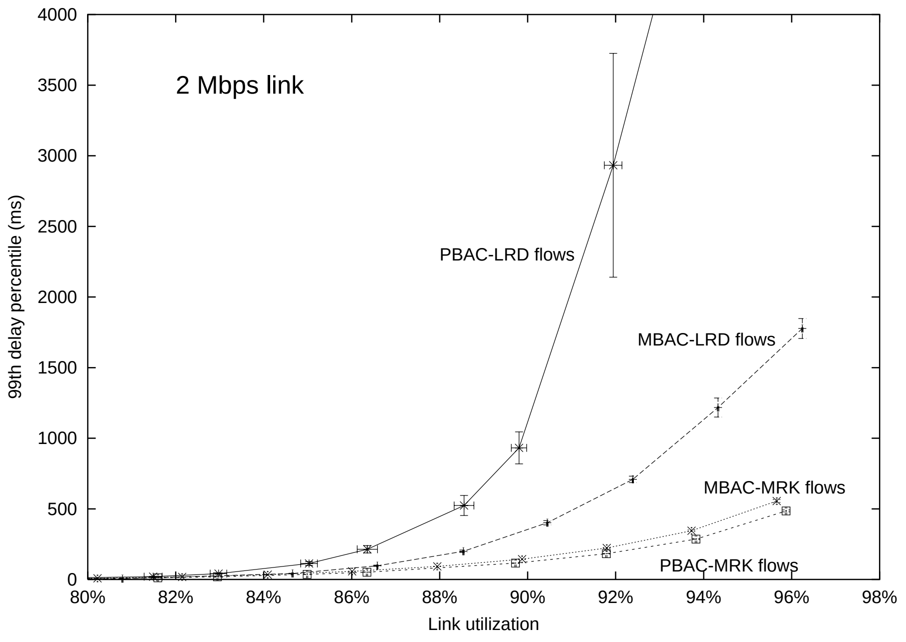


figure 17 - Delay performance vs. link utilization (2 Mbps link)

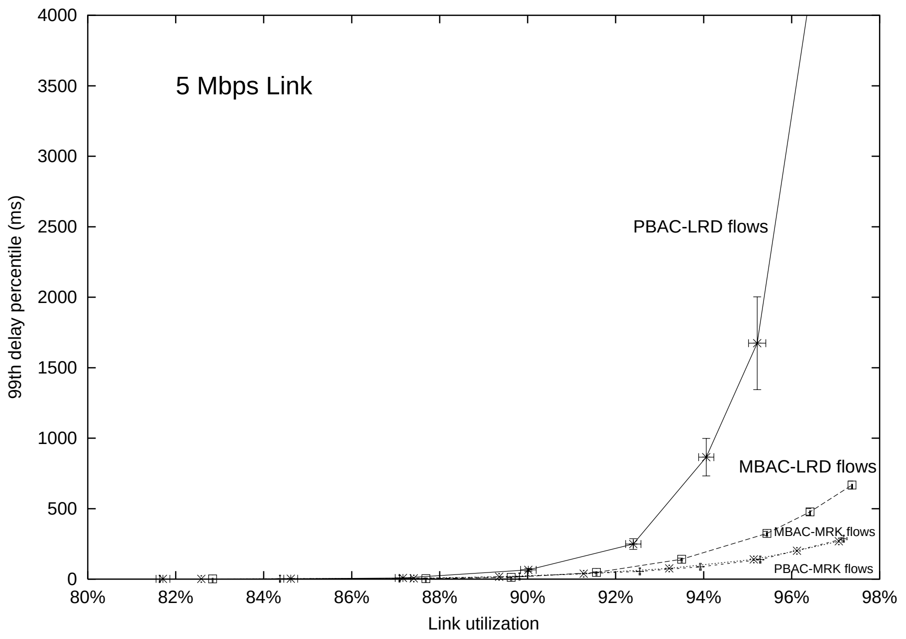


figure 18 - Delay performance vs. link utilization (5 Mbps link)

Minor differences can be noted in the capability of the considered schemes to achieve the performance target. A much more interesting result is the significantly lower MBAC delay versus the PBAC one. Rather than varying the offered load, figure 17 compares MBAC and PBAC by plotting their QoS performance versus the link utilization (following [21], the QoS versus utilization curve is called *Performance Frontier*). Specifically, the figure reports the delay/utilization performance frontiers of PBAC and MBAC in terms of 99th delay percentiles. The figure depicts results obtained for both LRD and Markovian flows, when

CAC thresholds range from low to full utilization, while the offered load is very high (about 600%). It is shown that better performance are obtained using a Markovian traffic model, but it is also emphasized the remarkable performance improvement provided by MBAC with respect to PBAC in LRD assumptions, especially for large link utilization. Under Markovian assumptions PBAC acts slightly better than MBAC, in fact no memory arises in offered traffic process, thus the best bandwidth control simply consists in monitoring the number of admitted connections. Instead, with LRD flows, MBAC performance frontiers assume intermediate values, better than PBAC-LRD performance frontiers but worst than the curves obtained under Markovian assumptions. Thus, MBAC appears to be more robust than PBAC to the traffic statistical properties. Moreover, in figure 18 the performance frontiers are plotted for a 5 *Mbps* link. Beside the general performance improvement in comparison to the 2 *Mbps* link scenario shown in figure 17, one can see that MBAC behavior, with Markovian traffic, is closer to the PBAC behavior, since the traffic granularity is reduced and the impact of a flow erroneously admitted is less significant. We argue that the impressive performance enhancement of MBAC over PBAC is due to the beneficial effect of MBAC in reducing the self-similarity of the accepted traffic aggregate.

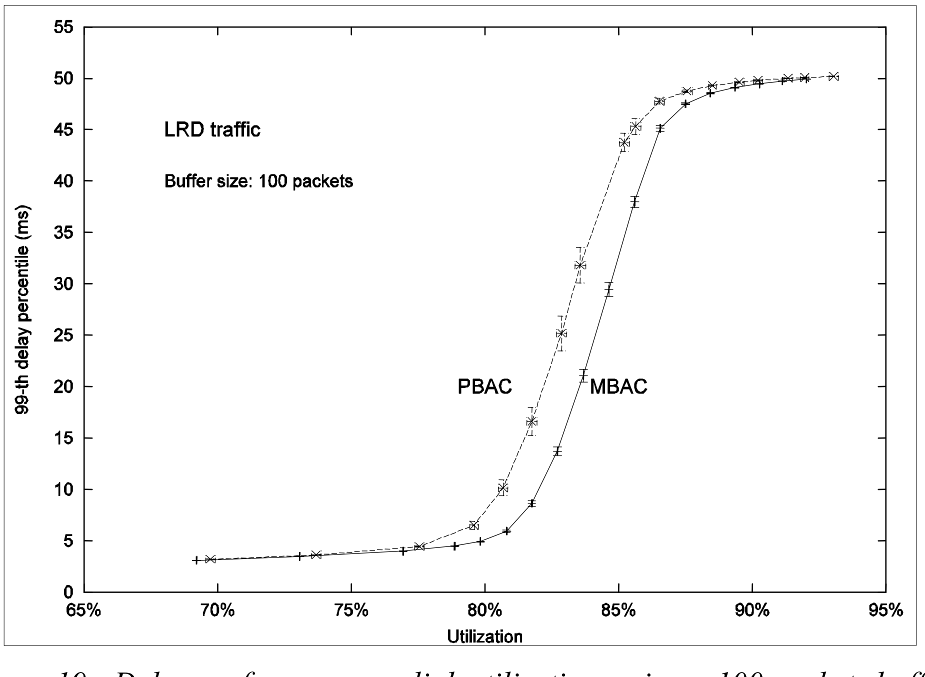


figure 19 - Delay performance vs. link utilization, using a 100 packets buffer

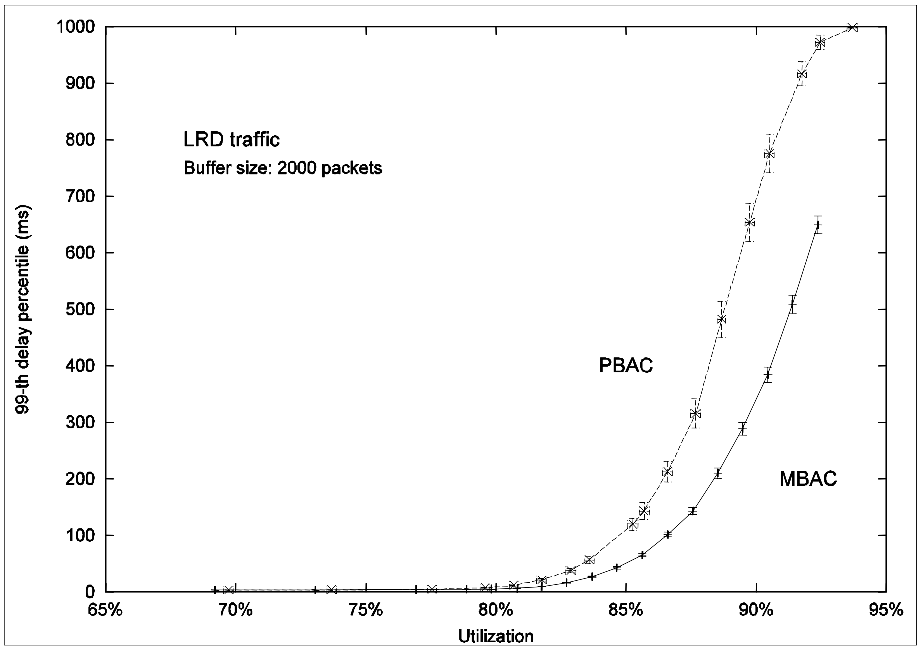


figure 20 - Delay performance vs. link utilization, using a 2000 packets buffer

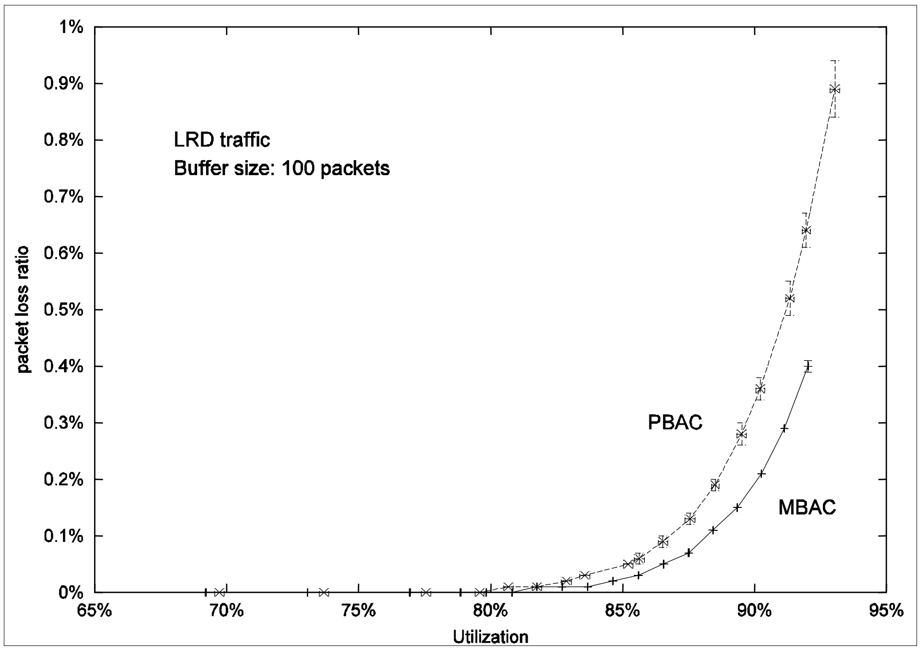


figure 21 - Packet loss ratio in high offered load condition, using a 100 packet buffer

Further results obtained by considering a finite buffer scenario are depicted in figure 19 and figure 20, where buffers having respectively 100 and 2000 packets length, are considered. Note that in the former case, in which a short buffer of 100 packets is considered, PBAC performance is comparable to the MBAC one only in extreme underload condition or conversely in a harsh over-utilized scenario. In the latter case, with a 2000 packets buffer, MBAC performance does not saturate at all, while in the PBAC scenario a greater buffer length should be needed in order to avoid losses occurring when the buffer is entirely filled.

These figures show, in the same conditions as before, i.e., in very high offered load conditions, that the performance gain resulting from MBAC scheme adoption is not impaired by a finite buffer application. Moreover figure 21 shows how MBAC approach allows a significantly improvement in packet loss ratio, which is controlled well under 1% even in a high offered load and a high link utilization scenario.

To quantify the time behavior of the two PBAC and MBAC traffic aggregate time series, figure 22 plots the estimated squared wavelet coefficients $d_X^2(j,l)$ versus the basis-function time scale. The 95% confidence interval under Gaussian assumption are depicted. While the two curves exhibit similar behavior for small values of the aggregation scale, the asymptotic slope of the PBAC plot is very different from the MBAC one. For reference purposes, the lines corresponding to *H*=0.5, and *H*=0.8 are also plotted in the figure. Note that the figure 22 appears to suggest that the MBAC-controlled traffic is not self-similar (Hurst parameter close to 0.5). An interesting consideration is that again in figure 22 the MBAC curve departs from the PBAC curve at a time scale of the order of about 100 seconds. Although a thorough investigation of this aspect was not performed in our work, and the complete understanding of the emergence of such a specific time scale is outside the scope of the present thesis, we suggest that it might have a close relationship with the concept of "critical time scale" outlined in [47].

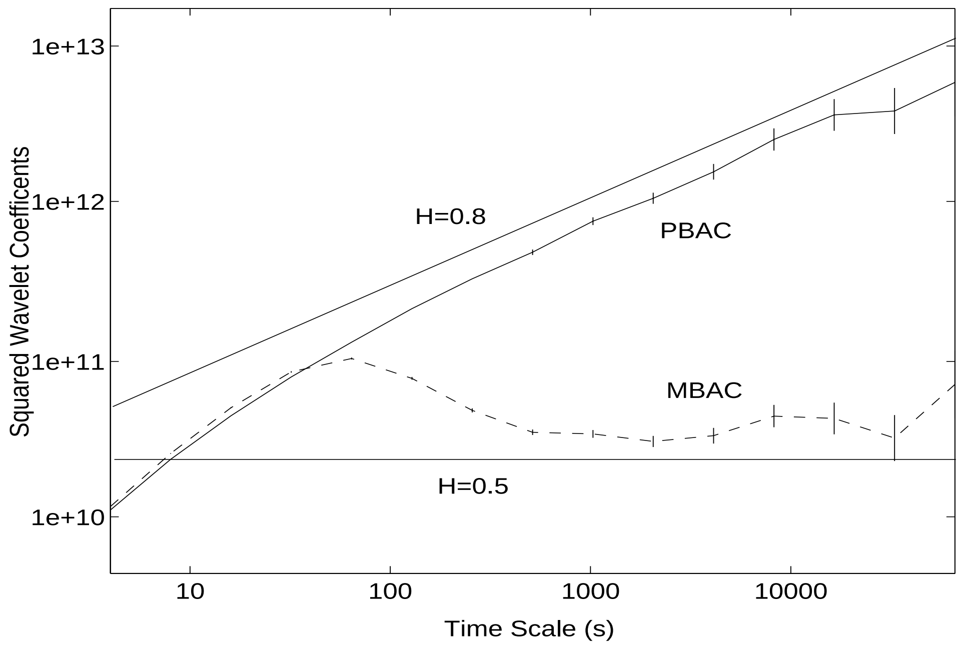


figure 22 - Wavelet coefficients plot

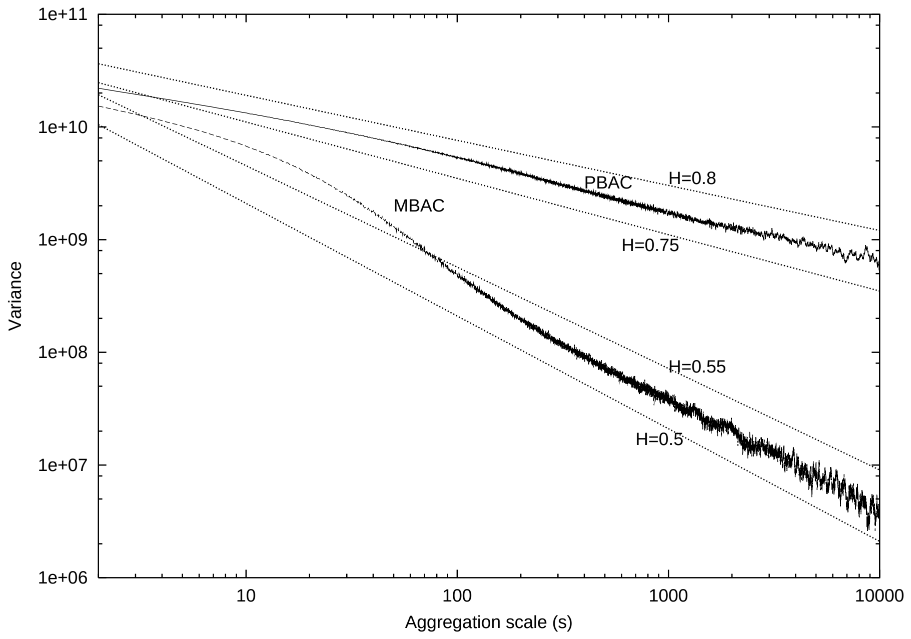


figure 23 - Aggregate variance plot

In a similar way, the time behavior of the two PBAC and MBAC traffic aggregate time series is reported in figure 23, in which a log-log plot of the aggregate variance, computed as described in section §2.2.3.3, is depicted. While the two curves exhibit similar behavior for small values of the aggregation scale, the asymptotic slope of the PBAC plot is very different from the MBAC one, suggesting that the MBAC-controlled traffic is not self-similar (H≈0.5). We recall that the asymptotic slope β is related to H by $\beta=2H-2$. The lines corresponding to H=0.5, $H=0.55$, H=0.75 and H=0.8 are plotted in the figure as reference comparison.

The Hurst parameter estimates are reported in table 3 and table 4, with the corresponding CAC settings (the maximum connections number for PBAC and the maximum link utilization for MBAC), and the achieved link utilization. For the wavelet estimates, 95% confidence intervals are also indicated (see section §2.2.3.3). The three methods described in section §2.2.3.3, provide congruent estimates. Results are impressive, and show that the Hurst parameter decreases from about 0.75, in the case of PBAC, to about 0.5 for MBAC. It is interesting to note that 0.75 is the Hurst parameter value theoretically calculated in [48], [103], [69] under different assumptions, when a flow has heavy-tailed periods of activity/inactivity with a shaping parameter c = 1.5 (the formula is $H = (3-c)/2$). We note that, as expected, the Hurst parameter does not depend on the link utilization. Finally, table 5 reports the Hurst parameter estimate obtained via the Wavelet method, when a finite and short buffer is employed. Even if these results relate to a short buffer scenario (100 packets instead of infinite), the values reported in table 5 for H are very similar to those of tables table 3 and table 4, where an infinite buffer size was adopted. In conclusion table 4 and table 5 quantitatively supports our thesis that self-similarity is a marginal phenomenon for MBAC controlled traffic (the achieved Hurst parameter is very close to 0.5, which represents Short Range Dependent traffic).

Threshold (connections)	Throughput (%)	Hurst (Variance)	Hurst (R/S)	Hurst (Wavelet)
105	71.8	0.73	0.79	0.78 [0.74, 0.82]
115	78.3	0.74	0.78	0.80 [0.76, 0.84]
125	84.5	0.71	0.79	0.75 [0.71, 0.79]
130	88.7	0.78	0.76	0.75 [0.71, 0.79]
135	91.7	0.72	0.72	0.77 [0.74, 0.81]
140	94.7	0.78	0.80	0.74 [0.70, 0.78]

table 3 - Hurst-parameter estimate for PBAC controlled traffic (infinite buffer size)

Threshold (% utilization)	Throughput (%)	Hurst (Variance)	Hurst (R/S)	Hurst (Wavelet)
70	69.1	0.55	0.48	0.55 [0.51, 0.58]
78	76.9	0.58	0.54	0.58 [0.54, 0.62]
86	84.5	0.55	0.51	0.60 [0.56, 0.64]
90	88.5	0.60	0.52	0.57 [0.53, 0.60]
94	92.4	0.51	0.46	0.56 [0.52, 0.60]
96	94.3	0.58	0.52	0.58 [0.54, 0.62]

table 4 - Hurst-parameter estimate for MBAC controlled traffic (infinite buffer size)

Threshold (% utilization)	Hurst (PBAC)	Hurst (MBAC)
70	0.75 [0.72, 0.80]	0.51 [0.47, 0.55]
80	0.74 [0.70, 0.78]	0.60 [0.56, 0.63]
85	0.75 [0.71, 0.79]	0.51 [0.48, 0.55]
90	0.75 [0.71, 0.79]	0.51 [0.48, 0.55]
94	0.79 [0.75, 0.82]	0.51 [0.48, 0.55]

table 5 - Hurst-parameter Wavelet estimate for PBAC and MBAC, using a 100 packets buffer

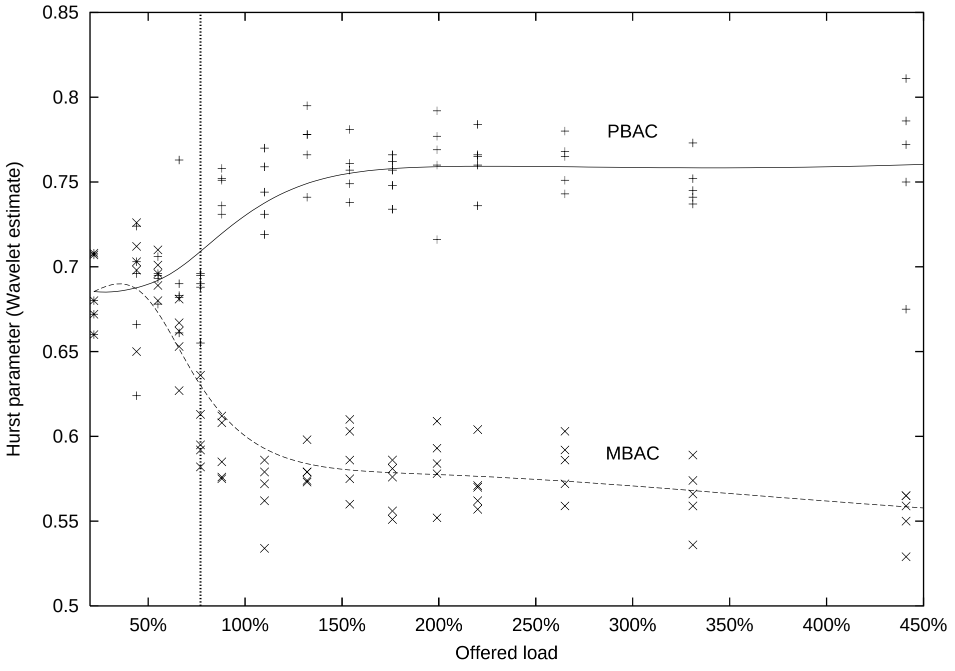


figure 24 - Hurst parameter vs. offered load

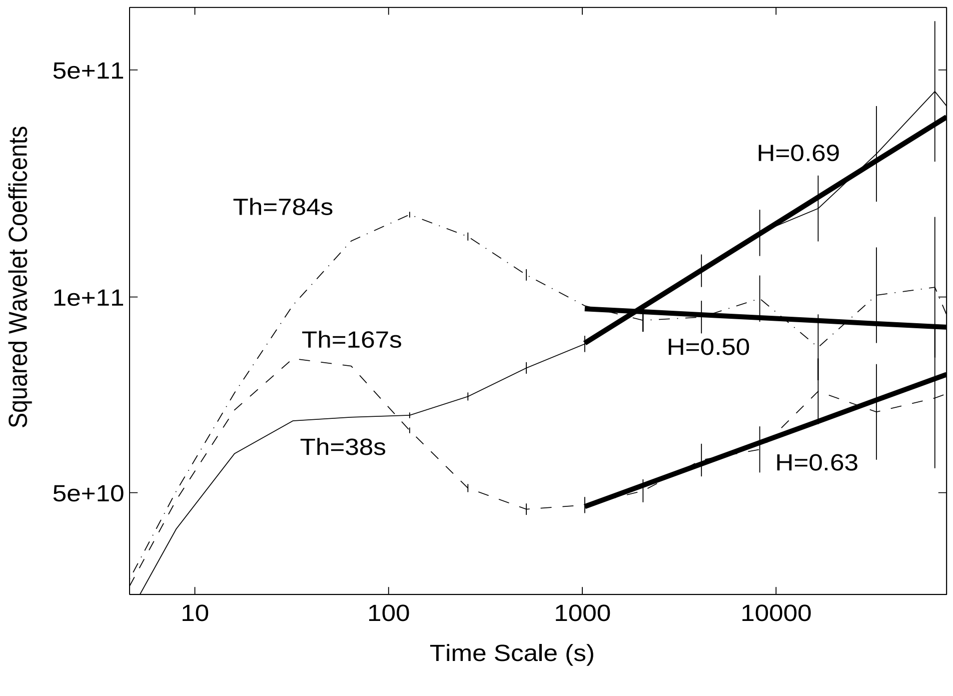


figure 25 - Wavelet coefficients plot for different holding time

Most of the previous results were obtained under constant overload conditions. The aim of the figure 24 is to show the behavior of the two CAC schemes in a wide range of offered load conditions, with a fixed target link-utilization (the thresholds chosen, 110 connection for PBAC and 77% for MBAC, give very similar throughput performance under the same offered load). The vertical dashed line corresponds to this target. Hurst parameter was estimated by the wavelet estimator. When the offered load is below the target, the Hurst parameter estimates[11] are quite similar because MBAC and PBAC do not enforce any rejection. By the way, in this situation, no need of access control arises and delay/loss performance figures cope with high QoS requirements. Instead, the effect of CAC rules becomes evident when the offered load exceeds the target utilization: the PBAC curve approaches to H=0.75**,** while the MBAC one declines and approaches to non LRD values. Moreover, the uncertainty of statistical results is shown by plotting several points for each simulated scenario, obtained with different seeds for the random generator.

The impact of the connection duration is drawn in figure 25. The ability of reducing traffic LRD is more effective as the duration increases. In fact, MBAC measurements are not able to efficiently track traffic variability when the holding time is too short in comparison to the filter memory.

§2.3 Conclusions

After a presentation of the mostly adopted admission control schemes, in this chapter we focused on the features of MBAC schemes and a comparison with traditionally adopted PBACs is also highlighted. In particular we presented some experimental results about the behavior of PBAC and MBAC schemes in case of Markovian and, mostly important, self-similar traffic generated by *On/Off* sources. In the former case, the PBAC performance slightly outperforms MBAC results, but no significant differences can be noted using MBAC or PBAC, especially if the ratio between the traffic source rate and the overall link capacity is small. Conversely, our simulation results show that the LRD of the traffic aggregate in a MBAC context is marginal, particularly when compared with that resulting from a traffic aggregate accepted by a PBAC scheme. For this reason MBAC allows to minimize the performance impairments due to the bursty nature of self-similar flows, thus making the system robust to statistical traffic properties. A greater QoS can be achieved using MBAC rather than PBAC schemes in order to control LRD flows.

We feel that there are two important practical implications of our study. Firstly, our study support the thesis that MBAC is not just an approximation of PBAC, useful when the statistical pattern of the offered traffic is uncertain. On the contrary, we view MBAC as a value-added traffic engineering tool that allows a significant increase in network performance

[11] In accordance with the fit test of the MatLab tool, the LRD hypothesis on the range from 2000 to 250000 seconds should not be rejected.

when offered traffic shows LRD. Secondly, provided that the network is ultimately expected to offer an admission control function, which we recommend should be implemented via MBAC, our results seem to question the practical significance of LRD, the widespread usage of self-similar models in traffic engineering, and the consequent network oversizing.

Chapter III

Support for multiple-levels QoS-needy unicast applications in DiffServ networks

The Differentiated Services architecture and the application of measurement based admission control schemes constitute a concrete framework for QoS enabled network services. RSVP protocol and the Integrated Services architecture are a valid alternative, but they present serious scalability and configurability drawbacks, which, conversely, are the better advantages rising from the adoption of DiffServ and MBAC. The point is how to enforce MBAC schemes, and admission control in general, in a legacy DiffServ implementation. Our purposes, hereinafter reported, is to change the network deployment in a evolutionary fashion, rather than with a revolutionary approach. So we propose to gradually enhance the network functionalities by adopting end-to-end admission control schemes and by upgrading core network routers with measurement modules to replace normal queue-length based active queue management mechanisms. We will show that this kind of approach promises a very effective enhancement of network performance, so that the network itself could be ready to: i) actually differentiate the service level; ii) give concrete QoS guarantees, even though in a statistical fashion; iii) support easy management and configuration procedures. Hence our proposal envisions a considerable revenue in terms of capability of the network to host high data rate applications with particularly stringent delay and jitter requirements, as in the case of video and, in general, multimedia and real time streaming.

Thus, in this and in the following chapter, a thorough analysis of a particular mechanisms is proposed, similar to endpoint admission control, and whose name is GRIP. Firstly we provide a presentation of the GRIP philosophy and we show how it somehow inherits some characteristics from EAC schemes and how it additionally complements the EAC operation with a mechanism that resembles MBAC operation at each network node. It will be also shown that GRIP can operate over a common DiffServ infrastructure, even though the effectiveness of the overall service support depends on the possibility to slightly modify the traditional packet dropping scheme adopted, but not in a mandatory way, over commercial and commonly envisioned routers. We will qualitatively and quantitatively discuss the impact of GRIP over network performances, also by means of a comparative study on the GRIP admission control features implemented via existing and proposed traffic handlers.

The first part of the chapter describes GRIP and its features (section §3.1), while the match of GRIP and DiffServ semantics, in given in section §3.2, with particular reference to the Assured Forwarding PHB. Afterwards, in section §3.3 it is deepened the question of QoS level that GRIP can provide in its different implementations, and several simulative results are presented. Section §3.4 additionally presents some application programming interfaces that allow a real and effective GRIP over DiffServ implementation (and more). Real performance are presented in section §3.5, while section §3.6 concludes and summarizes the chapter.

§3.1 GRIP

A particular set of EAC proposals are those in which some level of cooperation is required by core routers, and depending on the level of cooperation and intelligence required in network routers a finer QoS support is possible. The attempt is to convey more reliable network state information to the edge of the network. In such a view, EAC can be supported by a MBAC that runs internally to the network (i.e., in a whole domain or in each internal router or in the edge devices). This MBAC scheme locally determines, according to some specific criteria (which can be as simple as non performing any measure at all, and taking a snapshot of the link state, or as complex as some of the techniques proposed in [21], [47]), whether a new connection can be locally admitted.

As previously stated in §2.1.3, a number of proposal have been suggested as EAC schemes, taking into account the different aspects of the QoS provisioning. Here we treat a specific reservation framework where EAC decisions are driven by probing packet losses occurring in the internal network routers. The name of the proposed mechanism is GRIP (Gauge&Gate Reservation with Independent Probing). The Gauge&Gate acronym stems from the assumption that network routers are able to drive probing packet discarding (Gate) on the basis of accepted traffic measurements (Gauge), and thus implicitly convey congestion state information to the edge of the network by means of reception/lack of reception of probes independently generated by edge nodes. The GRIP endpoint operation is not different from a mere EAC but in the GRIP mechanism core routers are able to drive endpoint admission control by implicit signaling. The routers only have to distinguish probe and data packets, without any comprehension of the signaling semantic inside the packet payload, and to effectuate some running measurements over the network state. A probe packet arriving at the router indicates a request for a new connection, the router, basing on its network-state knowledge, can decide to drop the packet or to forward it. If the probe packet is discarded, the source will not receive a feedback and it is made able to implicitly determine that at least one router along the path has declared itself not capable of accommodating additional flows. According to the endpoint behavior previously described, the flow setup attempt will be aborted. The following figures show the GRIP mechanism in cases of acceptance and reject:

A unidirectional flow connection setup is considered in figure 26 as well as in figure 27. Hence, if bidirectional connection need to be established, the GRIP mechanisms can be separately adopted in the forward and in the return path. In any case, the local MBAC decision is driven by a specific congestion threshold for each network interface. These thresholds are freely set by each domain administrator depending on its provisioning target.

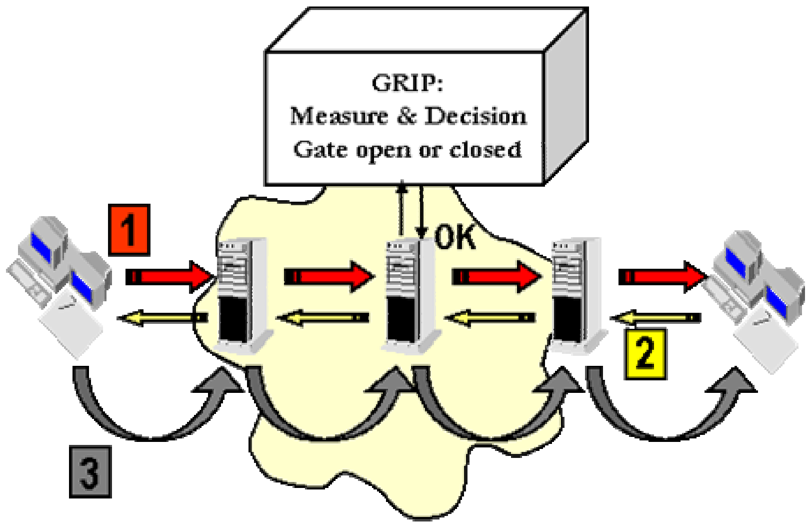


figure 26 - Basic model of GRIP (ACCEPT state)

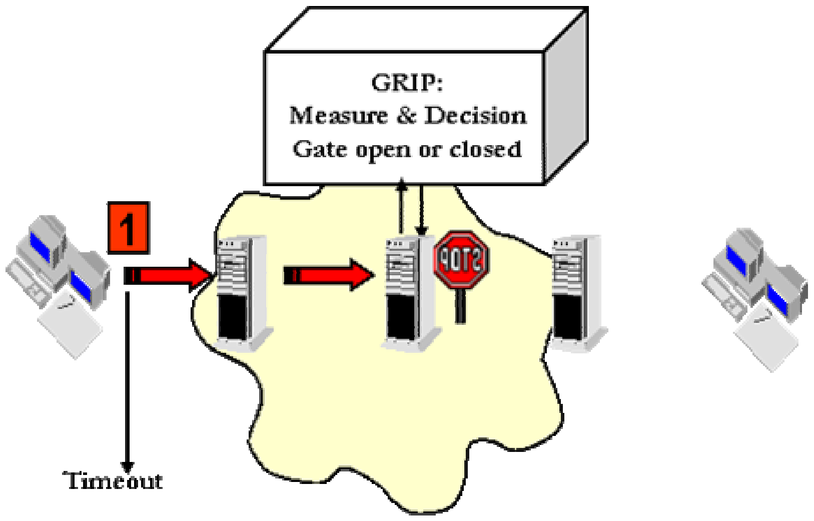


figure 27 - Basic model of GRIP (REJEC state)

§3.1.1 End Point Operation

The source node, triggered by the application via proprietary signaling, starts a connection setup attempt by sending in principle just one single packet, labeled as "*probe packet*" (the GRIP) probe. At the same time, a probing phase timeout is started. The role of the destination node simply consists of monitoring the incoming IP packets and detecting those labeled as probe. Upon reception of a probe packet, the destination node, if it wants to accept the connection, simply will relay a feedback packet (the GRIP ack or GRIP feedback). The decision whether to admit or reject the connection request is driven by the eventual reception of the feedback by the source. When a feedback packet is received in response before timeout expiration, the setting up flow is elected at the state of accepted, and the source node can start transmitting information packets, labeled as data packets. For unidirectional data flows feedback packets are labeled as data packets because the only goal of the feedback is to report back to the source the correct reception of the probe, while for bi-directional flows they could be labeled as probe packets because they aim also to test the reverse path.

Our solution seems then perfectly compatible with existing applications. In addition, this scheme leaves the service provider free to provide optional implementation details, including:

- definition of more complex probing phase operation, e.g., by including reattempt procedures after a setup failure, multiple timers and probes during the probing phase, etc;
- addition of proprietary signaling information (e.g., for capability negotiation) in the probing packet payload or in the feedback packet payload, to be parsed, respectively, at the destination node or at the source node.

So capability negotiation, recognized as an important functionality for QoS enabled applications, is in an important by-product of our solution. Note also that the basic GRIP operation, i.e., a single probe packet and a single feedback packet, is compatible with the H.323 call setup scheme using UDP, which encapsulates a H.225.0v2 call setup PDU into a UDP packet. As a last consideration, it is quite interesting to remark that this idea is extremely close to what TCP congestion control technique does, but it is used in the novel context of admission control: end points interpret failed receptions of probes as congestion in the network and reject the relevant admission requests.

§3.1.2 Router Operation

The endpoint operation is not different from a mere EAC but in the GRIP mechanism core routers are able to drive endpoint admission control by implicit signaling. The routers only have to distinguish probe and data packets (without any comprehension of the signaling semantic inside the packet payload) and to effectuate some running measurements over the network state. A probe packet arriving at the router indicates a request for a new connection, the router, basing on its network state knowledge, can decide to drop the packet or to forward it. If the probe packet is discarded, the source will not receive a feedback and it will be made able to implicitly determine that at least one router along the path has declared itself not capable of accommodating additional flows. According to the endpoint behavior previously described, the flow setup attempt will be aborted.

§3.2 GRIP over AF DiffServ

The Assured Forwarding Per Hop Behavior (AF PHB) has been devised by the IETF DiffServ Working Group to provide drop level differentiation. The intent of AF is to support services with different loss requirements, but with no strict delay and jitter guarantees. Another suggested use of AF is to provide differentiated support for traffic conforming to an edge conditioning/policing scheme with respect to non-conforming traffic. Now we show that, quite surprisingly, a standard AF PHB class is semantically capable of supporting per flow admission control. This is obtained by adopting the AF PHB as core routers forwarding mechanism in conjunction with an EAC mechanism running at the network edge nodes. The performance achieved by our proposed approach depend on the specific AF PHB implementation running in the core routers. We also prove that changes in the customary AF PHB implementations are indeed required to achieve strict QoS performance. To prove this point, we have evaluated the performance of GRIP through a simple and legacy AF implementation based on RED queues. Our results show that, regardless of the selected RED thresholds configuration, such an implementation is never capable of guaranteeing tight QoS support, but is limited to provide better than best effort performance. Additionally, by substituting RED active queue management mechanism with in principle not more complex measurement based dropping algorithms, we show that network performance can be considerably enhanced. Guideline for GRIP router implementation over Linux operative systems is eventually given.

§3.2.1 DiffServ semantic and GRIP

Quoting the recent RFC 2990 [52], *"both the Integrated Services architecture and the Differentiated Services architecture have some critical elements in terms of their current* *definition, which appear to be acting as deterrents to widespread deployment... There appears to be no single comprehensive service environment that possesses both service accuracy and scaling properties*". In fact, the IntServ/RSVP paradigm [18], [105] is devised to establish reservations at each router along a new connection path, and provide hard QoS guarantees. In this sense, it is far to be a novel reservation paradigm, as it inherits its basic ideas from ATM and the complexity of the traffic control scheme is comparable. In the heart of large-scale networks, the cost of RSVP soft state maintenance and of processing and signaling overhead in the routers is significant and thus there are scalability problems. In addition to complexity, we feel that the lack of a total and ultimate appreciation in the Internet market of the IntServ approach is also related to the fact that RSVP needs to be deployed in all the involved routers, to provide end-to-end QoS guarantees; hence this approach is not easily and smoothly compatible with existing infrastructures. What we are trying to say is that complexity and scalability are really important issues, but that backward compatibility and smooth Internet upgrade in a multi-vendor Internet market scenario is probably even more important.

Following this line of reasoning, we argue that the success of the DiffServ framework [14] [75], does not uniquely stays in the fact that it is an approach devised to overcome the scalability limits of IntServ. As in the legacy Internet, the DiffServ network is oblivious of individual flows. Each router merely implements a suite of scheduling and buffering mechanisms, to provide different aggregate service assurances to different traffic classes whose packets are accordingly marked with a different value of the DSCP field in the IP packet header. By leaving untouched the basic Internet principles, DiffServ provides supplementary tools to further move the problem of Internet traffic control up to the definition of suitable pricing/service level agreements (SLAs) between peers. However, DiffServ lacks a standardized admission control scheme, and does not intrinsically solve the problem of controlling congestion in the Internet. Upon overload in a given service class, all flows in that class will suffer a potentially harsh degradation of service. RFC 2998 [23] recognizes this problem and points out that *"further refinement of the QoS architecture is required to integrate DiffServ network services into an end-to-end service delivery model with the associated task of resource reservation"*. It is thus suggested to define an *"admission control function which can determine whether to admit a service differentiated flow along the nominated network path"* [52].

Scope of this section is to show that such an admission control function can be defined on top of a standard DiffServ framework, by simply making smart usage of the semantic at the basis of the Assured Forwarding Per Hop Behavior (AF PHB [50]). It is obvious that this function must not imply a management of per flow states, which are alien to DiffServ and which would re-introduce scalability problems. Considering the GRIP as discussed above, one can note that:

- the scope of GRIP admission control function is Internet-wise (i.e., not limited to a single domain);
- GRIP is deployed by pure endpoint operation: edge nodes involved in a communication are in charge of taking an explicit decision whether to admit a new flow or reject it;
- these edge nodes rely upon the successful delivery of probe packets, e.g., packets tagged with a suitable DSCP label, independently generated by the endpoints at flow setup.

The internal differentiated management of probes and packets originated by already admitted flows could be performed in conformance with the AF PHB definition, and the GRIP would candidate to play the role of admission controller in DiffServ, although GRIP was originally proposed in [6] out of the scope of a DiffServ implementation. Also, the possible degree of QoS provided is delegated to each individual DiffServ domain, and depends on the AF PHB specific implementation and tuning done at each core router of the domain. However, we remark that, strictly speaking, GRIP is not a new reservation protocol for the Internet (in this, differing from the SRP protocol [3], from which GRIP inherits some strategic ideas). Instead, GRIP is a novel reservation paradigm that allows independent endpoint software developers and core router producers to interoperate within the DiffServ framework, without explicit protocol agreements.

§3.2.2 GRIP AF Router Operation

A particular implementation of the DiffServ router output-port operation supporting the AF PHB is depicted in figure 28. Packets routed to the relevant output are classified on the basis of their DSCP tag and dispatched to the relevant PHB handler. Let us now focus our attention to a specific module in charge of handling AF traffic belonging to a given class x.

A measurement module is devised to run-time measure the aggregate AF class x traffic (or AFx traffic). The measurement module depicted in the figure does not interact with the AFx1 packets forwarding, i.e., these packets are forwarded to the FIFO buffer placed at the output regardless of the measurements taken. On the basis of such measurements, this module triggers a suitable dropping algorithm on the AFx2 traffic. With respect to the general AF PHB operation, our AFx2 dropping algorithm depends on AFx traffic measurements.

The simplest dropping algorithm is represented by a gate (smoother dropping algorithms for AFx2 packets - e.g., RED-like algorithms - may be considered to improve stability). When the measurement module does not detect congestion on the AFx traffic, it keeps the gate opened (we call this *accept* state). When the gate is open no AFx2 packets are dropped. Conversely, the measurement module keeps the gate closed (*reject* state) when congestion is detected, i.e., it enforces a 100% drop probability over AFx2 packets. Note that this operation does not violate the AF drop level relationship, as AFx1 dropping probability is lower than the AFx2 one. Finally, with reference to figure 28, the AFx PHB class handler stores packets in a FIFO buffer, to ensure that packets are forwarded in the order of their receipt, as required by the AF PHB specification [50]. Packets transmission over the output link is eventually managed by a scheduler, which has the task of merging the traffic coming from the different PHB handlers implemented within the router output port.

From the above GRIP-oriented description, it is easy to see how the GRIP mechanism can be implemented over such AF routers. In fact, let us assume that:

- the considered AF class x is devoted to the support of QoS-aware flows, requiring AC;
- AFx1 packets are data packets, generated by flows which have already passed an admission control test;
- AFx2 packets are probe packets injected in the network by flows during the setup phase (in principle, one AFx2 packet per flow).

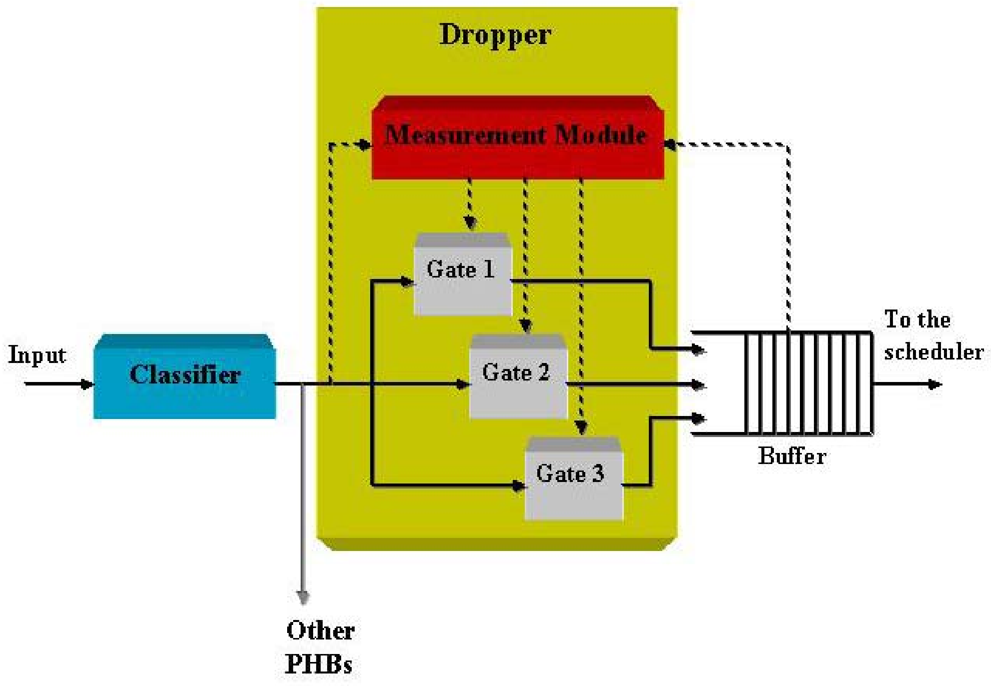


figure 28 - DiffServ Router supporting AF PHB

According to the described operation, an AFx2 packet is delivered to its destination only if it encounters all the routers along the path in the accept state. This operation provides an implicit binary signaling pipe, semantically equivalent to a one-bit explicit congestion notification scheme, without requiring explicit packet marking, or, worse, explicit signaling messages, contrary to the DiffServ spirit. Conversely, the described router output port operation, combined with an endpoint admission control logic, allows overlaying an implicit signaling pipe over a signaling-unaware DiffServ framework.

§3.2.3 Possible roles of the AFx3 level

In the above description, the AFx3 drop level appears in principle unnecessary. However, it may be convenient to use this level. A first possibility is that the AFx3 level be used to mark non-conforming packets, which can be eventually delivered if network resources are available. Secondly, AFx3 packet marking can be enforced over flows that have not successfully passed the described admission control test. This allows deploying a service model where high QoS is provided to flows that pass the admission control test, while best effort delivery is provided to initially-rejected flows. These latter flows may occasionally retry the setup procedure, by simply marking occasional packets as AFx2, and may eventually receive the upgraded AFx1 marking when network resources become available (as testified by the eventual reception of an AFx1 feedback). The usage of the AFx3 level as described above is targeted to increase the link utilization. However, [50] requires the drop probability for AFx3 to be greater (or at most equal) than AFx2. This implies that the link utilization is bounded by the possibly strict mechanism that triggers AFx2 packets dropping: when AFx2 packets receive a 100% dropping probability, all AFx3 packets must also be dropped to conform to the [50] specification.

A more interesting possible usage of the AFx3 level consists in providing a second control (probing) channel, in addition to AFx2. According to this solution, AFx1 traffic measurements trigger a dropping algorithm on the AFx3 traffic too, with stricter dropping conditions than the AFx2 dropping algorithm (i.e., AFx3 packets are assumed to detect congestion, and notify it via packet drop, before AFx2 packets). For instance, this AFx3 probing class could request admission for flows with higher peak rate and bandwidth requirements than flows supported via the AFx2 probing class. Also, being the AFx3 channel more reactive to congestion conditions, its usage can be envisioned to provide lower access priority to network resources. This would improve fairness and avoid some kinds of sources to “steal” a large part of network resources.

§3.2.4 Security considerations

As all admission control functions, our solution presents the risk of theft of resources through the unauthorized admission of traffic. In fact the user may bypass the admission control test and directly sends AFx1 packets. Administrators are then expected to protect network resources by configuring secure polices at interfaces (access routers) with not trusted customers. Similar protections must be provided at the interface between different domains. In particular, it may be necessary to restrict the access to the AF classes used for admission controlled traffic. For example, a DiffServ domain should remark AF packets when they come from an not trusted adjacent DiffServ domain. In more generality, we remark that policing and conditioning rules enforced at the border routers of each domain depend on the usage of the considered AF PHB class within the specific domain and thus have to be accounted of in the definition of each specific PDB supporting admission control.

A quite obvious security hazard is flooding the network with AFx2 packets. The objectives are twofold. On one side, denial of service situations can be easily created, as a massive loading of the network with AFx2 packets prevent the setup of normal connection. On the other side, the goal might be to affect fairness: the continuous transmission of AFx2 packets at a rate higher than normal connection requests is meant to gain faster access to resources when these are made available by a router along the path. This implies that some form of traffic conditioning and policing is necessary over AFx2 streams. While it is simple to recognize an hard attack, by monitoring the AFx2 packets crossing an edge router, it may be not straightforward for DiffServ boundary routers to recognize smoother fairness attacks. However, note that the same fairness problem is present also in more complex reservation mechanisms, such as RSVP where malicious users can continuously require setup to increase their access possibility with respect to normal users.

§3.3 The degree of QoS support in GRIP

Scope of this section is to provide some qualitative insights about the performance achievable by the GRIP operation. Quantitative and tunable performance may be independently provided and specified by each administrative entity. Uniform implementation across a specific domain allows defining a quantitative view of the performance achievable within a considered DiffServ domain, similarly to PDB operation. In this way, the refinements deemed necessary in [52] to provide service accuracy in the DiffServ architectural model could be considered as accomplished.

In fact, the performance achievable by the described endpoint admission control operation depends on the notion of congestion as the triggering mechanism for AFx2 packet discarding, which is left to each specific implementation. Each administrative entity may arbitrarily tune the optimal throughput-delay/loss operational point supported by its routers, by simply determining the aggregate AFx1 traffic target supported in each router. The mapping of AFx1 throughput onto loss/delay performance, in turn, depends on the link capacities and on the traffic flow characteristics offered on the AF class x. With this approach, it is possible to construct PDBs offering quantitative guarantees. A building block of such PDBs is the definition of specific measurement modules and AFx2 dropping algorithms. A generic dropping algorithm is based on suitable rules (or decision criteria). An example of a trivial decision criterion is to accept all AFx2 packets when the measured throughput is lower than a given threshold and reject all AFx2 packets when the AFx1 measurements overflow this threshold. The resulting performance depends upon the link capacity and the traffic model.

It is well recognized that target QoS performance can be obtained by simply controlling throughput (i.e., by aggregate measurements taken on accepted traffic). This principle is at the basis of more sophisticated state of the art MBAC implementations described in [21], [47]. As a simple quantitative example, with 32 Kbps peak rate Brady *On/Off* voice connections offered to a 2 *Mbps* (20 *Mbps*) link, a target link utilization of 75% (92%) leads to a 99th percentile per hop delay lower than 5 *ms* - see figure 29 (reproduced from [8]).

Tighter forms of traffic control are possible. As a second example of a decision criterion, it has been demonstrated that hard QoS guarantees can be provided, under suitable assumptions on the offered traffic (i.e., traffic sources regulated by standard Dual Leaky Buckets, as in the IntServ framework) and with ad hoc defined measurement modules in the routers [6]. When hard QoS guarantees are addressed, it is furthermore necessary to solve the problem of simultaneous activation of flows. In fact, the Gauge&Gate operation implies that simultaneous setup attempts (i.e., probes arriving at the router within a very short time frame) may see the router in the accept state, and thus may lead to concurrent activation of several flows, which can, in turn, overload the router above the promised QoS level. Actually, this is a common and recognized problem of any MBAC scheme that does not rely on explicit signaling. Although it does not compromise the stability of described operation (overloaded routers close the gate until congestion disappears - see the mathematical formalization of such a problem and the computation of the "remedy" period in [47]), this can be a critical issue if strict QoS guarantees are aimed at. It is possible to solve this problem by introducing an aggregate stack variable [6], which takes into account "transient" flows, that are flows elected at the state of *accepted* but not yet emitting information packets. This stack protection scheme avoids the concurrent activation of a number of flows, which could overload the router above the promised QoS level: the price to pay is a slight under-utilization of the available link capacity. The same effect can be obtained in a traffic load estimation environment by artificially augment the estimate with a term that takes into account the traffic that could be transmitted later by a new admitted source. This estimation bias will be filtered out in a time that is proportional to the digital filter memory.

Another important issue is what happens when traffic flows with widely different peak rates are offered to the same link. Several approaches may be adopted. The simplest one is to differentiate traffic aggregates into classes of similar traffic characterization (in terms of rate), and associate to each class a different AF PHB. A second possibility is to multiplex traffic onto a same AF PHB class, but differentiate probing packets by using the AFx3 drop level for traffic flows with higher peak rate.

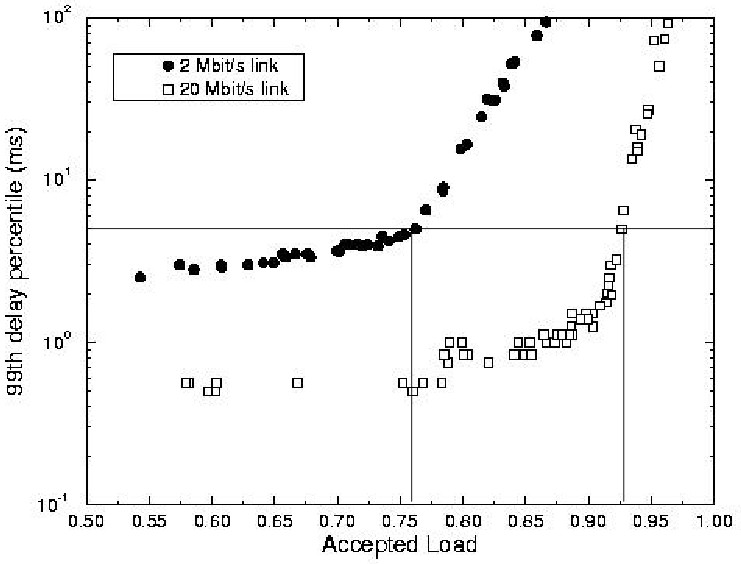


figure 29 - 99th delay percentile versus throughput, for different EAC schemes and related parameter settings

§3.3.1 GRIP over pure DiffServ (RED/RIO queues)

GRIP becomes effective as long as two conditions jointly occur. Firstly, each DiffServ router along the path must distinguish a probe from an information packet. This is achieved by marking probe packets with a different DiffServ label than data packet. Secondly, the performance of GRIP is related to the capability of routers to locally take decisions about the degree of congestion, and suitably block probing packets when congestion conditions are detected. To this purpose, the congestion level can be computed via local traffic measurements on the accepted traffic [7]. The described operation translates the problem of provisioning a given QoS level within a domain, into the much more simple problem of suitably configuring a packet forwarding/dropping mechanism within each router in a domain. For example, a domain can configure its routers so that all AFx2 packets are dropped (i.e., all incoming connections are blocked) when the average AFx1 traffic exceeds a given threshold. The higher the threshold, the worse the QoS performance provisioned. However, RED-like mechanisms are usually deployed, as shown in following subsections.

§3.3.1.1 RED/RIO queue management

Traditionally, a FIFO policy has been used in router queues. In this case, if the input rate is greater than the output rate, packets are stored in the queue until it gets full: at this point, incoming packets may be discarded. Although this method is very simple, it has some drawbacks. Firstly, it can be very unfair: sources producing large bursts are more penalized than quasi-constant sources. Secondly, a large number of consecutive packets may be dropped, making many TCP connections switch to the Slow Start phase, which can lead to synchronization problems. Finally, heavily loaded queues result in bigger delays. There exist several proposals (see [43], [64], [73], [88] and [100]) that address the problems of FIFO queuing. In these proposals, a transmission protocol running at the endpoints of a connection controls the emission rate, in order to reduce the occupation of router queues. Such solutions, however, are difficult to put into practice and their efficiency is somewhat questionable. The best place to control the evolution of a queue of a router seems to be the router itself [43]. In a DiffSer router, packet dropping is primarily managed by a specific queue management function that, within a given class, decides which packets should be discarded when congestion arises. That is, DiffServ adopts a selective discarding policy, especially in the context of the Assured Forwarding (AF) behavior.

The Random Early Detection (RED) mechanism [43], proposed by S. Floyd, is an efficient queue management method that has become very popular. Using a fairly simple algorithm, RED succeeds in keeping queue occupation low, while being fair with respect to packet loss. RED can be easily modified so as to discard packets according to their priority: WRED (Weighted RED) and RIO (RED with In and Out) are two instances of such an algorithm that will be described later in this section. RED is intended to deal mainly with persistent congestion in a queue. Its packet discard function is based on the average queue size, so that instantaneous fluctuations present in highly bursty traffic do not have a strong impact in the performance of the algorithm. In its definition [43], RED does not impose any formula for computing the mean queue occupation. However, examples in the literature always use a low-pass filter where the average queue size (*avg*) is computed as follows:

- when the routes is switched on: $avg = 0$;
- when a new packet arrives,
    - if the queue is not empty: $avg = (1-w)\, avg + w\, q$;
    - else: $avg = (1-w)\, m\, avg$;

where m is the queue empty period, and w is the weight assigned to each new measure. Remark that this pseudo-code takes into account those periods during which the queue remains empty, with the assumption that m minimal-sized packets might have been transmitted during that interval. Based on the mean queue size (*avg*), RED uses a non-decreasing drop-probability function to decide whether a packet should be discarded or not, and, thanks to this feature, consecutive packet drops are avoided[12]. Random discards are a fair solution, since the probability of dropping a packet of a given flow is proportional to the number of packets of that flow that are in the queue. In RED, packets are randomly discarded only if the average queue size is between two thresholds th_{min} and th_{max} , as shown in the figure 30.

Actually, the probability of dropping two consecutive packets is reduced by means of the following algorithm:

- when the routes is switched on: $count = 0$;
- when a new packet arrives:
    - If $avg \in [th_{\min}, th_{\max}]$ then
        - increment *count* by *1*;

[12] Remember that the loss of several consecutive packets of a connection mat trigger TCP Slow Start mechanism.

        - Set the dropping probability $P_b = P_{\max}\frac{avg - th_{\min}}{th_{\max} - th_{\min}}$;
        - Drop the packet with probability $P_a = \frac{P_b}{1 - count \cdot P_b}$;
    - Else if $avg > th_{max}$, then drop the packet;
- If the last packet is dropped, then set *count=0*;

The variable count contains the number of packets that were accepted in the queue after the last packet-drop event. Hence, this variable permits to reduce the probability of dropping consecutive packets. Note that the probability of discarding the packet increases linearly from *0* to P_{max} as the mean queue size goes from th_{min} to th_{max}, as also shown in figure 30. If the mean queue size is greater that th_{max}, then *all* packets are discarded[13]. If sources react adequately, the mean queue size should rapidly decrease to an acceptable value.

The operation of RED has served as a model for designing methods aiming to improve the performance of TCP, notably ECN (Early Congestion Notification). When this mechanism is used, a TCP source is informed of the congestion by means of a flag in the IP header of the ACK segment. In this case, RED does *not* discard any packet: instead, it sets the flag to tell the source to reduce its emission rate. Notice that ECN is orthogonal to DiffServ. A detailed description of ECN can be found in [84].

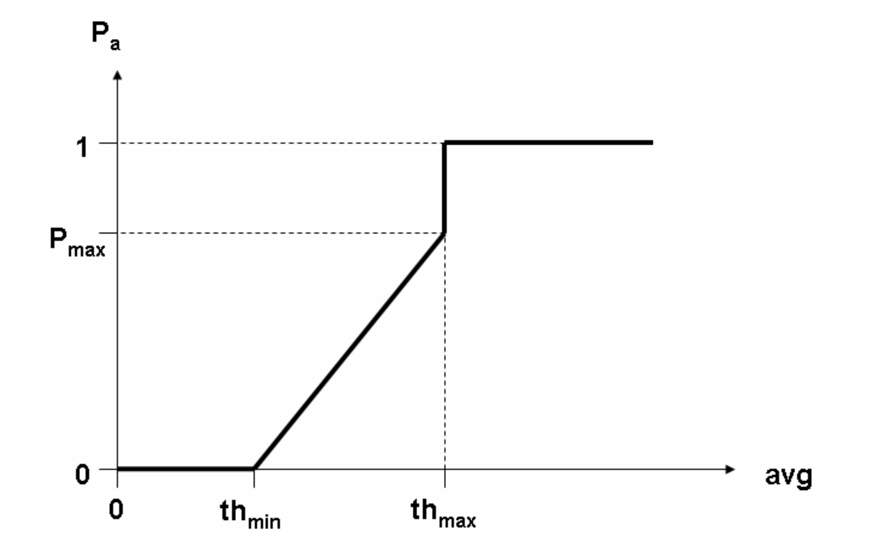


figure 30 - RED drop probability function

[13] Alternatively, the dropping probability could linearly grow from P_{max} to 1 as the queue estimate grows from th_{max} to 100%.

§3.3.1.2 Weighted RED (WRED)

WRED, originally proposed by Cisco, consists in using different sets of RED parameters for different sets of flows. By assigning different values of P_{max}, th_{min} and th_{max}, to each set, it would be possible to offer a differentiated distribution of packet loss [27]. In the Assured Forwarding behavior, the router must discard packets depending on their drop precedence, often referred to as *green* (AFx1), *yellow* (AFx2) and *red* (AFx3). WRED can be used for this purpose, by using three different sets of parameters. In WRED, the mean queue size is computed as in RED. When a packet arrives, the variable *avg* is always updated, no matter the precedence of the packet. However, the computation of P_a is different than in RED. figure 31 illustrates a set of parameters that could be used in the DiffServ architecture. For high-precedence packets (*red* packets), the algorithm follows an aggressive drop policy: as soon as the average queue size gets higher than a (fairly low) value, packets are not allowed to transit through the router. On the other hand, for low-precedence (*green*) packets the mechanism authorizes the building up of a much higher queue, following the principle that, in a well-dimensioned network, such packets should never be discarded.

§3.3.1.3 RIO (RED with In and Out)

RIO is one of the first algorithms introduced for service differentiation purposes. In the original proposal, Clark and Fang [29] describe a whole architecture that served as a reference for DiffServ-related work in the IETF. RIO was first conceived as means to differentiate between two precedence levels (called *In* and *Out*), but this model can be extended to accommodate more than two levels. In RIO, differentiation is partially based on the mean queue size. Contrary to WRED, RIO calculates a mean queue size *for each precedence level:* that is, the mean avg_n is updated each time a packet of precedence less than or equal to n arrives at the queue. In the context of DiffServ, avg_0 reflects the mean number of *green* packets in the queue, avg_1 corresponds to the total number of *green* and *yellow* packets, and finally avg_2 takes into account all packets (as in RED). This makes possible to isolate the drop probability for each precedence level: for low-precedence packets, only an excess of packets having the same precedence level can produce a packet drop. For high-precedence packets, an excess of packets of any precedence level may yield a packet drop.

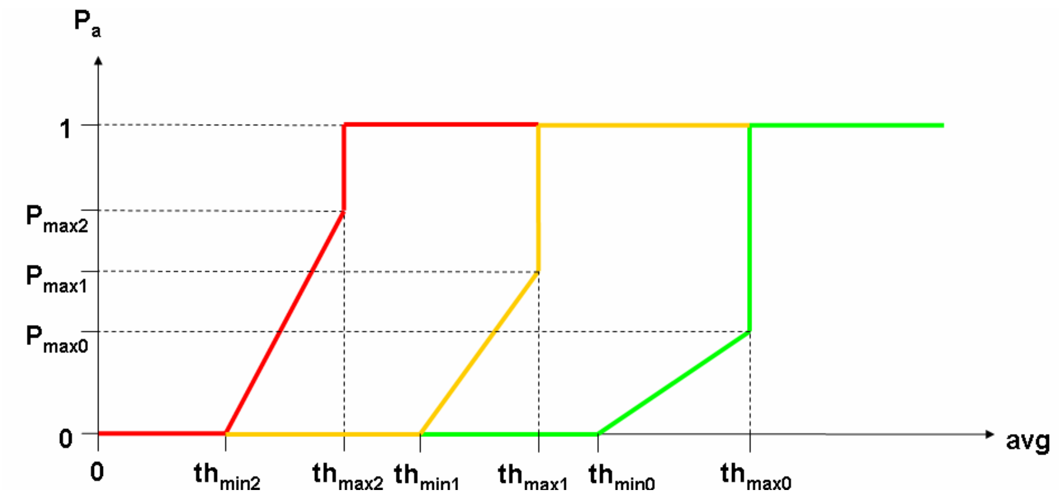


figure 31 - WRED drop probability functions

Clark and Fang [29] suggest also to use a different set of RED parameters (P_{max}, th_{min} and th_{max}) for each precedence level. Similar values as those used in WRED may be chosen. However, less aggressive threshold values could be used, since the mean queue size for high-precedence packets augments more rapidly than in (W)RED, due to the computation method employed. Differentiated computation of mean queue sizes makes RIO a powerful algorithm for precedence based packet discarding, but its implementation is more involved than that of WRED, which is already available in commercial routers [27].

§3.3.2 Performance of GRIP over legacy AF PHB (with RED/RIO)

In the previous paragraphs, we have concluded that arbitrary degree of performance guarantees can be obtained by designing specific AF PHB implementations based on runtime traffic measurements. A question that comes out naturally is the following. As long as Random Early Detection (RED) queue management is customarily considered as the natural AF PHB implementation[14], what are the performance of GRIP when it is operated over RED queues? We tried to answer the question by means of a simulated test bed.

In our simulation program, we have assumed, for convenience, only two drop levels, namely AFx1 and AFx2. We have adopted a single buffer for both AFx1 and AFx2 packets. AFx1 packets are dropped only if the buffer is completely full.

[14] We recall that the AF PHB specification does not recommend, by any means, a specific implementation. Indeed, RED have emerged as the natural AF PHB implementations, since they provide improved performance when TCP traffic (i.e., the traffic traditionally envisioned for AF) is considered. We now face a very different problem, i.e., what happens to performance when widespread RED AF PHB implementations are used for a completely different purpose, i.e., to support admission controlled (UDP) traffic.

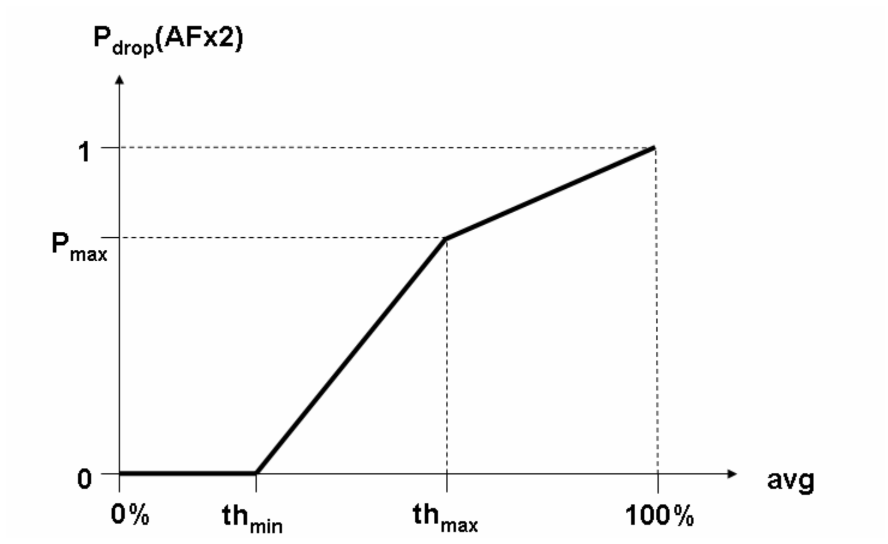


figure 32 - Buffer management scheme

Instead, AFx2 packets are dropped according to a RED-like management, where the AFx2 dropping probability is a function of the queue occupancy, computed accounting for both AFx1 and AFx2 packets[15]. As shown in figure 32, the AFx2 dropping probability versus the queue occupancy is a piecewise linear curve: no AFx2 packets are dropped when the number of packets stored in the queue is lower than a lower threshold. As the number of packets gets greater than the lower threshold, the AFx2 dropping probability increases linearly, until a value P_{max} is reached in correspondence with an upper threshold. After this value, the dropping probability is increased linearly until the number of packets fills the buffer capacity (see figure 32). Depending on the considered implementation, the AFx buffer occupation is either sampled at the arrival of an AFx2 packet, or suitably smoothed/filtered in order to capture the moving average of the AFx queue occupancy. However, in all implementations proposed in the literature, an AFx2 packet that finds no packets stored in the buffer is always accepted (regardless of the fact that the smoothed AFx queue occupation may give a value different from 0). We will see in what follows that, ultimately, this specific condition prevents GRIP to achieve effective performance, regardless of the RED parameter settings considered. Performance results have been obtained via simulation of a single network link, loaded with offered connections arriving at the link according to a Poisson process. Each offered connection generates a single probe packet. A sufficiently large probing phase timeout has been set to guarantee that connections are accepted when the probing packet is not dropped (in the simulation, we have attempted to simulate conditions as close to

[15] The described operations can also be seen as a particular case of a standard WRED implementation, where the T_{low} and T_{high} thresholds for AFx1 packets both coincide with the buffer size.

ideal as possible, in order to avoid that numerical results were affected by marginal parameters settings as round trip time, probing phase timer, etc). Accepted connections have been modeled as Brady *On/Off* sources, with peak rate equal to 32 *Kbps*, and *On/Off* periods exponentially distributed with mean value, respectively, 1.0 second for the *On* period, and 1.35 seconds for the *Off* period (yielding an activity factor equal to *0.4255*). Each connection lasts for an exponentially distributed time, with mean value 120 *s*. The link capacity has been set to 2 *Mbps*. Therefore, The link results temporarily overloaded when more than 146.9 active connections (=2000/(0.4255*32)) are active at a given instant of time.

The figure 33 to figure 35 report the number of accepted connections versus the simulation time for three different load conditions: underload (normalized offered load equal to 75% of the link capacity), slight overload (110% offered load), and harsh overload (400% load).

In the above figures, we have considered a very large AFx buffer size, such that no AFx1 packet losses occur (thus, QoS of accepted traffic is quantified in terms of AFx1 packet delay). We have tried several different RED parameters configuration, but, for simplicity of presentation, we report results related to a basic parameters settings where a single threshold is considered: all AFx2 packets are accepted whenever the number of AFx packets is lower than this threshold, while all AFx2 packets are dropped when the number of AFx packets is greater than this threshold (very similar results are obtained with more complex RED configurations, as it will be clear from the following discussion). Similarly, for simplicity, no smoothing on the buffer occupancy has been performed.

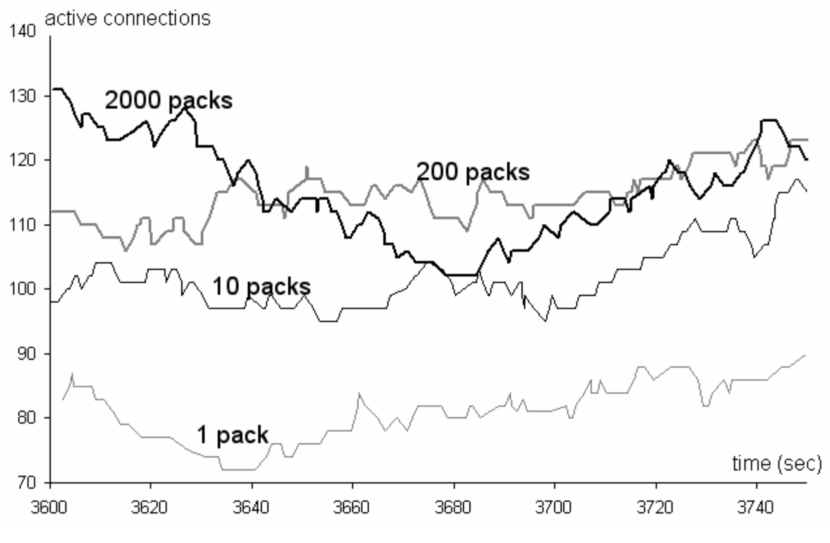


figure 33 - Active connection vs. simulation time, with 75% offered load; "packs" indicates the AFx2 threshold.

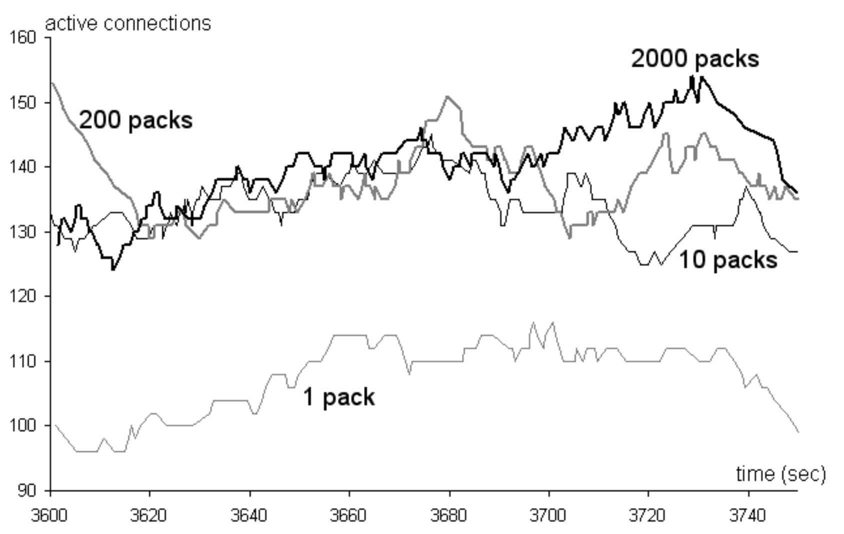


figure 34 - Active connections vs. simulation time, with 110% offered load. "packs" indicates the AFx2 threshold.

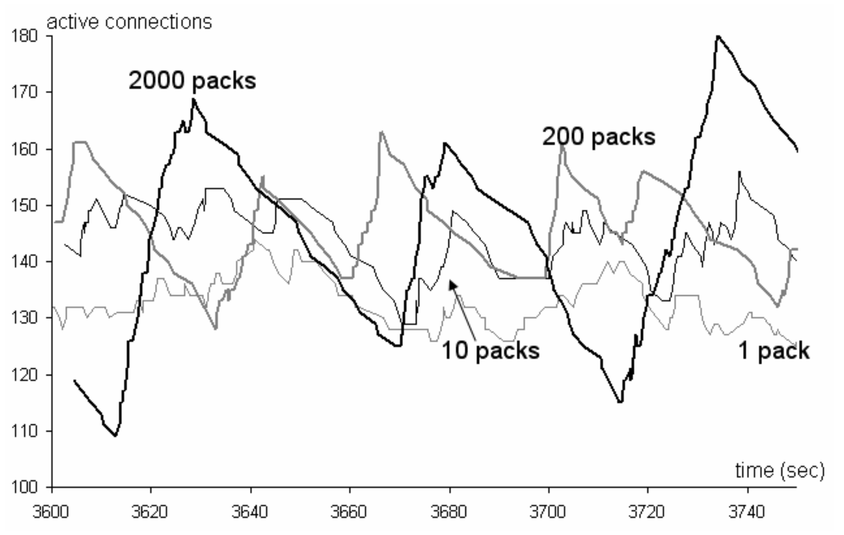


figure 35 - Active connections vs. simulation time, with 400% offered load. "packs" indicates the AFx2 threshold.

AFx2	Offered	Throughput	95th delay percentile	99th delay percentile
1 pack	110%	68,64% ± 0,28%	2,3 ± 0,014	3,3 ± 0,019
1 pack	400%	90,89% ± 0,08%	80,6 ± 1,2	175,2 ± 1,3
10 packs	110%	88,71% ± 0,22%	55,4 ± 0,7	148,444 ± 3.0
10 packs	400%	97,72% ± 0,03%	586,3 ± 4,8	987,2 ± 10,2
200 packs	110%	93,11% ± 0,28%	235,9 ± 4,0	433,8 ± 3,5
200 packs	400%	99,02% ± 0,02%	1282,8 ± 15,9	2023,3 ± 75,1
2000 packs	110%	96,39% ± 0,24%	1433,9 ± 20,0	1975,6 ± 26,9
2000 packs	400%	99,77% ± 0,03%	4656,0 ± 24,9	5921,8 ± 58,2

table 6 - Throughput and delay performance.

The figure 33 shows that, in low load conditions, a tight threshold setting (just one packet, i.e., an AFx2 packet is dropped as long as just 1 AFx packet is stored in the buffer) is overly restrictive, and exerts a high and unnecessary blocking probability on offered connections. With larger thresholds (200 and 2000 packets), we note from figure 33 that the number of accepted connections fluctuates in the range 100 to 120, meaning that just a small fraction of the offered connections are blocked by the GRIP operation (as it should ideally occur if a stateful admission control algorithm were operated).

Much more interesting and meaningful (for our purposes) are the results presented in figure 34. Here, we see that, in slight overload conditions, the only RED configuration setting that allows to keep the accepted load lower than the target 75% value (i.e., about 110 accepted connections) suggested by figure 29, is the threshold set to just 1 packet. Even with a very small threshold value, such as 10 packets, we see that the average number of accepted connections gets much greater than 110, thus resulting in unacceptable delay performance for accepted traffic.

Throughput and delay performance are quantified in table 6, for 110% and 400% offered load conditions. The table reports the AFx1 throughput, as well as the 95th and 99th delay percentiles experienced by accepted flows. Confidence intervals corresponding to a 95% confidence level are also reported in the table to quantify the accurateness of the numerical results.

From the table, we see that the only case in which we meet target QoS performance for IP telephony (i.e., 99th delay percentile of the order of few ms) is the case of threshold set to one packet, and light overload. It is quite impressive to note that the smallest possible RED threshold (i.e., drop an AFx2 packet whenever the AFx queue is not strictly empty) does not succeed in guaranteeing QoS in large overload conditions. This result allows us to conclude that, regardless of the RED parameters configuration, a RED implementation is never capable of guaranteeing QoS in all load conditions. As long as a RED implementation always accepts an AFx2 packet when the AFx queue is empty, performance will be at best equal to that reported in table 6, for a threshold equal to one packet.

As figure 35 shows, in high overload conditions, thresholds greater than 1 packet cannot even avoid that, temporarily, the number of accepted connections is greater than 146.9, i.e., that the link is overloaded. In such a case, load oscillation phenomena occur: as clearly shown in figure 35, the link alternates between periods of significant overload (in which the AFx buffer fills up), and remedy periods [47], where the router locks in the reject state, until congestion disappears. The result is that 95th and 99th delay performance are of the order of several seconds (see table 6).

To conclude this section, we observe that, although RED implementations are intrinsically not capable of providing performance guarantees, indeed a proper parameter setting allows to achieve reasonably better than best effort performance. For example, with a threshold set to 10 packets, table 6 shows that, in very high (unrealistic) load conditions, the 99th delay is still lower than one second (although the link has already been congested, as proven by the link load fluctuations, of the order of 15% of the link capacity, and leading to temporary link overload, as shown in figure 35). Notably, with the same 10 packets thresholds, the 99th delay percentile drops down to less than 150 *ms* when light overload conditions are considered. Such degree of QoS support might be considered sufficient in a short-term perspective, where current AF implementations might be utilized to support admission control.

§3.3.3 Performance of GRIP over MBAC-aware AF PHB

In this section we firstly show that performance effectiveness is achieved by means of non-traditional configuration of the forwarding mechanisms. The performed measures have the goal to confirm the results obtained trough simulation on GRIP and to prove the efficiency of this mechanism under real environments. We evaluate the performance of a more general RED-like dropping profile enforced on the AFx2 packet class. Specifically, when the load is below a th_{min} threshold, all the AFx2 packets are forwarded, while they are all dropped if the load is above a th_{max} threshold. Between these two thresholds, a linear dropping probability is enforced, i.e., being B the measured load, the dropping probability on the AFx2 packets is:

$$(19) \qquad P_{drop} = \frac{B - th_{\min}}{th_{\max} - th_{\min}}; \; with \;\; B \in [th_{\min}.th_{\max}].$$

B is conveniently obtained as a smoothed version of the instantaneous load. In particular, a typical numerical implementation that is widespread adopted consists of a discrete time scale, with sample time T = 100 *ms*, and, at each time step k, the smoothed offered load $B(k)$ is updated as follows:

$$(20) \qquad B(k) = \alpha B(k-1) + (1-\alpha)M(k);$$

where M(k) is the traffic load, in *bps*, attempting to enter the link buffer during the time slot k. By choosing α=0.99, a time constant of about 10 *s* time is obtained for the filter memory.

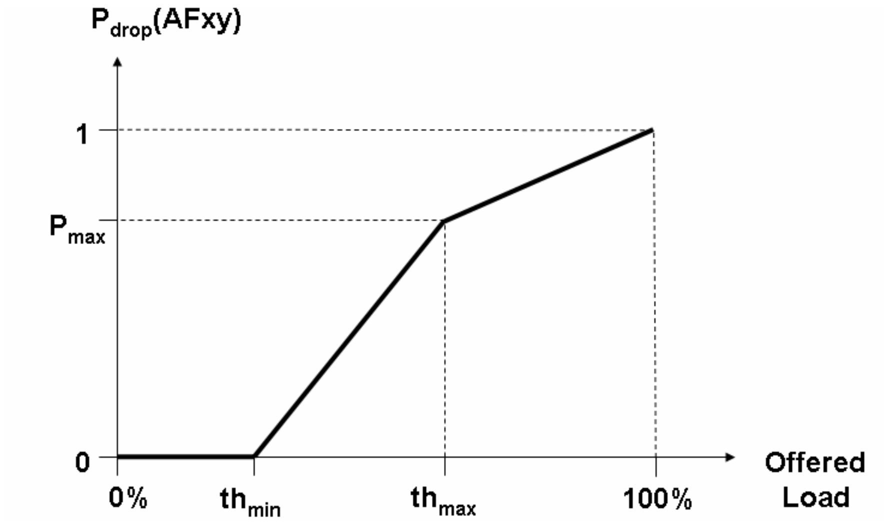


figure 36 - Measurement driven piece-wise linear dropping function

As illustrated in the following, the additional AFx3 drop level available in the AF-PHB specification may be optionally used (with an associated additional dropping profile) instead of AFx2 when a more congestion-sensitive probing is needed, i.e., when a connection requires a greater amount of resources.

§3.3.3.1 Simulated performance

Our simulation results have been obtained with a proprietary C++ event-driven simulator developed by our self. The simulation time was divided into 101 intervals: a first warm-up 100 minutes interval followed by other 100 intervals in which statistics are computed. For simplicity, as in many other admission control works [21], [47], the network model consisted of a single bottleneck link. The link capacity was set equal to 2 *Mbps*, and an infinite buffer size was considered. Since we used an infinite buffer size QoS is characterized by the delay experienced by data packets rather than packet loss as in [65]. 99th delay percentiles have been considered as delay performance metric (the same considerations apply as in §2.2.4). Simulation experiments were obtained in a dynamic environment consisting of randomly (Poisson) arriving flows. We have considered two different traffic scenarios. In the first scenario, flows are homogeneous. Each traffic source is modeled as variable bit rate (VBR) source, alternating heavy-tailed *On/Off* periods. While in the *On* state, a source transmits at a Peak Constant Rate (PCR) randomly generated in the small interval 31 to 33 *Kbps*. (to avoid source synchronization effects at the packet level). Conversely, while in the *Off* state, it remains idle. The mean value of the *On* and *Off* periods have been set, respectively, equal to 1.0 *s* and 1.35 *s*. To stress the system with long range dependent traffic, the duration of the *On* and *Off* periods have been drawn from two Pareto distributions, with the same shaping parameter c = 1.5 (infinite variance). The same *On/Off* profile has been considered in the second scenario, but the peak-rate of the sources has been uniformly generated in the range 20 to 80 *Kbps*.

In this second scenario, we have additionally considered the possibility to adopt, as *probe* label, the AFx3 packet marking for flows whose PCR was greater than 40 *Kbps*. Conformingly, we have set a more restrictive drop profile for the AFx3 drop level. The rationale behind this choice is that a flow with higher peak rate is more critical, in terms of resource consumption, than a flow with small peak rate. However, the described GRIP operation is not able to distinguish, during flow setup, flow with different peak rates (signaling is implicit - no traffic specification parameters are notified to the router). By using an additional AFx drop level, we are able to differentiate flows during their probing phase. In particular, the router has been configured so that a (high peak rate) flow that probes the network via an AFx3 packet will be blocked in advance with respect to a low peak rate flow.

The duration of an accepted flow is taken from a lognormal distribution [16] with mean 300 *s* and standard deviation 676 *s*, but connection duration is extended to the end of the last *On* or *Off* period. Because of this, the real connection lifetime exhibits longer mean (320 *s*) and infinite variance. The flow arrival process is Poisson with arrival rate λ connections per second. For convenience, we refer to the normalized offered load ρ, as already defined in equation (9). Depending on the simulation experiment, the arrival rate ranges from underload conditions (less than 20% of the link capacity) to severe overload conditions (up to 650%).

The specific MBAC implementation uses a discrete time scale, with sample time T = 100 *ms*. The connection is admitted with a probability $1\text{-}P_{drop}$, in accordance with (18) and using estimated bandwidth $B(k)$, as in (19) with α = 0.99. By tuning the parameters in (18) (i.e., th_{min} *and* th_{max}) performance figures can be obtained for various accepted load conditions.

figure 37 shows the different impact of homogeneous peak rate connections (~32 *Kbps*) on the system performance (leftmost plot), with respect to the heterogeneous peak rate case (PCR from 20 to 80 *Kbps*, rightmost plot). In both simulations the offered traffic is 90% of the link capacity. In the heterogeneous scenario, a visible degradation of performance occurs. There is an increase in bandwidth utilization variability, with instants of time in which the offered load is even higher than the link capacity.

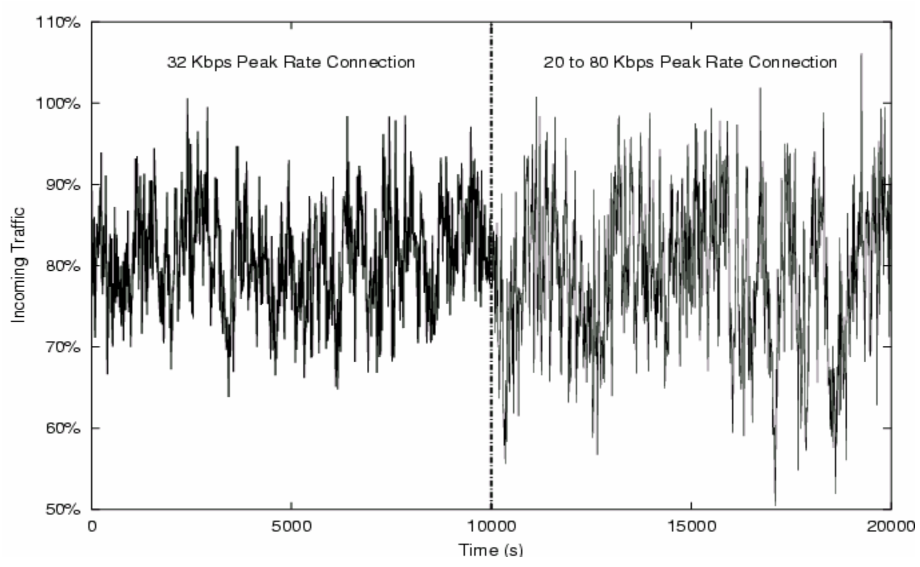


figure 37 - 10000 seconds temporal behavior comparison between uniform (~32 Kbps) and multiple peak rate sources (20 to 80 Kbps)

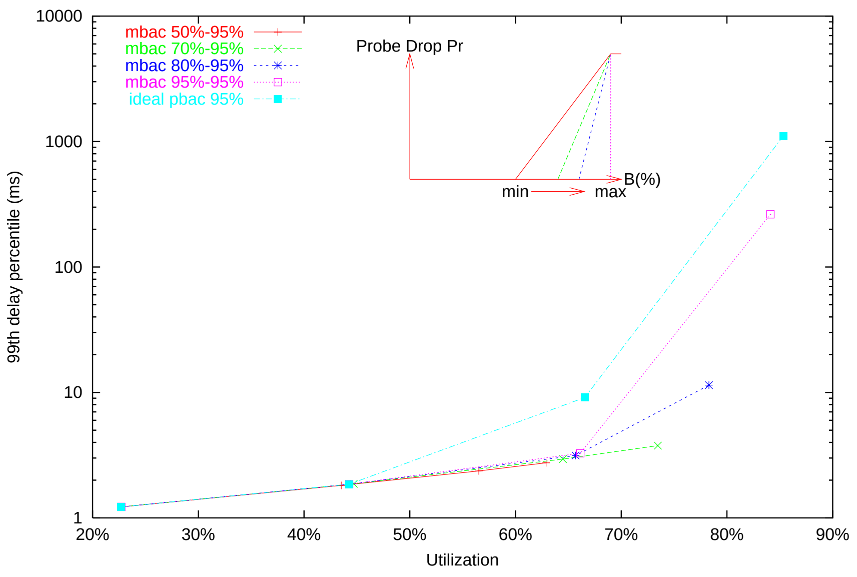


figure 38 - Offered load: 20% to 90%; source peak rate: ~ 32 Kbps

The throughput/delay performance of the adopted admission control scheme is evaluated in figure 38 to figure 42, for various packet dropping algorithm configurations, and 99th delay percentiles have been considered as delay performance metric. The figure 38 shows the sensibility of the admission control scheme as the dropping profile changes. The th_{max} threshold is fixed to 95%, while the th_{min} ranges from 50% to 95%. For each threshold configuration, different points are obtained by tuning the offered load from 20% to 90%. Better delay performance are achieved when the minimum and maximum thresholds (figure 36) are distant from each other. As a reference comparison, the figure reports also results obtained by assuming that a standard - parameter based - admission control mechanism is adopted. The fact that PBAC performs worse than a mechanism based on run-time measures, when traffic is LRD, has been thoroughly discussed and motivated in [11] and in the previous chapter. The performance of different threshold settings has been evaluated also in figure 39. This figure differs from the previous one in two fundamental aspects: firstly, the offered load has been set very high (about 650%); secondly, different throughput values have been obtained by suitably translating the thresholds, while maintaining their relative distance constant. Clearly a given setting performs better when, for a same carried load, the delay is lower. In figure 40 and figure 41, the same performance investigation is carried out considering sources with peak rate in the range from 20 to 80 *Kbps*. The AFx3 probe marking has been adopted for sources whose peak rate is greater than 40 *Kbps*.

The figure 40 reports results for various AFx2 dropping profiles configured with the th_{min} and th_{max} thresholds indicated in the label. The AFx3 dropping profile has been kept fixed with parameters th_{min} = 50%, th_{max} = 95%. In figure 41 the AFx3 dropping profile is obtained from the one of AFx2, by shifting back the th_{min} and th_{max} thresholds of 10%. Results are very different from the corresponding figure 39, and show that it is preferable to set the minimum threshold equal to the maximum one. The reason stays in the augmented traffic variance, due to the peak rate variability, which wastes the improved accuracy of the measurement based operation when separate thresholds are adopted. Instead, by setting tight thresholds, in very high load (650% in figure 41), the probability that a source with high peak rate gets accepted dramatically reduces, and thus the system achieves better performance.

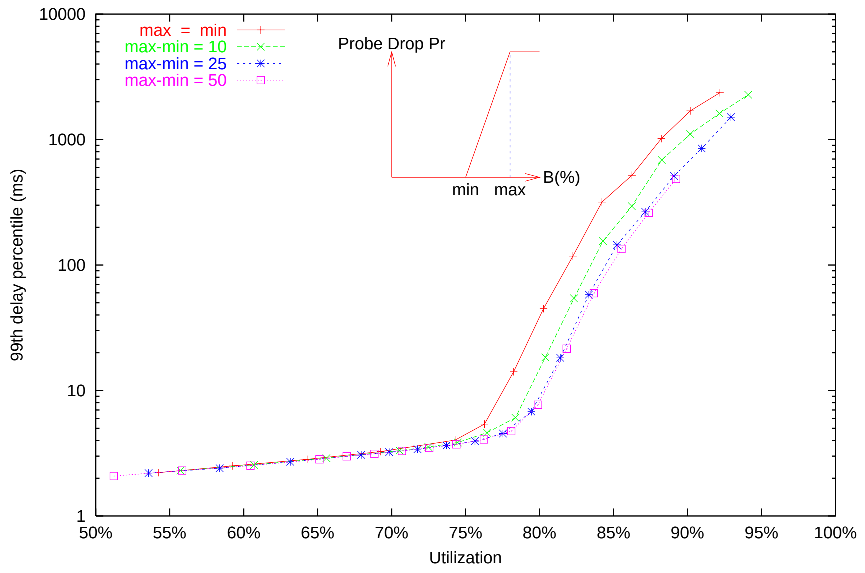


figure 39 - Offered load: 650%; source peak-rate: ~32 Kbps

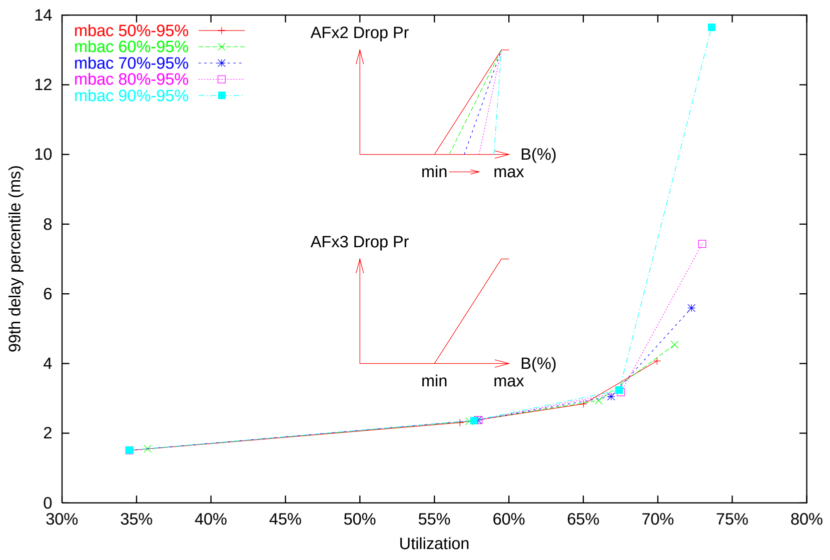


figure 40 - Offered load: 30% to 130%; source peak rate: 20 to 80 Kbps; sources with peak rate greater than 40 Kbps use AFx3 probes

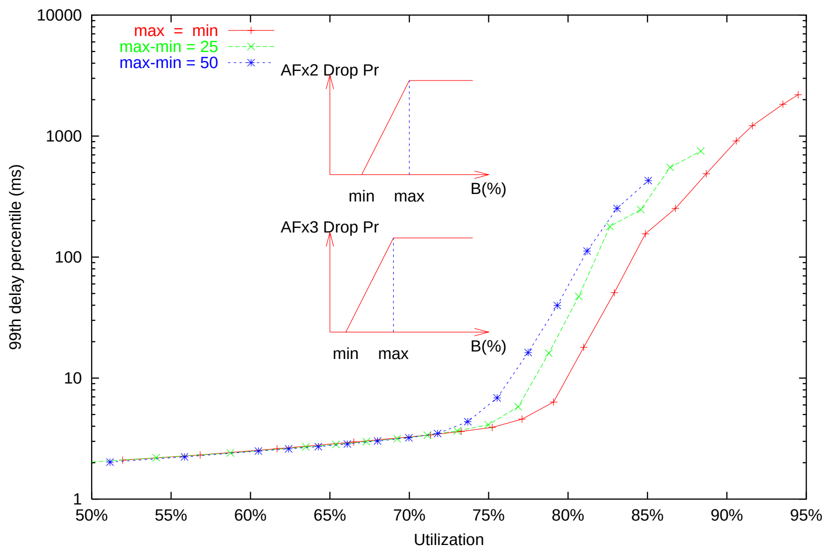


figure 41 - Offered load: 650%; source peak rate: 20 to 80 Kbps; sources with peak rate greater than 40 Kbps use AFx3 probes

Finally, the beneficial effects coming from the use of a double level of probing, AFx2 and AFx3, is highlighted in figure 42. In this figure we consider a very high offered load and sources emitting at a peak rate in the range from 20 to 80 *Kbps*. The leftmost curve in figure 42 considers a scenario in which AFx2 packets are used for every probing attempt. Conversely, the rightmost curve has been obtained by using AFx3 probe packets for flows with peak rate greater than 40 *Kbps*. Results show the clear superiority of this second approach.

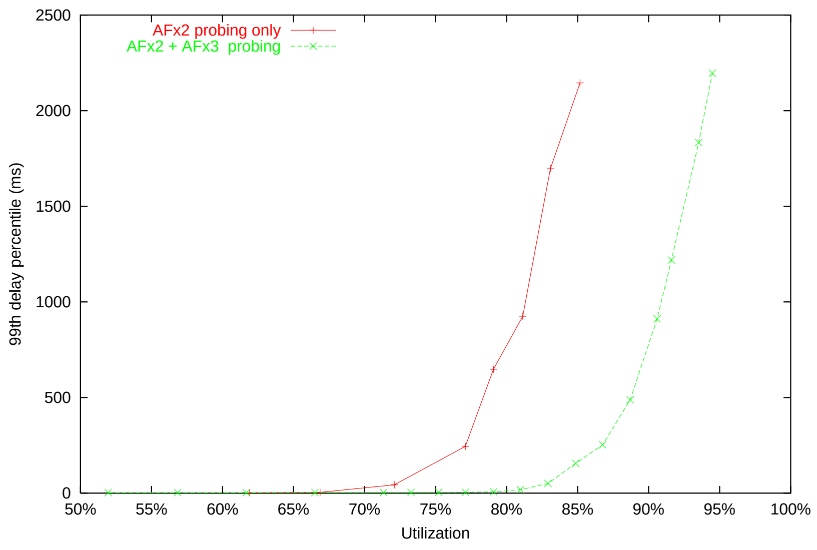


figure 42 - Comparison between AFx2 probing and AFx3+AFx3 probing with spread peak-rate distribution (20-80 Kbps) and very high offered load (650%)

§3.4 API for network traffic regulation

Distributed per-flow admission control is a promising solution for DiffServ networks. Its deployment in DiffServ domains requires the ability to suitably configure, in each network router, low-level packet forwarding mechanisms, such as packet dropping algorithms driven by traffic measurements. Hence, we propose a modular Application Program Interface (API) that allows to flexibly and adaptively configure the forwarding/dropping behavior associated to a router output queue, well beyond the traditional RED active queue management schemes.

This section presents a DiffServ router architecture capable of supporting advanced traffic control functions. We show that our API allows the support for a distributed admission control function which relies on a measurement based admission control mechanism running at the router site. Numerical results, obtained via simulation, show the effectiveness of the router configurations in achieving enhanced quality of service in various operational conditions.

§3.4.1 Characteristics of the API

Despite the generality of the AF-PHB specification, it is frequently assumed that the AF-PHB will be implemented at each core router as a RED mechanism. As a consequence, most current implementations [26], [32] are based on a monolithic approach. The queue is seen as a single entity, where measurement mechanisms (filtering and smoothing of the queue occupation) are tightly integrated with packet dropping functions, and both are indivisible entities from the pure buffering capabilities of the queue. The only way to marginally operate on such an implementation is via the specific RED configuration thresholds and parameters.

However, the apparently consolidated scenario described above does not account for novel ways to use low-level mechanisms. Hence, it emerges the need for more flexible and modular node-level APIs, that allow the deployment of non traditional low-level forwarding/dropping mechanisms, specifically devised for admission control purposes. Clearly, the same APIs would find application in the deployment of improved Active Queue Management schemes.

Such a middleware level should be designed with the following concepts in mind[16] [83]:

- *Flexibility*: the router must be plug-driven, i.e., if a module configuration (a dropper, a meter, etc.) must be changed, or a new module has to be loaded, this operation must be performed on-the-fly, with no impairments on the run-time router operation.
- *Programmability*: it must be possible to add new variations for supported components (droppers, schedulers, counters, etc), as well as program the interaction among various components; for example, an adaptive active queueing management mechanism may require a dropper to be adaptively driven by statistics collected from one or more queues, and suitably filtered according to ISP-specific algorithms (which can differ depending on the QoS provisioning level). The identification of elementary components, as well as rules for their interfacing, gives us the ability to dynamically reconfigure the router and rapidly adapt it to completely different operational conditions.
- *Openness*: there is the need for standardized application program interfaces (APIs) that allow multi-vendor independent development and rapid deployment of value-added functionalities, which, when installed, can allow to achieve new behaviors from the same device. The openness of the router will permit independent developers to focus on the system architecture rather than on the specific device.

[16] These ideas are the foundation of the European research project POLLENS, whose specific goal is to "propose a flexible and dynamic middleware platform enabling to explicitly program novel traffic control solutions and scheduling mechanisms, to configure network architectures on the fly and add intelligence to the IP routers in order to support future value-added services still to be developed today".

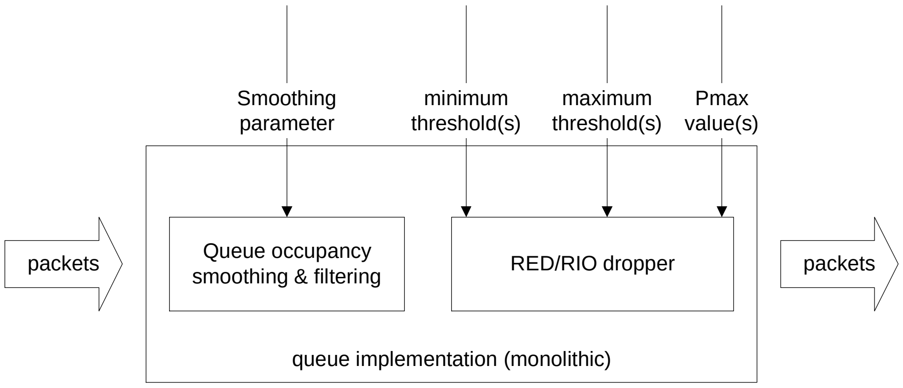


figure 43 - Monolithic queue implementation: the queue management mechanisms are indivisible from the queue kernel module

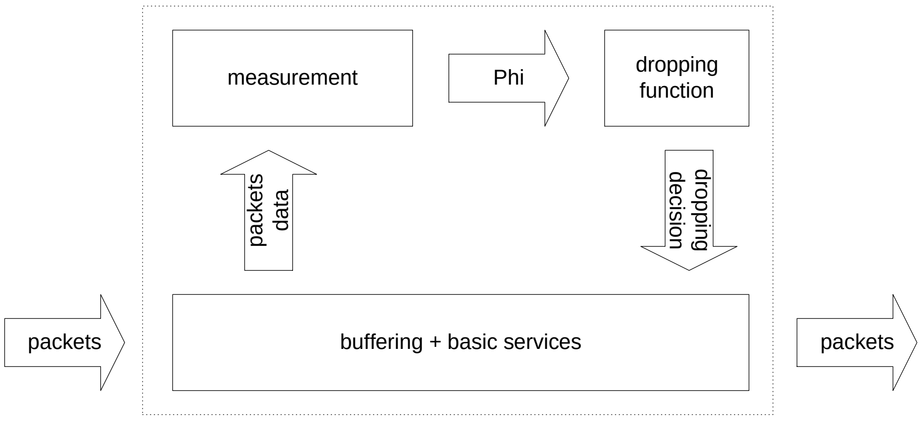


figure 44 – Modular queue implementation

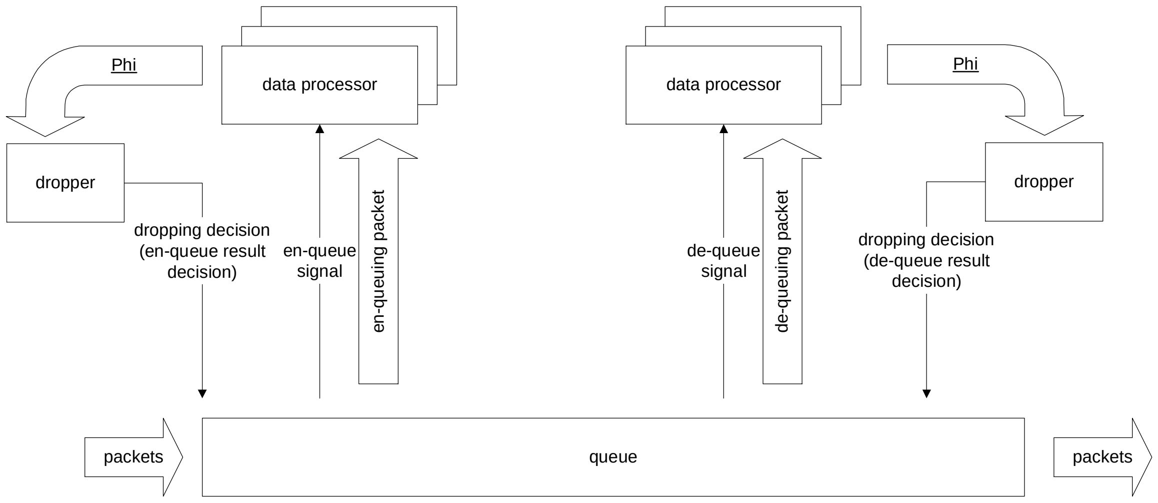


figure 45 - Architecture implementation. Three basic elements and their interconnection are shown: droppers, data processors, queues

§3.4.2 Packet dropping mechanisms for distributed AC support

We demonstrated through performance simulations that the RED-like approach to active queue management has to be changed in order to support MBAC mechanisms, without violating the DiffServ philosophy. Now we describe a possible implementation of the solution proposed in a theoretical fashion at the beginning of this chapter, in which the packet dropping probability behaves like in figure 36. This approach particularly applies to AFx2 and AFx3 packets. In fact, what makes GRIP compatible with DiffServ is the use of probe dropping/forwarding (i.e., implicit signaling, rather than explicit signaling), to convey a reject/accept information at the sender node. In order to change the behavior of the packet buffering, some changes are required to the traditional scheme proposed in figure 43, where a monolithic approach is depicted. The aim of figure 44 is to show a new modular queue implementation scheme, where buffering functions, measuring and dropping modules are independently instantiated and configured. This allows to easily change the behavior of each single module, or even to replace an entire module without turning off the overall system.

§3.4.3 The packet-level traffic control API

The discussion carried out in section §3.4.2 has shown that packet dropping mechanisms implemented at core routers may have a complete different goal versus the traditional ones. In the context of admission control support, new low-level packet forwarding/dropping mechanisms should be implemented at the network routers. Hence, there is the need for node-level programming interfaces that allow to build these mechanisms on top of the hardware router implementation. This is in conflict with the traditional queue kernel module implementation, illustrated in figure 43. To overcome this problem, we have developed a modular architecture, devised to allow the implementation of different packet-level dropping/forwarding mechanisms. Our architecture is composed of the following basic building blocks (see figure 45): queue, data-processor, and dropper. Each module exposes an API that provides supplementary services, as explained in the following.

Other than buffering packets, the queue provides additional services the other modules can access to: *en-queue entry point*: it provides to the registered modules an asynchronous signal that notifies the arrival of a packet. Besides, supplementary information is carried to the data processors, such as packet size, specific header fields, etc; *de-queue entry point*: same as above, but for a packet departure. The idea is that the whole system works as an event driven system, that reacts to the events of packet en-queue/de-queue. Concerning implementation, the de-queue event will be generated by an external entity when the packet must leave the queue, e.g., when the packet must be sent to the Network Interface Card. Specifically, such event is generated by the packet scheduler that decides which, among the available queues, must be selected to send the next packet. Symmetrically the en-queue event is generated by another entity that decides when the packet is ready to be en-queued. It must be pointed out that we are not interested in what these entities are and how they generate the signals; we only know that the queue must handle them.

The dropper is a module that provides a decision, generally consisting in dropping/forwarding a packet. The architecture of the dropper is illustrated in figure 46. A dropper is bound to the en-queue (or de-queue) hook of the queue. When an en-queue (de-queue) signal is generated, this signal is passed to the dropper, which responds with an *action list*. This can be as simple as a packet dropping/forwarding decision, or a complex list of actions (e.g. "forward the current packet, and drop the one at the head of the queue").
An action list is selected based on:

- *the default available action lists* - every dropper has a built in default action list. The minimal set of action lists is drop-packet and forward-packet;
- *the user selected action list* - At module instantiation, or at run time, the user can select a different set of actions for the dropping and/or forwarding decision;
- *state information* - based on measurements taken on the system, and provided by a data processor bound to the dropper; since the state information is, in general, multidimensional, such an information will be passed via a vector Phi, where Phi=phi1, phi2, phi3,., phin, with n arbitrary;
- *reference parameters* - such as thresholds, probabilities, etc.

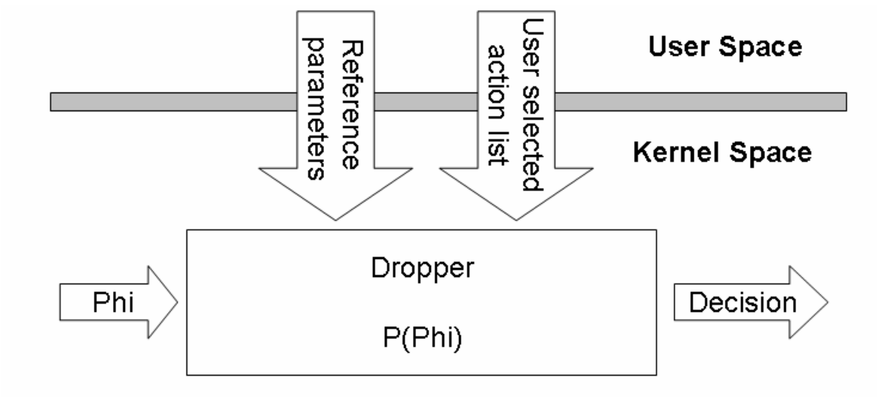


figure 46 - Dropper architecture

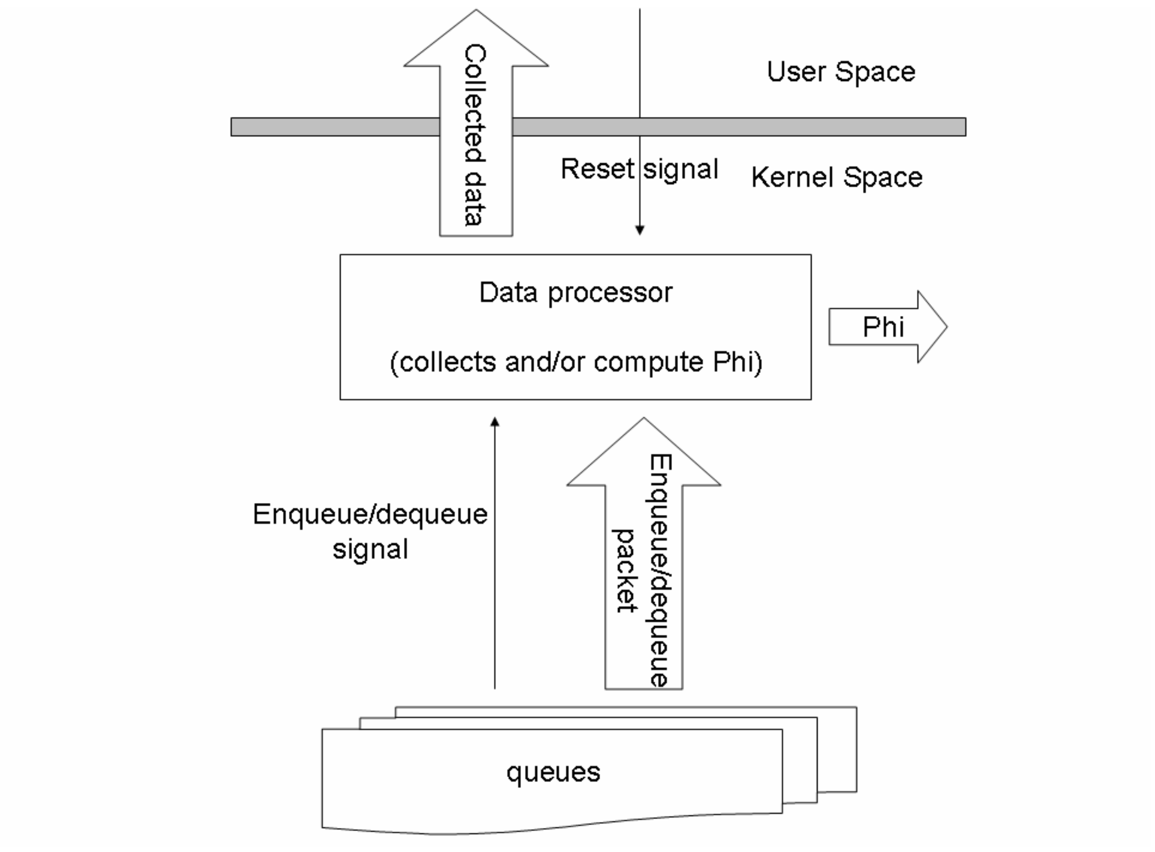


figure 47 - Data Processor architecture

We have called data processor the module that manages measurements and system states. A data processor uses the en-queue and/or de-queue signals to collect a number of statistics: offered load (packets/second, bytes/second), carried load, queue occupation. A supplementary task of the data processor is to filter the collected running statistics, where the filter parameters can be eventually modified during run-time operation. As shown in figure 45, note that more than one data processors can be instantiated on the same queue (for example, to provide measures at different time scales). The architecture of the data processor is shown in figure 47. While the role of the dropper is completely realized into the kernel space, i.e., it is a pure element that have been moved out by the modularization process of the queue, we propose a more interesting role for the data processor. In fact, an important role of the data processor is the capability of exposing the collected and processed data up to the user space. In particular, it may be essential to expose suitable traffic measurements and filtered queue congestion state to external applications (middleware applications, management applications, etc). This can be done easily by binding a data processor to the en-queue/de-queue event, without connecting it to any dropper. In this case we distinguish these classes of data processors as the *passive* ones (since they do not take part into the decision process).

Another important relationship we want to point out, is how the data processors and the queue can be connected. In most cases it is natural to associate one data processor to one queue, but it can be interesting to connect one data processor to two or three queues, and correlate events coming from them, as well as relate measures.

The idea behind the developed modules is to allow the implementation of new traffic management mechanisms by simply binding the modules together[17]. Modularity enables us to reuse already developed droppers or data processor in different scenarios. Moreover, by differently connecting the same modules, it is possible to change or extend the router functionalities. To build a valid system, the user firstly instantiates a mandatory queue module, then instantiates additional data processor and the dropper objects. Finally the modules may be interconnected according to the following rules:

1. The abstract queue provides two entry points:
    a. The en-queue entry point will be activated in response to the en-queue packet request of the operating system (or any form of underlying layer)
    b. The de-queue entry point will be activated in response to the de-queue packet request of the operating system (or any form of underlying layer)
2. Zero or more data processors can be bound to the en-queue/de-queue event hook
3. Zero or one droppers can be bound to the en-queue/de-queue event hook
4. The dropping decision can be taken only fetching the dropper decision accumulator, in case one dropper is bound to the en-queue (de-queue) event. The event sequence is:
    a. The en-queue (de-queue) packet request has been received by the queue
    b. The packet is notified to the en-queue (de-queue) registered data processors
    c. The data processors elaborate the *Phi* vector, and they send it to the registered droppers (for each data processor)
    d. The dropping decision is fetched from the en-queue (de-queue) registered dropper. The dropper is responsible to respond to the *fetch decision* request generated by the queue.
    e. One or more actions are taken by the queue, as to the dropping decision
5. A data processor can be bound to one or more queues
6. A data processor can be bound to zero (or more) droppers
7. One dropper can be bound to one (and only one) queue

[17] Clearly, the introduced flexibility has a price in terms of code optimization. A goal of the ongoing research activity is to understand the scalability limits of the proposed software platform.

8. En-queue data processors and de-queue droppers can be mixed (and vice versa). So for example an en-queue data processor can be bound to a de-queue dropper, but the user must always consider that the dropping decision will be fetched when a de-queue event will occur. Anyway, mixing en-queue and de-queue modules is not recommended.

§3.4.3.1 Examples of traffic control schemes implementations

Traffic management mechanisms can be implemented by suitably binding the described modules together[18]. A mechanism is implemented by first instantiating a mandatory queue module, and then binding eventual data processors and droppers. The dropping decision can be taken only fetching the dropper decision accumulator, in case one dropper is bound to the en-queue (de-queue) event. In particular the event sequence is: i) the en-queue (de-queue) packet request has been received by the queue; ii) the packet is notified to the en-queue (de-queue) registered data processors; iii) the data processors elaborate the state vector *Phi*, and send it to the registered droppers (for each data processor); iv) the dropping decision is taken and enforced on the corresponding packet (i.e., the dropper is responsible to respond to the "fetch decision" request generated by the queue, which then takes the specific dropping/forwarding action).

Here we show how to implement the GRIP router described in section §3.4.2 with one data processor and two simply droppers (two instantiations of the same dropper template). We recall that the router measures AFx1 traffic, and drops AFx2 and AFx3 packets according to a dropping profile illustrated in figure 36. The figure 48 shows the proposed GRIP architecture. The data processor has the task of measuring the incoming traffic, before it is actually offered to the queue (part of this traffic might be lost during congestion). The dropping actions are taken by the en-queue/de-queue droppers according to the *Phi* state information provided them. During congestion, they drop both AFx2 and AFx3 packets entering the queue, as well as AFx2 and AFx3 packets exiting the queue. The role of the *Phi* vector, in this specific case, is to select the right dropping function via the color parameter, and refresh the decision stored in the decision accumulator. As shown in figure 49, the traffic data collected and filtered by the data processor are sent to the dropping functions via the vector parameters, phi_2 and phi_3.

[18] While modularity allows reusing already developed modules, as well as changing the forwarding behavior by simply composing modules in a different manner, the introduced flexibility has a price in terms of code optimization. A goal of the ongoing research activity is to understand the scalability limits of the proposed software platform.

We set $phi_3=phi_2$ due to the simple fact that we want to discard the probe packets simply setting a more aggressive threshold in the AFx3 case. Finally, note that AFx1 packets should never be dropped, and thus the dropper is fed with a constant state parameter fictitiously meaning that no congestion is encountered (e.g., $phi_1=0$).

As a second example, to show the possibility of module reuse, we provide an implementation of RIO. In figure 50 it is shown a solution for the dropping mechanism presented in [29]. The IN meter must collect data regarding the incoming *in-packets* and compute the avg_{in}, while the OUT one computes the avg_{total}, which takes in account both the *in-packets* and *out-packets*. The droppers are exactly the same ones presented in the GRIP implementation, but this time the dropping piecewise linear dropping function has been set as RED-like one. This can be done during module instantiation or even at run time.

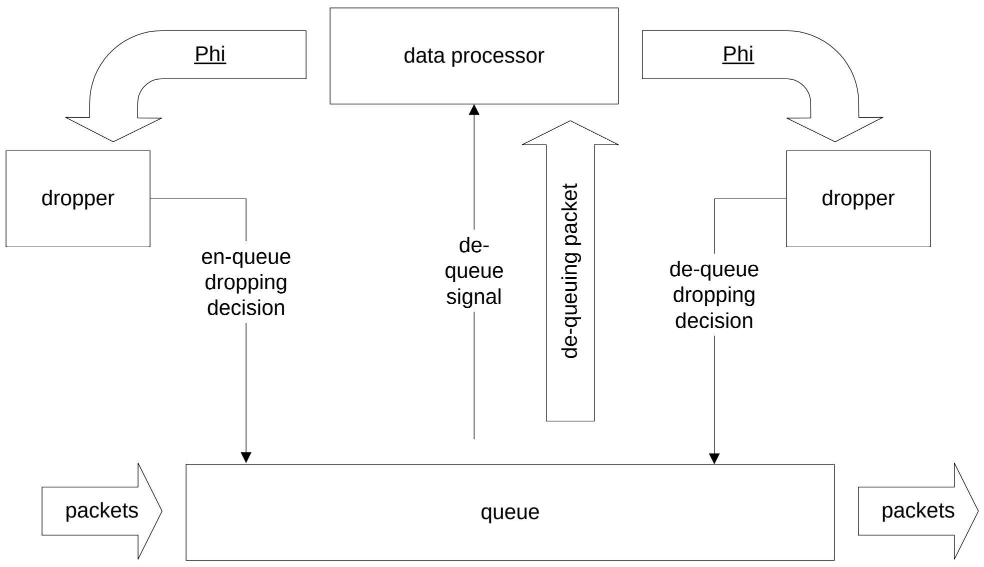


figure 48 – A simple GRIP implementation

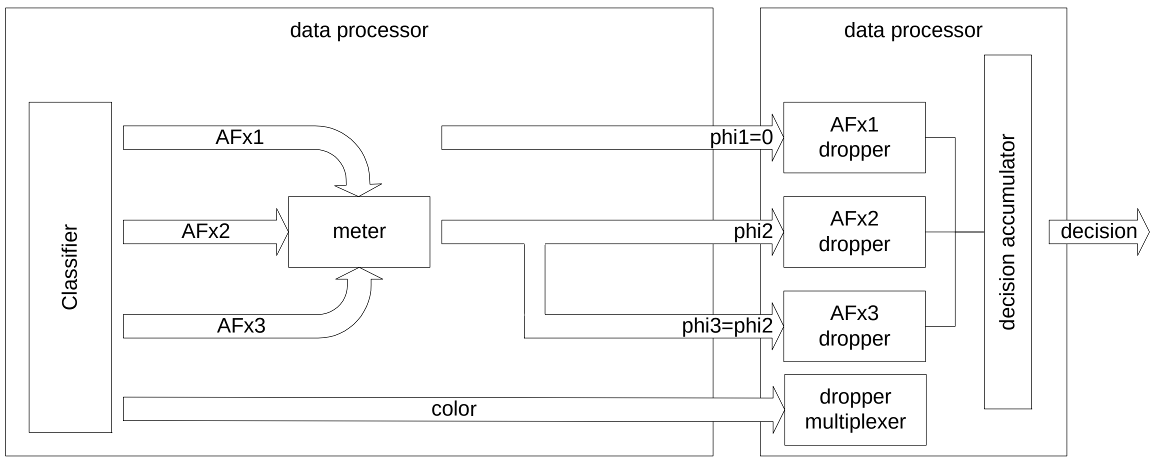


figure 49 - A closer look at the data processor and dropper implementations

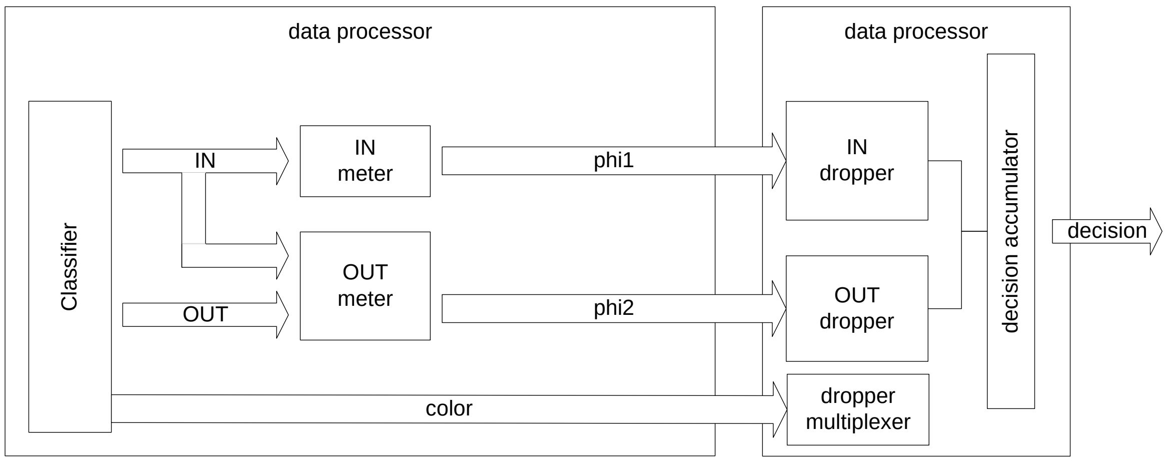


figure 50 - A proposed RIO implementation

§3.5 Performance of GRIP over a real test bed

A set of Application Program Interfaces (API), has been actually implemented over a Free-BSD system. As discussed in section §3.4, the particular API is meant for advanced traffic regulation issues, even though it is a widely general set of tools that allow the regulation of very simple, in principle, low level components. In fact, API proposed in section §3.4 basically affects the behavior of network interfaces while queuing and marking packets. Additionally, the use of API permits a dynamic control of service classes, including the possibility of adjusting service characteristics while running. API allows the deployment of GRIP in a light and quite natural shape, while leaving space to different implementations. This means that GRIP and many other mechanisms can be implemented using the proposed API, and they could be easily modified, reconfigured or upgraded in some other new operational behavior. Together with the definition and design of our API, a great effort has been dedicated to the validation of the API operation, and GRIP was the natural test application to be used in a real test bed. Thus we report here some performance obtained in real network where GRIP, API, MBAC-endowed core routers and many other DiffServ compliant mechanisms (especially as regards the policy functions to be enforced at the edge of the DiffServ domain), run under real traffic condition, and sometime in very uncommon overloading conditions. A part of the simulation scenarios presented through Chapter II and Chapter III, was reproduced in such a real network, and some performance figures are here reported to summarize the results we have obtained.

§3.5.1 The real test bed and relative GRIP performance

In this section we briefly report and discuss some tests that were really performed by implementing DiffServ AF PHB and GRIP with measurement based admission control support over a prototypal open routing platform[19]. Firstly a presentation of the test bed, used for experimentation and performance measurements, is given in what follows:

- A *traffic generator* acts like a GRIP source, i.e., it generates GRIP compliant connections. Each connection is unidirectional and it is modeled as a Brady's *On/Off* flow with exponential distribution for both *On* and *Off* periods. Moreover the packets are distinguished by means of a label, so that it is possible to identify each connection easily. This node runs over a PC with Intel Pentium II processor at 800 MHz, and 512 MByte RAM.
- A traffic sink, *the GRIP destination*, where GRIP connections are addressed. This is used only for packet reception, but for the GRIP feedback to be sent towards the traffic generator. Since this node has not particular hardware requirements, a PC with Pentium II – 266 MHz, and 64 MByte RAM is employed.
- The *sniffer* have the task to record and store incoming and outgoing traffic, avoiding synchronization problems (e.g., Network Time Protocol). We have used customized tools[20], so that we can calculate for each packet various statistical properties and aggregate them in terms of 95th and 99th percentiles. The hardware adopted for the sniffer is a PC with 1,7 GHz Intel Pentium IV processor and 512 MByte RAM.
- Two *hubs* are employed in order to let traffic and signaling packets to reach the sniffer without affecting the normal network operation. In fact, hubs simply forward each packet received from a port over any other port in the hub.
- A *PDP*, i.e., a policy decision point, is used to remotely tune the GRIP router behavior. This node does not perform any CPU-consuming operation, so that we used a PC with a Intel Pentium II at 266 MHz and 64 MByte RAM.
- A *GRIP router* endowed with GRIP and measurement based packet dropping algorithms tunable through API. This router runs over a PC with 1,7 GHz Intel Pentium IV processor and 512 MByte RAM (the same as for the sniffer).

[19] Some performance evaluation of the real GRIP behavior has been carried out in the frame of the EU founded ITEA POLLENS project, in collaboration between University of Palermo and CRES.
[20] The adopted virtual statistic wire for statistics collection and elaboration has been developed in the frame of ITEA POLLENS project, in CRES laboratories.

- Two ethernet segments with different bandwidth capacity, i.e., 100 *Mbps* in the link between the traffic generator and the GRIP router, and 10 *Mbps* between GRIP router and traffic destination. The adoption of different ethernet segments is meant to allow the network overload before the GRIP router, while the admission control is in charge of reject connection requests when the link between the GRIP router and the GRIP destination reaches an utilization target.

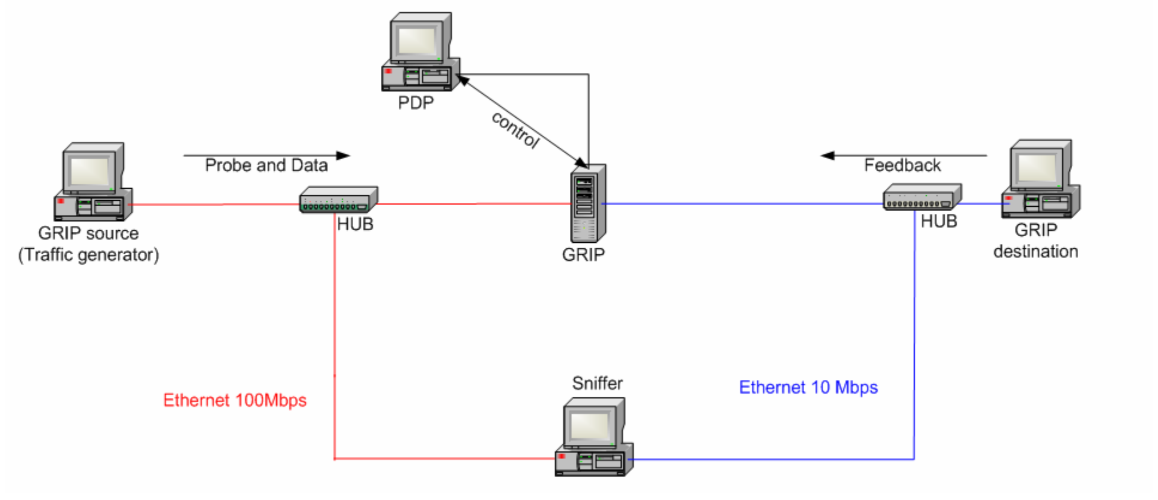


figure 51 – Test bed architecture

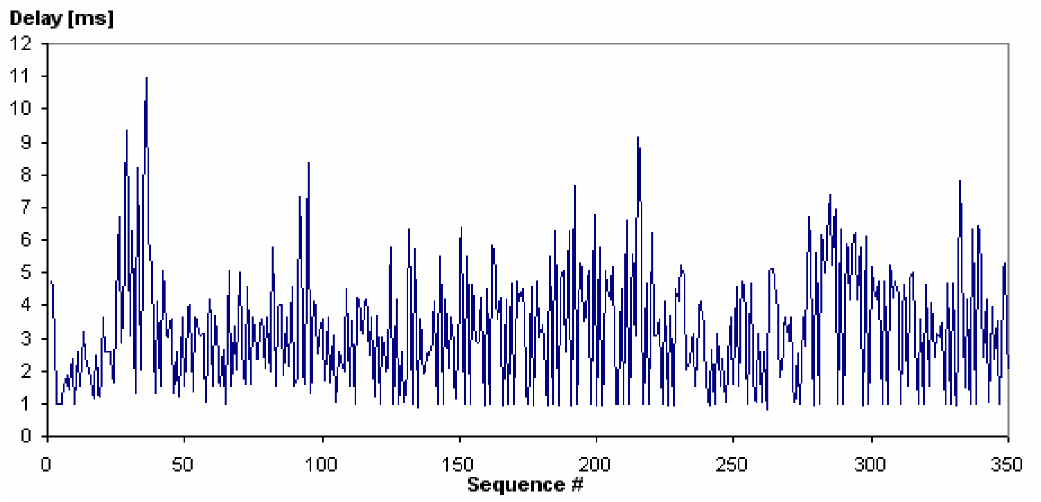


figure 52 – Packet delay vs. sequence number (one connection)

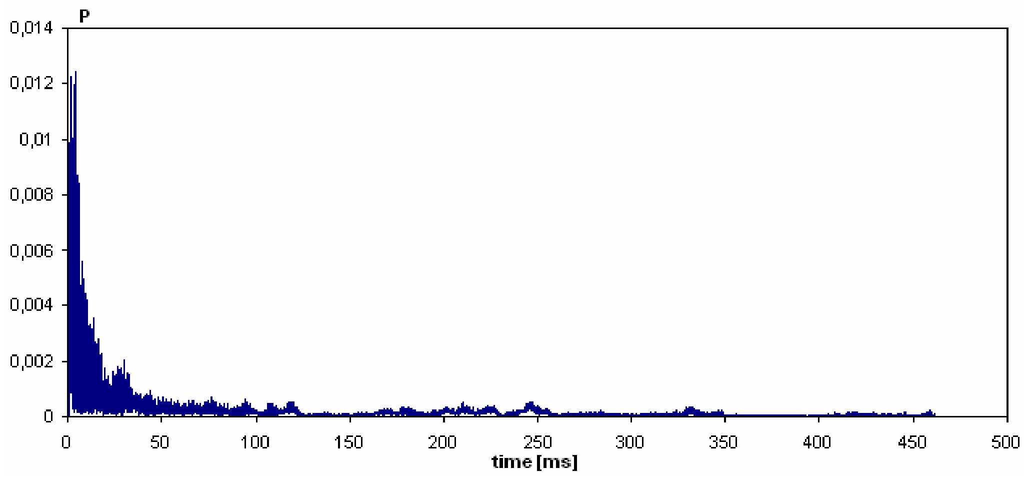


figure 53 – Delay distribution function

The first performance figure reported here is about the packet delay versus the experiment time, expressed as the sequence number associated to the packet for a particular connection. The network load, offered by the traffic generator is about 110% of the bottleneck capacity, i.e., the link to the GRIP destination, which would be not able to sustain that traffic neither if it were a single constant bit rare (CBR) flow. The dropping algorithm is tuned with the following parameters: $P_{max} = 1$, $th_{min} = 25\%$, $th_{max} = 75\%$. As shown in figure 52, the delay experienced by packets varies over time, as the network load varies too, but it is strictly bound to few milliseconds. In any case, this behavior implies that more effective performance figures have to consider statistics as the mean values, the distribution function and some relevant distribution percentiles. Thus a statistic description of the performance shown in figure 52 over time, is given in figure 53, where the empiric delay distribution function is represented for the same test. From the figure it is clear that most of the values are within the range from 0 to 10 *ms*, while the probability that delay outcomes that threshold cumulates a very small area. Note also that, even though the figures evidence the system is working as expected in term of qualitative behavior, some packets can very seldom experience a bad service.

We also want to show the experimental numerical results obtained by varying minimum and maximum thresholds in the dropping scheme, while the offered load remains fixed to the 110% of the receiver link capacity. Results reported in table 7, show that network performance can be suitably tuned by means of the fine tuning of the dropping thresholds in the measurement based GRIP admit/reject decision. It is worth noting that the proposed operation makes the network provider able to limit the delay as well as the throughput, no matter the load offered by external nodes. Obviously, delay and throughput are not independent, meaning that very low delays can be maintained only by limiting the throughput to not overcoming 50% of network capacity. However, the experimental results reported here for significantly limited admitted traffic are in conformance with simulative results, and this leads to expect an analogous behavior also in case of high accepted traffic, i.e., in case of utilization targets of the order of 80-90% of the bottleneck.

th_{min} - th_{max}	*99th delay percentile*	*Throughput*
0-1	6.50±0,08	0.404±0,038
0,1-0,9	7,30±0,07	0,442±0,005
0,5 - 0,5	6,15±0,06	0,499±0,001
0,25 –0,75	6,70±0,03	0.420±0,002
0,40 – 0,60	7,80±0,06	0,481±0,038

table 7 – End-to-end performance of GRIP controlled application with 110% offered load

§3.6 Conclusions

This chapter introduced to GRIP functionalities and application fields, even though other scenarios can suit with GRIP operation and some other applications will be considered in the following of this thesis. In particular, in this chapter we have shown that the problem of provisioning a DiffServ domain can be transformed into the problem of configuring packet discarding algorithms in core routers, in conjunction with the support of a distributed admission control function based on probing (i.e., GRIP). We have also shown that the standard DiffServ AF PHB is already semantically capable of supporting the GRIP as a stateless and scalable admission control mechanism. The driving idea is that accept/reject decisions are taken at the edge of the network on the basis of probing packet losses, being probes tagged with a different AF level than packets generated by accepted traffic. In turn, these losses are driven by the dropping algorithm adopted in the specific AF implementation running at each network router. It is important to understand that, following the spirit of DiffServ, the above described operation does not aim at providing a quantified level of assured QoS, but, in conformance with PHB and PDB specifications, it provides a reference framework over which quantitative performance specification may be deployed. The key to QoS guarantees is left to each specific implementation, i.e., it is left to the implementation-dependent quantification of the notion of congestion, which triggers the gate mechanism for AFx2 packet discarding. Each administrative entity is in charge of arbitrarily determine the optimal throughput/delay operational point it wants to support.

The AF PHB implementations so far considered in the literature, based on (RED) thresholds set on the queue occupancy, do not affect the described endpoint operation and thus may be considered to support admission controlled traffic. Hence, we proposed an evaluation of performance relative to such an admission control function for various router configurations. An important result is the understanding that, when probes originating from different flows are mapped on different drop levels, significant performance enhancements can be achieved. Moreover, an additionally important contribute of our work was to prove that RED implementations are definitely not capable of achieving tight QoS support, regardless of their parameter settings. In fact, the measurement mechanism adopted is overly simple, and this translates into a poor QoS support, but still much better than best effort, since admission control is still enforced on setting up connections. Conversely, significantly better results was obtained by replacing the legacy queue occupancy measurement module with a run-time traffic estimator, even when this module simply operates a traffic sampling and computes an auto regressive filtering over sampled data.

As shown in the second part of the chapter (sections §3.3 and §3.4) and especially in performance figures, the application of packet dropping mechanisms to the new context of admission control requires the deployment of new packet-level traffic control functions, very different from the traditional approaches based on RED/RIO queues. To this purpose, we have proposed and tested a packet-level traffic control API, based on three components: queue, droppers, and data processors. Each component is implemented in a modular and flexible fashion, and it is open to be configured and tuned while running. Further research issues include the understanding whether the overhead, given by the introduction of the modularization process, will lead to scalability problems in the routers.

Chapter IV

DiffServ/GRIP QoS-aware extensions for signaling interoperability and multicast

This chapter complements the discussion carried out in Chapter III, where the basics of GRIP and its implementation was proposed. From the discussion above, GRIP emerges as a powerful tool that provides a kind of connection blocking service, whose performance are strongly dependent on the actual implementation of some low level services, like packet marking, dropping and forwarding mechanisms. Hence, we demonstrated that GRIP works very well in a stateless network domain, and the only required feature the GRIP relies on is the possibility to differentiate the packet dropping algorithms that operate on *probe* packets and *data (information)* packets respectively. In turn, GRIP requires also to suitably label packets in at least two groups: probes and data. The nature of the GRIP service seems to be tailored on unicast and unidirectional applications, with a single sender and a quasi-passive receiver. This resembles to be somehow true, at least as regards its origins. However, GRIP has been also proposed for bidirectional unicast applications, with an approach that recalls to the full duplex communication operation obtained through two paired half duplex transmissions: a sender opens a GRIP session and the receiver responds with an additional GRIP section over the inverse path, so that the endpoints start transmitting data only upon the completion of both GRIP sessions. Unluckily, this first extension does not account for two fundamental issues: i) the interaction with some other reservation protocol and admission control scheme that could run in some network segment between endpoints; ii) the support to QoS for multicast applications. As to point i), it makes sense to exploit all available network capabilities instead of limiting the scope of the GRIP action to endpoints and MBAC modules located in a part of network routers. Moreover, if a connection path traverses multiple stateful (non-GRIP) and stateless (GRIP-enabled) domains, the GRIP solution cannot provide any affordable connection blocking services, since it is not able to recover any knowledge of the congestion state of routers located in the stateful domains. Hence, it looks better to limit the GRIP scope to GRIP-enabled domains, while inter-domain signaling extensions are required to make different QoS protocols (e.g., GRIP and RSVP) to interoperate. As to point ii), all considerations previously carried out apply, but for the additional need to extend the connection blocking service to multicast groups when requesting to be joined to a given multicast router with QoS requirements.

The following of the chapter firstly introduces to signaling interoperability issues (section §4.1), with particular attention to the GRIP-RSVP interworking and a general signaling interoperability approach proposed by the IETF NSIS Work Group. Secondly, multicast extensions of GRIP operation are proposed in section §4.2. Eventually, section §4.3 briefly summarizes and concludes the chapter.

§4.1 Signaling interoperability

The Internet scenario is characterized by heterogeneous networks and deeply different administrative domains. Given a specific data path, it is very frequent that a significant number of nodes belonging to that path are not QoS aware. Furthermore, while access network segments are often QoS-aware and stateful, it is very common to go across stateless network domains in the core part of the network. Additionally, there are multiple possible approach to the QoS in the IP network. In Chapter I we introduced two possible frameworks, IntServ and DiffServ in order to provide service differentiation and somewhat defined levels of QoS. In comparison with the Best Effort IP networking, that can be considered suitable only for traditional Internet applications, IntServ introduces two services that result adequate for different levels of multimedia applications such as IP telephony, audio or video streaming which are very sensitive to delay caused by network congestion. IntServ relies on a reservation protocol like RSVP or similar, and needs for stateful routers to be deployed in the network. This turns in a sometime unacceptable network overload and scarce scalability. On the contrary, the approach proposed by DiffServ is much more soft and sometime evolutionary then the IntServ one. It is definitely scalable and permits a strong customization in the services, including the relevant possibility of endowing the network with additional QoS aware tools. The main drawback of DiffServ is that no service guarantees are provided, but the protection of some classes with respect to other classes, until the network traffic does not overwhelm the network capacity.

However, other different kind of solutions exist for IP networks, so that it is possible that a flow has to traverse multiple heterogeneous domains, with totally different QoS approaches and QoS tools, which are very difficult to coordinate. In general, it does not appear possible to allow a flow reservation to be performed over such a multiplicity of adjacent networks. Nonetheless, some protocols operate separately in each domain with a local semantic but with similar purpose [22]. In this framework, it is emerging a new idea proposed by the IETF NSIS Work Group. The proposal focuses on the possibility to provide some means to allow the interoperability of different QoS signaling protocols across heterogeneous networks. NSIS is a group that deals with Next Step In Signaling in the framework of IP networks, and its members are now debating on the way to obtain a scalable and light extension for signaling application features over a generic heterogeneous networks interconnection. Te basic idea is that a general purposes protocol should adapt to each specific network domain property, thus acting as an envelop able to transport any kind of domain specific signaling protocols, with the additional capability of translating protocols and signaling message format while hopping from a domain to another.

NSIS capabilities are developed in three types of entity (NSIS Entities - NE), which are the functions within a node, which implements an NSIS protocol. Note that NEs can be located in the data path or elsewhere in the network, but for the case of path-coupled signaling, because of NEs have to stay on the data path. The NSIS entities are:

- NSIS Initiator (NI): NSIS entity envisioned to initiate the NSIS signaling in order to setup or manipulate network state.
- NSIS Responder (NR): NSIS entity where NSIS signaling is terminated and that can optionally interact with applications as well.
- NSIS Forwarder (NF): NSIS entity between a NI and NR, which may interact with local state management functions in the network. It also propagates NSIS signaling further through the network.

§4.1.1 The use of GRIP for signaling interoperation purposes

The driving motivation of our work, is that some network domains, e.g. based on DiffServ data-plane, might not explicit support a per-router and/or per-domain admission control rule [10]. Hence, for such domains, explicit signaling is not a viable approach. To partially solve this issue, we suggest to adopt a GRIP paradigm devised to provide implicit signaling via data-plane packet delivery operation. In this case, the implicit signaling should decide to locally admit a new flow to cross the DiffServ domain upon the successful and timely delivery, through the domain, of probe packets independently generated by an entity located at edge router, e.g., by a NSIS initiator (NI) entity. The key idea is to use failed receptions of probes to discover that a congestion condition occurs in the network segment between a (NSIS) initiator and a (NSIS) responder, and to abort a reservation procedure. Since GRIP is not able to communicate per-flow traffic and QoS parameters, in principle it cannot exert a QoS control as tight as in the case of explicit mechanisms. However, it is important to notice that GRIP can indeed operate in a differentiated manner on the basis of traffic and QoS parameters, by tuning parameters like:

- Marking of probes, to be performed in compliance with the flow traffic and QoS requirements;
- The dropping behavior enforced for marked probes;
- The estimate of traffic in the core routers, aimed at controlling the probe dropping.

It is relevant to highlight the differences between this work and the QoS framework depicted in [23]. The goal of [23] is to deploy end-to-end, quantitative QoS by applying the IntServ service models (CLS and GS) end-to-end across a network containing one or more DiffServ regions. To accomplish this task, [23] discusses the interoperability of the IntServ signaling protocol (RSVP) with custom signaling implemented in a non RSVP-aware DiffServ domain. In other words, signaling interoperability is only a secondary issue in the context of [23], while the primary focus is to provide an end-to-end IntServ support model across DiffServ domains. Conversely, we are interested in developing tools that allow the Internet to be equipped with a pure "connection blocking service", regardless of the QoS provisioning level within each administrative domain. These tools are used by QoS-aware applications running at endpoint and/or border router side, and represent a solution for end-to-end IntServ support as well as for any other kind of reservation framework.

A number of mechanisms can be used in order to enable a service provider to offer support for QoS in a stateless network domain, even though, by definition, no states can be handled nor maintained in such a kind of networks. As a consequence, signaling interoperation entities (e.g., NEs) can only be placed at the edge of the stateless domain, and no resource reservations can be performed over internal nodes. This means that, in a stateless domain, core routers are not able to set apart resources for a specific flow, nor manage it separately from the others. Conversely, core routers can be endowed with GRIP features, with the aim of providing a connection admission control in a sort of virtual link between two edges of a stateless domain.

From a NSIS perspective, this means that no NFs can be placed in the core of a stateless domain, but NI and NR can be embedded in edge nodes. Following the approach proposed in [49], the NSIS protocol suite is split into a signaling transport layer and a signaling application layer. The former (NTLP) is a generic transport protocol, and it is signaling independent, since no correlations arise with specific signaling methods. The latter (NSLP), on the contrary, contains mechanisms specific of a signaling application. A different NSPL is needed to support each specific signaling application, while a single NTLP gives support to any kind of NSLP. QoS mechanisms are specific of NSLP protocols, while NTLP provides a simple message transfer service. Thus, the complexity of the signaling is mainly located in the NSIS signaling application layer. In the GRIP case, the role of NSLP stays in the management of GRIP procedures for probing purposes and local node measurements, and in the appropriate marking of NTLP packets in the DiffServ domains (or equivalent operation for any other kind of stateless domains).

In what follows we will focus on RSVP, as a practical example, and we will show how RSVP messages are useful to provide an admission control service into the DiffServ domain using the measurement based criteria proposed with GRIP. This QoS guarantees are provided to network customers, although non–QoS-aware domains have to be crossed during packet exchange. Moreover this approach is able to merge the scalability of GRIP/MBAC scheme in core networks, plus the advantages of RSVP policy in access networks. A further goal is to show how this framework (and in particular border router interface) can be easily implemented and introduced in a real access scenario, without a relevant cost for the provider[21].

With the aim of supporting RSVP and GRIP interworking, the GRIP implementation to be deployed has to take into account more that two QoS levels. In other words, there is the necessity for separate sets of paired (data and probe) packet labels per each pre-determined QoS level, in order to independently manage the related decision criteria. Specifically, when the source node wants to setup a flow, it firstly selects the domain QoS level Qi, which best matches the flow QoS requirements. Then, it proceeds by sending a packet marked as probe for the QoS level Qi. Each router manages a separate pair of queues[22] for each QoS level, and exerts a dropping behavior on probes for the QoS level Qi on the basis of the measurements taken on the relevant aggregate traffic for QoS level Qi. For instance, the four available AF classes could be adopted for four different GRIP-enabled QoS classes. Additionally, a more interesting issue is related to the extension of the proposed approach to manage traffic flows with highly different bandwidth requirements. In fact, the described GRIP implicit signaling operation does not require a core router to be able to parse the contents of a probing packet.

[21] The key could be the use of a programmable router, whose API must satisfy some defined criteria. As landmark we refer to the active router developed in the European founded POLLENS project (ITEA if0001a).

[22] At least, two *virtual* queue in the same *physical* queue are required (see the RIO approach).

Hence, in terms of router operation, the decision whether to admit or reject a flow does not take into account the amount of resources (e.g., peak or average bandwidth) required by the incoming flow. However, we have already shown that it is possible, in the hypothesis that AF classes are selected for GRIP operation, to partially transfer the semantics of the admission control request down to the data-plane by using a differentiated set of probes for different traffic profiles. In general, the idea is to use, for a given QoS level Qi, a single DSCP for admitted traffic, and multiple DSCP marks for probes. In section §3.3.3 and in [9] we have detailed the case in which three DSCPs are reserved for the implicit signaling operation (where the value three is related to the number of drop levels available in a DiffServ AF class). Traffic whose peak (or average) bandwidth requirements is above a given value use a different DSCP label for their probes. Consistently, network routers start dropping these latter probes in advance with respect to the "normal" probes. The rationale is that flow whose bandwidth requirements are "large" need to be blocked in advance with respect to traffic whose bandwidth requirements are "small".

Clearly, the obvious drawback of the described approach is that the number of packet labels supported by a domain might be extremely small with respect to the large variety of possible traffic descriptors. However, the described approach at least provides a rough mechanism to take into account different traffic profiles, without abandoning the advantages provided by the GRIP approach.

§4.1.2 Edge-to-edge and cross-domain implicit signaling operation

The implicit signaling description given for GRIP was based on the simplified assumption of end-to-end admission control operation, where the end points of the admission control procedure were the two end hosts, and where the core routers involved in the admission control procedure were considered to belong to the same network domain. To make GRIP deployable, a necessary extension is to localize the GRIP operation into the network domains that are not expected to deploy alternative state-based admission control forms, meanwhile allowing to extend the implicit signaling concepts to make them compatible with explicit signaling approaches adopted in network domains involved in the admission control operation. A general framework which allows coexistence of implicit and explicit signaling is to decouple the admission control in the following elements:

- Intra-domain resource reservation mechanisms. These mechanisms should be limited to provide admission and congestion control functions whose scope is limited to a single administrative domain, and whose design is related to the specific requirements of the considered domain (e.g., a radio access network, a core backbone, a small campus LAN, etc). The degree of QoS support provided within each domain will depend on the tightness of control that the edge to-edge mechanism will be capable to support. Schemes ranging from explicit per-flow resource reservation mechanisms (such as RSVP), down to aggregate forms of traffic control (e.g., via measurement based mechanisms, such as the one of GRIP) should be allowed to independently operate in different domains. The ultimate goal is that each domain should be placed in the ideal condition of determining the suitable throughput/QoS support tradeoff within the domain.

- Inter-domain signaling mechanisms. To allow heterogeneous domains to exchange basic control information, a cross-domain signaling procedure should be deployed. Our view of such a cross domain signaling exchange is twofold:
    - one possibility is to deploy a novel standard for cross-domain information exchange. The drawback of such a solution is that the format and the contents of these control packets needs to be standardized, and this may limit the timely deployment of this cross-domain mechanism.
    - A much more simple, and appealing possibility, is to allow the coexistence of the above mentioned cross-domain signaling protocol under development with an implicit cross-domain signaling scheme, based on drop of signaling packets and thus usage of the principle of packet loss as a way to notify congestion. The rationale is that QoS provisioning in the Internet should be outlined following an evolutionary approach, so that each individual domain could independently and asynchronously upgrade its network schemes to provide support for QoS.

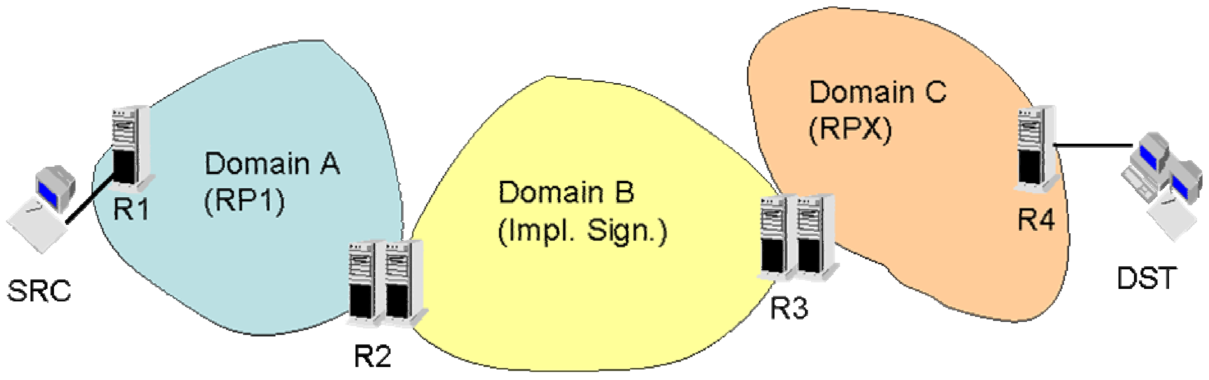


figure 54 - A multi-domain scenario

Note that intra-domain and inter-domain resource reservations are both in the scope of NSIS signaling layer protocols (NSLP). In both cases, a common NTLP should provide common transport facilities to NSIS entities, while different NSLPs should be suitably used, and possibly nested by NFs, in different scenarios (note that, as to the nested protocol, a NF triggering the intra-domain procedure can also be considered a NI, as well as the NF connecting with the following domain, would act like a NR). NSLP translation will be needed in order to setup a reservation through a path crossing several administrative domains, with possibly different network architectures. Note that the case of a stateless domain can be seen by NFs like a single virtual link between two NSIS nodes: the NSLP is aware of the behavior of such a virtual link and operates in a suitable way (e.g., a DiffServ cloud can be seen like a link to probe with GRIP signaling mechanisms - a similar approach was considered also in the IntServ operation across DiffServ [23]).

§4.1.2.1 Explicit-to-Implicit signaling mapping

Consider, as a concrete example, the scenario depicted in figure 54. Here, the source to destination path comprises three different domains, namely A, B, and C, each running a possibly different intra-domain reservation protocol (namely RP1, Implicit Signaling, and RPX). Each reservation protocol has its own scheme. For simplicity of presentation, we assume that domain border routers coincide (e.g. R2 functionally acts as a edge router for both Domain A and B - though in practice these functionalities will be split in the two involved edge routers).

The distinction between inter-domain and intra-domain resource reservation mechanisms is particularly effective in the case of Implicit Signaling. In fact, a source node which wants to setup a QoS flow sends an admission control request to the node, in the first domain, in charge of processing the reservation request (in the example, node R1 in Domain A). The semantics related to the request (traffic specification, QoS requirements) are contained in the signaling packet. This signaling packet carries both application-level information to be used at the destination node, as well as information upon which intra-domain reservation protocols rely on to perform their reservation tasks. Node R1 converts these QoS semantics in procedures supported by the specific intra domain resource reservation mechanism, and triggers the specific edge-to-edge reservation mechanism RP1 running through Domain A. At the end of the reservation procedure, if the domain is capable of admitting the flow, then the signaling packet is forwarded out of the domain by the egress node and forwarded to edge node R2 in the adjacent domain. Otherwise, it is dropped. Assume that Domain B does not support any intra-domain explicit signaling mechanism, but it is capable to support the implicit signaling operation described in the previous sections (e.g, GRIP). Edge node R2 is thus in charge to:

- read the contents of the explicit signaling message;
- find the best match according to the QoS requirement and to the traffic specification, with the probe tagging supported in the domain;
- tag the signaling message as a probe and send it across the network domain;
- in case of successful delivery, edge router R3 will intercept the signaling packet. It will send a feedback packet to R2 in order to complete the implicit signaling exchange, and will invoke the intra-domain resource reservation mechanism for Domain C (if any).

Note that, if the end-to-end signaling path is expected to provide an ack on the same forwarding path, router R2 may wait for ack reception and for piggybacked feedback information necessary for the completion of the Implicit Signaling loop over the end-to-end ack. In the same fashion, RP1 procedures could be completed with a feedback coming from R2 *after* the Implicit Signaling is completed (i.e., signaling procedures could be nested).

Task of the Domain B administrator is to determine a proper matching between the information delivered into the inter-domain signaling packet and the marks - probe(s) and data - used inside the domain. Finally, the signaling packet arrives at the ingress node of Domain C. Here, the ingress node recognizes that the packet is for signaling, but it finds that the packet does not carry information useful for the reservation protocol RPX (i.e., the packet and even the relevant marks are incompatible with this domain inner procedures). Therefore, Domain C can decide to:

- run a generic (e.g., un-parameterized) edge-to-edge signaling procedure;
- drop the packet (i.e., drop the entire flow);
- simply forward the packet, with no admission control, on a best effort basis.

In case of end-to-end reservation procedures with receiver-initiated resource reservation (e.g., this is the case of RSVP, in which resources are reserved by means of a Resv message, sent by the destination node as a feedback for the Path message sent by the source node), the implicit signaling operation can also be triggered by the reception of the message carrying the reservation indication (e.g., the Resv message in the RSVP example) in the upstream direction. In this case, R2 could transparently forward uplink signaling messages; then R2 should parse upstream signaling, store end-to-end signaling messages and start a probing phase with (possibly dummy) probe packets in the downstream direction. If this independent intra-domain signaling session succeed, then the stored end-to-end signaling message is forwarded inside Domain A. Although the above example is very loose, and several problems need a thorough investigation, nevertheless it appears that a GRIP-like tool and its implicit signaling can be the "glue" for the coexistence of highly heterogeneous edge-to-edge reservation mechanisms. Moreover, note that the outlined approach allows the coexistence of domains running a reservation protocol with Best Effort domains. Clearly, the QoS provisioned to the considered end-to-end flow will be bottlenecked by the worst case domain. But in the same time, domains that run a reservation mechanism are capable of limiting the traffic admitted, and thus locally guaranteeing QoS support.

A thorough understanding of this latter issue is of importance. The cross-domain reservation scheme described above is not necessarily aimed at providing an end-to-end QoS support or performance guarantees. Conversely, it is devised to guarantee each domain that the performance encountered by packets crossing the given domain are kept under control (depending on the degree of tightness of the reservation protocol adopted). In other words, our view of the performance provided is domain-centric, rather than an end-to-end guaranteed performance view. Possibly, suitable routing schemes and SLAs can find a path that comprises only QoS aware domains. Note that this is line with the way of operation of other functions in the Internet (e.g., routing), which allow different domains to adopt different schemes.

§4.1.2.2 Implicit-to-Implicit signaling mapping

Mostly interesting is the case of two adjacent domains, both relying on some form of implicit signaling. In fact, to extend the per-flow implicit signaling approach, it is sufficient to provide a mark to mark matching (e.g., DSCP to DSCP, if DiffServ is aimed at) at the interconnection node. Specifically, consider the case illustrated in the figure 55, and assume that the QoS classes supported in Domains A and B are different, either in their number as well as in the supported QoS levels. Via external agreements (where the QoS level semantics are considered), Domain B may determine a best matching between probe labels used in Domain A and corresponding probe labels used in Domain B. A corresponding matching for information labels is also provided. In DiffServ, this translates into a simple DSCP to DSCP table, according to which packets incoming from Domain A are labeled when forwarded

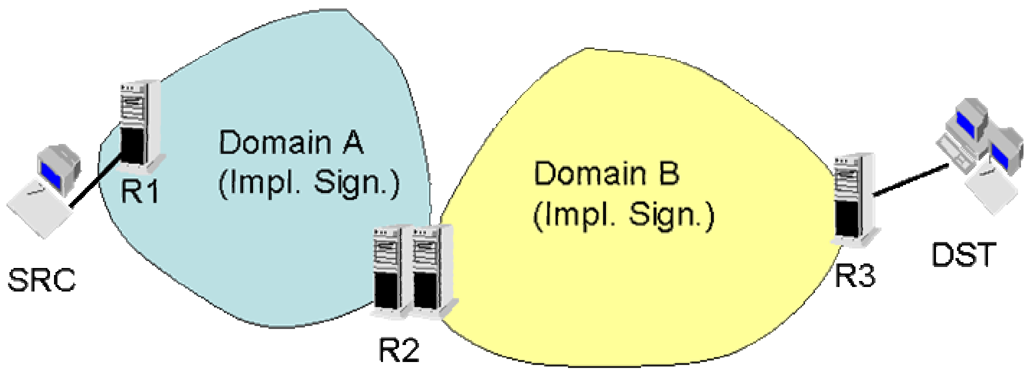


figure 55 - Implicit-to-Implicit Signaling mapping

through Domain B. If performed properly, this marking matching has several important properties. Firstly, probes in Domain A remain probes also in Domain B. This implies that a per-flow admission control request in Domain A is propagated, with no need to parse the content of the signaling packet, also through Domain B. This allows to extend per-flow admission control even through domains connected at gigabit speed, with no need to invoke higher layer entities which would be hardly scalable. Secondly, the domain administrators may rely on a loose matching between QoS levels across different domains. This implies that the QoS model internally deployed in each domain does not need to be the same across different domains. In other words, the extension of the GRIP-like implicit signaling does not automatically implies an extension of the same QoS semantics (which are agreed offline) across different domains.

§4.1.3 Deployment considerations

As discussed in section §3.2, implicit signaling is suited to be deployed over DiffServ network by using the AF specification and GRIP. However, commercial routers do not typically allow to configure the dropping behavior of AF drop levels via bandwidth measurements. In fact, they typically determine the dropping probability via (filtered) measurements of the queue size and RED/RIO policies. The results reported in section §3.3.3 highlight the effectiveness of GRIP over available AF implementations, and much more over, AF router endowed with bandwidth measurement tools. Furthermore, protecting implicit signaling from possible route changes, due to the eventual dynamics of routing protocols, is a task of signaling interoperation protocols such as the NSIS transport layer protocol (see the General Internet Messaging Protocol for Signaling proposed in [89]); we can think to enhance NSLP protocols with additional probing features, in order to periodically send packets over

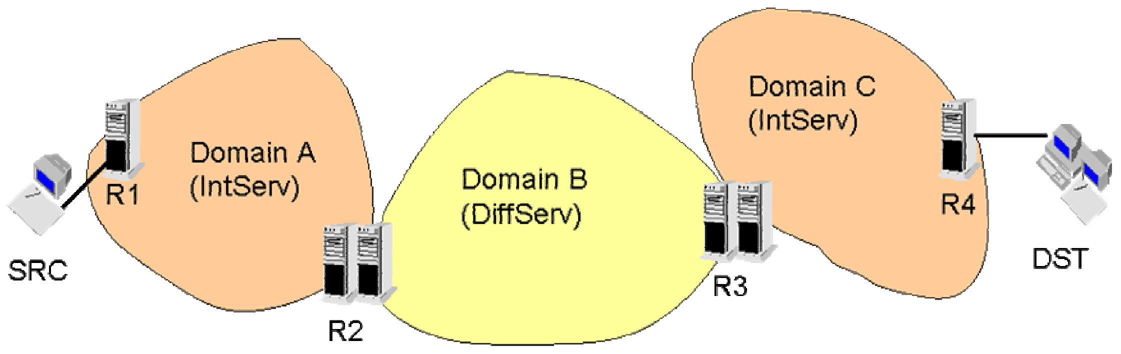


figure 56 - IntServ-DiffServ scenario

NTLP after the setup of a flow to refresh both the inter-domain and end-to-end path. On the other side, DiffServ will be probably deployed in the core network where forwarding mechanisms such as MPLS, will limit the frequency of route changes below typical session duration.

§4.1.4 RSVP transparent signaling across GRIP-DiffServ domains

In this section we will focus on a scenario similar to the one dealt with in §4.1.2. More in detail, the following represents a possible application of the concept presented in §4.1.2. Two alternative solutions for RSVP and GRIP interoperability are then reported.

§4.1.4.1 RSVP over GRIP

Let's consider an RSVP-handled connection between two hosts (SRC and DST – see figure 56), and suppose that the path between the two endpoints traverses three different administrative domains. Domain A and Domain C use a stateful IntServ network architecture, while Domain B is a stateless DiffServ domain.

When the source SRC needs to setup a flow to the destination, it opens an RSVP session by sending a Path messages, that semantically operates like a probe injected in the network. Then, the sender waits for a Resv message, which semantically operates like a feedback. The Path signaling packet injected in the network carries application-level information to be used at the destination node, DST; these information can be read by any RSVP router along the path. Of course, this is not the case of routers inside Domain B, where DiffServ is deployed. When the Path message firstly reaches the ingress node of Domain A (i.e. R1), this node recognizes, in the order, that the packet is a signaling packet, and it contains information usable by RSVP reservation protocol. This signaling packet is meant for end-to-end reservation between SRC and DST (or between R1 and R4, if, for security reasons, reservation is operated by the network provider, once given a user request). Anyway, the Path packet triggers the specific edge-to-edge reservation mechanism (RSVP) through Domain A. If routers in the path segment between R1 and R2 have resource enough to accommodate the connection, then the Path message is forwarded out of the domain by the egress node. In Domain B, instead of RSVP, GRIP over AF PHB is used. In this case the original Path message is simply labeled as a probe packet and forwarded from ingress router R2 to egress router R3. If multiple paired PHBs are available in Domain B for different QoS classes, a SLA between IntServ and DiffServ domains establishes the mapping between the IntServ requested service and DiffServ PHBs. If congestion is not present in Domain B, Path message is not dropped and can be further forwarded to Domain C, where it is recognized as an RSVP signaling message. Thus the RSVP reservation procedure is started also in Domain C, and the connection request can eventually be delivered to DST. Finally, if DST accepts the connection, it sends a Resv message to the sender (or it triggers R4 to generate such a message). The RSVP Resv message crosses back Domain C, then enters Domain B where is labeled as a feedback and not parsed by DiffServ core routers. If this message reaches the egress router R2 before the timeout adopted by the GRIP expires, then the GRIP entity located at R2 forwards the message to Domain A, and reservation procedure for DiffServ domain is completed. Otherwise the Resv message is dropped and SRC will not receive a feedback for its connection request. In case that the admission control in the DiffServ domain gives a positive outcome, the Resv message is parsed by RSVP routers in Domain A and finally by the source. If the reservation procedure gives positive response separately in each domain, then SRC activate the flow by emitting information packets. In case the signaling feedback (Resv) does not arrives back in due time, the flow setup is aborted. The flow message chart relative to the described operation is reported in figure 57.

As to NSIS terminology, SRC acts like a NSIS Initiator (NI), DST acts like a NSIS Responder (NR), while edge routers and RSVP routers operate the NSIS forwarding (NF). Furthermore, edge routers run NSLP translation (from RSVP support to GRIP signaling and vice versa).

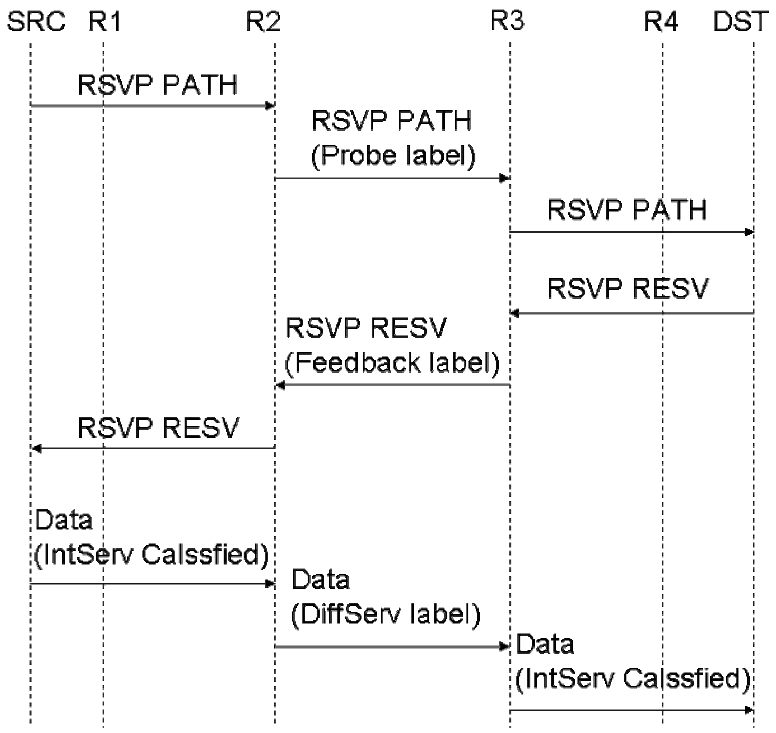


figure 57 - Message sequence chart for RSVP over Implicit Signaling

§4.1.4.2 RSVP plus GRIP

Here we propose an alternative solution to the case depicted in the previous subsection. Let's consider again an RSVP-handled connection between two hosts (SRC and DST – see figure 56), and suppose that the path between the two endpoints traverses three different administrative domains. Domain A and Domain C use a stateful IntServ network architecture, while Domain B is a stateless DiffServ domain. When the source SRC needs to setup a flow to the destination, it opens an RSVP session by sending a Path messages (or it triggers R1 to do it). Then, the sender waits for a Resv message. The Path signaling packet injected in the network carries application-level information to be used at the destination node, DST; these information can be read by any RSVP router along the path. Of course, this is not the case of routers inside Domain B, where DiffServ is deployed. R2 does not affect the transmission of new IntServ flow requests. When a RSVP Path is received by R2 from the IntServ area (sender), this message is labeled with a suitable DSCP (e.g., Best Effort) and forwarded to the final destination, across the DiffServ domain. The same is for Resv message generated by the receiver, when the connection is accepted by the destination: RSVP Resv messages coming from the IntServ area are simply labeled with a default PHB and forwarded to the destination, i.e., the IntServ sender. R2 operation starts as soon as an RSVP Resv message arrives from the receiver, upon the Resv message has crossed the DiffServ core area. When RSVP carries a IntServ reservation requests, the RSVP packet payload specifies the receiver traffic specification (RSpec) and the class of service to be started. Based on these data the router selects the PHB group better suited to accommodate the flow, when/if accepted. The semantics of such a logical mapping are not specific part of the proposed signaling interoperation function, and are based on either end-to-end QoS arguments, as well as internal DiffServ domain administrative management issues. One possible mapping may be implemented as follows: based on the flow descriptor in the RESV message, the R2 selects the PHB group, among the available classes, with the goal of supporting, within a PHB Group, flows with RSpecs as much similar as possible (e.g., maps incoming flows with peak rate in a given range on the same PHB Group; if a class is not available, the connection is blocked). This operation makes it easier to configure, for a given AF PHB group, the congestion threshold within each core router. After the selection of the appropriate PHB group, R2 starts a GRIP session in the scope of the DiffServ domain, towards R3. In the meantime R2 removes the RESV message from the network and stores it in memory, together with the selected DSCP chosen. A GRIP probe is sent and a GRIP timer starts running. When R3 intercepts a probe coming from R2, it invokes the GRIP connection decision criterion and eventually acknowledges the probe packet with a GRIP ack in the reverse path, towards R2. If this ack message does not arrive at R2 within the probing phase timeout, the R2 assumes that the probe has been dropped because of congestion, and thus that the flow setup procedure needs to be aborted. Unlike R2, the R3 node does not need to store any additional state/table other than the standard RSVP state. R3 simply needs a default DSCP to label incoming Resv messages. Thus, if R2 receives a GRIP ack before the GRIP timer expires, the previously stored RSVP Resv is forwarded to its final destination, in the IntServ area. An entry is added in the R2 mapping table, according to information stored with the intercepted Resv message. Such a mapping associates the flow ID (sender/destination, IP/port pairs) to the selected PHB group (specifically, the set of DSCP values used by the DiffServ domain to support the PHB Group). Then the Resv message is processed by the RSVP part of R2, and delivered to the sender, which starts transmitting data packets. Afterwards, as data packets arrive at router R2, they are marked according to the DSCP value reserved for data packets stored in the relevant mapping table.

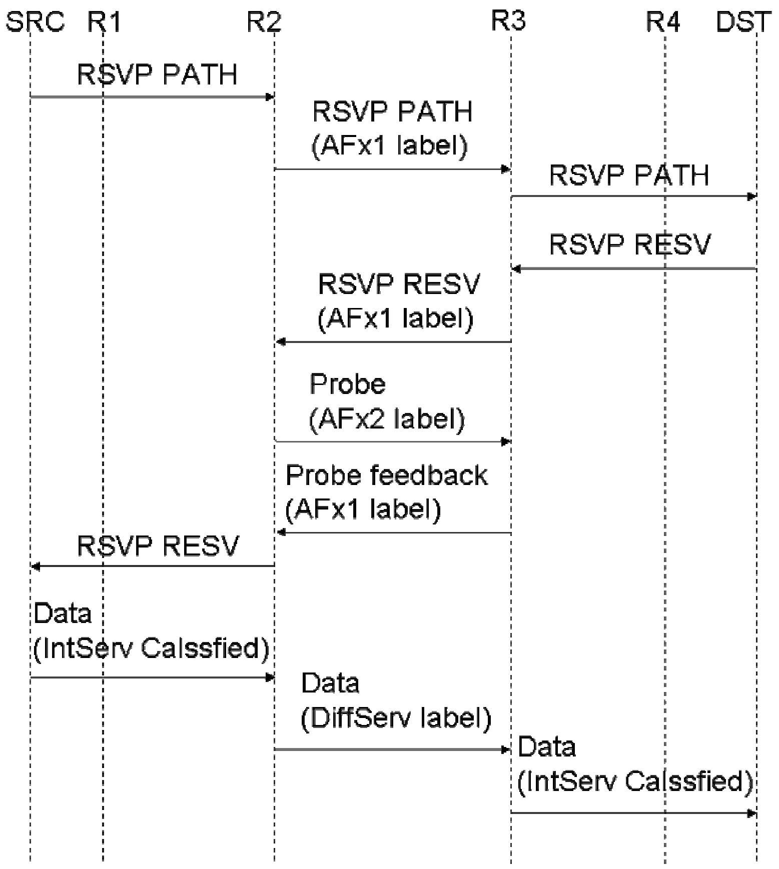


figure 58 - Messages for RSVP plus GRIP. An IntServ service is mapped on an AF PHB

The flow message chart relative to the described operation is reported in figure 58. Again, as to NSIS terminology, SRC acts as NI, DST ply the role of NR, while edge routers and RSVP routers operate the NSIS forwarding (NF). Furthermore, edge routers run NSLP translation from RSVP support to GRIP probe/feedback and vice versa.

§4.1.4.3 Service mapping

Internet access providers usually differentiate their customers on the base of a subscribed contract that specifies both characteristics of the service to be provided and the data flow constraints. What we suppose is that any consumer receives and experiences a QoS level that is consequence of the billing rate. Requests for IntServ services must be mapped into the underlying capabilities of the DiffServ network region. Aspects of the mapping include:

- selecting an appropriate PHB, or set of PHBs, for the requested service;
- performing appropriate policing at the edges of the DiffServ region;
- exporting IntServ parameters from the DiffServ region (e.g., for the updating of AdSpec);

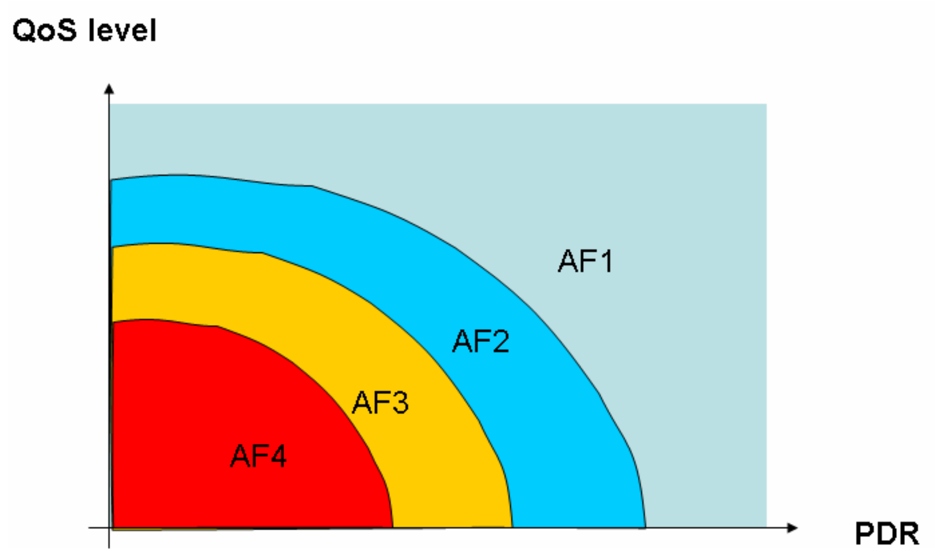


figure 59 – QoS vs. PDR in a DSCP mapping table

- performing admission control on the IntServ requests that takes into account the resource availability in the DiffServ region.

When a new flow asks for resources on the DiffServ network, the interrogated border router (BR) has to decide the class to associate to the flow. This decision is based on the identity of the client and the requested amount of resources. It can also decide to map the flow to Best Effort class or to block that flow. Typical criteria to decide the DSCP to use are peak (mean) data rate, and burst size. Issues to be considered are in the associated Service Level Agreement (SLA). If we define a metric to evaluate a level of service and a metric to evaluate the amount of requested resources we can map the DSCP decision on a two-dimensional plane whose axes are the aforesaid parameters. In figure 59, it is qualitatively shown the result of these operations. In the table reported, each point is univocally identified by a QoS level (that represents SLA) and peak data rate (PDR); the color of the point is the DSCP that will be used. The domain administrator must fill a similar table and store it in a repository. Once a decision were due, the border router would access the repository and check the color (i.e. DSCP) associated to a point (i.e. flow load).

§4.2 The multicast case

In this section we show an application of GRIP in a quite unexpected framework. In fact here we focus on multicast in DiffServ network. The idea to adapt GRIP-like algorithms to support QoS multicast services was envisioned while considering that stateless IP networks architectures are not provided with QoS mechanisms for the differentiation of QoS levels in a multicast application[23]. However, the QoS model envisioned in the DiffServ architecture leaves the network provider free to enforce additional tools, and a widespread discussion is still open in the Internet arena as regards the QoS support in unicast and multicast DiffServ scenarios. Since the DiffServ model looks very promising due to its scalability and flexibility, and considering that DiffServ is now assuming a great importance and, as a matter of fact, it represents the new research direction for the QoS field, we decided to dedicate our effort in the provisioning of an efficient mechanisms endowed with capabilities that support different levels of QoS in a DiffServ network. Hence, it has been proposed a mechanism that copes with the DiffServ architecture design and, at the same time, proposes an evolutionary implementation of a QoS-aware multicast protocol, guaranteeing backward compatibility with existing network architectures and multicast protocols and approaches. A particular implementation of the proposed approach over Linux operative system where DiffServ plus proprietary implicit signaling protocols have been deployed, is also shown in the conclusive part of this chapter. This implementation allows to differentiate two QoS levels for multicast delivery groups: the Best Effort users group, and the QoS users group, as in a premium service strategy. The two main issues in the study and in the real implementation are:

- the differentiation of users inside the same multicast transmission;
- the support of Quality of Service by means of mechanisms able to verify the resource availability before allowing new users to join a QoS group.

It comes into play the problem of supporting dynamic joining of any users to a multicast group while requesting a QoS-aware service, and avoiding multicast resources reservation in compliance with DiffServ principles. However, some additional features can be exploited from the presence of multicast routers, which are a minority, and have to afford some stateful operations. Our solution inherits some ideas from GRIP and combines two ideas: firstly, resource availability along a new QoS path is verified via a probe-based approach; secondly, QoS is maintained by marking replicated packets with a special DSCP value, before forwarding them on the QoS path.

§4.2.1 Multicast applications in DiffServ network and related work

Internet applications extend over a very broad range, with very significantly different burden for the network infrastructures. In particular, as network resources grow and access

[23] The activity presented here was in part founded by the Italian Government in the frame of the FIRB VICOM project.

networks speed up, the interest for new and much more bandwidth consuming application augments. The results is that we are now witnessing the emergence of several Web-based real time multimedia and multicast applications, such as video conferencing, staggered multimedia information retrieval, etc. Among the huge number of proposals that have appeared in the literature in the area of QoS multicast (see [94] and [95]), most of the works concern to the development of multicast QoS routing protocols, as clearly shown in [25], [91] and references therein contained. Instead of solely considering protocols that select multicast paths under QoS constraints, we argue that the issue of endowing current multicast protocols with resource reservation and admission control mechanisms has to be deeply dealt with. Conversely, many works tend to confine these themes to mere and somehow straightforward extension of related unicast protocols. For example, [41] proposes a reservation mechanism for multicast traffic based on a reference Internet model very similar to that proposed in the IntServ approach. In addition, the RSVP protocol specification [18] has been devised to efficiently support multicast traffic. Again, we argue that novel problems arise in QoS multicast when the reference unicast QoS architecture is not based on explicit and stateful resource reservation protocols, like in the case of the DiffServ framework [14]. The potential problems, and the complexity of supporting multicast in a DiffServ environment are sketched in [14] and, in greater details, in [15], [94], [106] and therein contained references. In particular, [15] highlights the following issues.

- Firstly, dynamic addition of new members to a multicast group can adversely affect existing traffic, if resources are not explicitly reserved after each join, since replicated packets get the same DSCP [14] of the original packet and thus experience the relevant treatment consuming unreserved resources.
- Secondly, resources should be reserved separately for each multicast tree associated to a given sender, to allow simultaneous sending by multiple sources, with QoS constraints.
- Thirdly, it appears difficult to support heterogeneous multicast groups, i.e., groups in which different users have different necessities.

More into details, as regards the last point, participants who can cope with a Best Effort service should coexist with participants needing specified and different levels of QoS assurances, so that the same multicast group can deliver differentiated services. For instance, a user could browse a multicast multimedia session in Best Effort mode and then decide to switch to a QoS mode (possibly by paying for it). The paper [94] proposes a solution for QoS support in DiffServ based on tunneling mechanisms (multicast packets encapsulated in

DiffServ), which, in coherence with Internet principles, pushes the complexity to the borders of the DiffServ domains. The work presented by paper [107] devises a solution which succeeds in maintaining the integrity of negotiated SLA by means of weighted traffic conditioning and receiver-initiated marking schemes; admission control operations are performed by suitably signaling network management entities (e.g., Bandwidth Brokers, or suitable functionality within ingress routers). In turn, this solution also supports heterogeneous requirements within a multicast group by encapsulating the markings for each of the branches with such heterogeneous QoS requirements in the packet header, at the ingress router of the domain.

In the following of the present chapter it is our intention to propose a QoS-aware multicast solution ready to be deployed in DiffServ IP networks, even though some additional features could be upgraded later on. The goal of our proposed solution is twofold:

- to provide flexible QoS support with respect to heterogeneous multicast groups, and
- to maintain compatibility with currently deployed multicast protocols, with specific reference to PIM (Protocol Independent Multicast) [35], [40].

Furthermore, our solution aims at furnish:

- QoS guarantees to the final user;
- benefits for service providers, in terms of both traffic reduction (e.g., by avoiding flow duplication) and management tools (e.g., service negotiation with the user);
- open source implementation over Linux routers.

§4.2.2 The PIM-SM multicast protocol and QoS extensions

First of all it has to be defined a multicast protocol that could be used in a DiffServ network. Thus, as a starting point for the introduction of QoS features for multicast, we have selected PIM (Protocol Independent Multicast), in the Sparse Mode (SM) version [35], [40], as the reference multicast routing protocol. The advantages of PIM stay essentially in its independency on the underlying unicast routing protocols, and in the fact that it allows several different configurations of the multicast distribution tree: central core router configuration are possible, where a unique group has at its disposal a shared distribution tree; in alternative, different distribution trees can be used per specific source sending to a group; the coexistence of shared and source-specific trees for the same group is also permitted. Moreover, PIM-SM is suitable in order to introduce QoS as an incremental addition to existing multicast protocols, allowing backward compatibility; PIM-SM can also be adopted without relevant modification in the DiffServ router operation and in the underlying routing protocols, resulting in a flexible solution with respect to heterogeneity of users within a group, and of source/core-based distribution trees.

To provide QoS multicast over DiffServ, two important issues need to be addressed, and it must be done by simply applying different forwarding disciplines to packet aggregates, based on their DSCP value [14]. The first issue is how to differentiate the service level supported on different paths inside the multicast distribution tree. The second one is how to provide an admission control function in order to support dynamic joins, from QoS-enabled users. To solve the first issue, the authors of paper [15] suggest to add an additional field in the Multicast Routing Table (MRT), which specifies the DSCP(s) to be used for each output link. It is thus possible to specify whether the next hop is QoS-enabled (i.e., mark packets with a specified DSCP, corresponding to a negotiated QoS level), or not (i.e., mark packets with the Best Effort DSCP). This solution allows heterogeneous users to share the same multicast distribution tree, as well as it allows deployment of several different QoS levels in the network. Our solution inherits from [15] this marking strategy. However, this handy tool is not sufficient by itself to provide QoS-aware and differentiated services in a multicast environment. In our proposal, we combine a marking strategy very similar to that proposed in [15] with a data-plane DiffServ-compliant admission control function, i.e., the GRIP mechanism presented above, that can be adapted within the PIM-SM framework.

By assuming PIM-SM as the multicast protocol, a router, called Rendezvous Point (RP), is used for traffic sources S, as the root of the distribution tree for the multicast group. Hosts Hi access Designated Routers (DRi) on their LAN, which act on behalf of those hosts as far as the PIM-SM protocol is concerned. In other words, the DR manages, on one side, all local group management information (e.g., via the IGMP protocol), and, on the other side, it emits PIM join/prune messages towards the RP. We assume that all routers are DiffServ capable, while a few routers (A, B and C in figure 60) additionally support PIM. Intermediate DiffServ routers, indicated in figure 60 with the label "R", are transparent to the multicast protocol. Multicast routers are the node of the multicast tree, thus replicating each packet received, in order to deliver it to every multicast destination. Routing information are provided by the PIM protocol operation (see [40]) and stored in a table called Multicast Routing Table (MRT). Of course, multicast routers are stateful, according to the standard multicast protocols, while DiffServ routers are stateless. In this scenario, scalability derives from the fact that only a fraction (possibly small) of the network routers should be multicast capable.

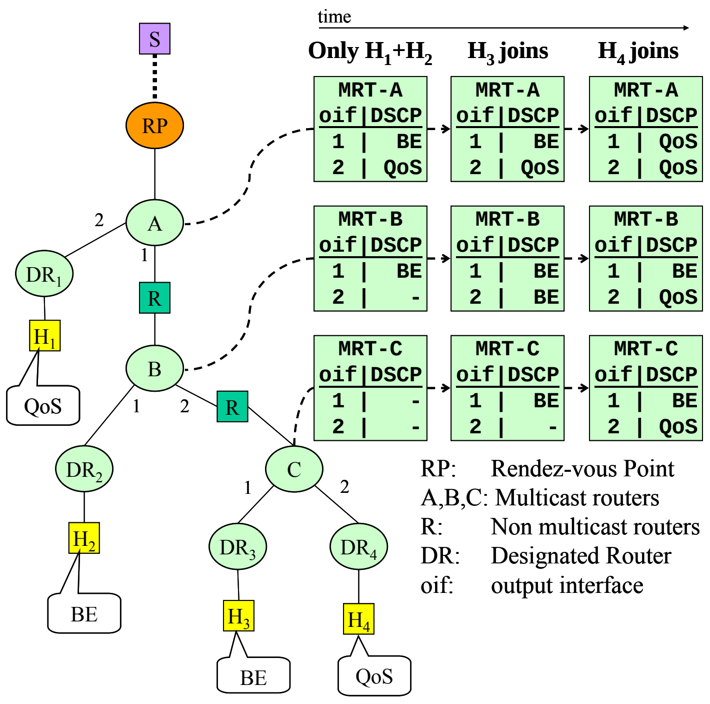

time
Only H_1+H_2
H_3 joins
H_4 joins
S
RP
A
B
C
R
R
DR_1
DR_2
DR_3
DR_4
H_1
H_2
H_3
H_4
QoS
BE
BE
QoS
1
2
MRT-A oif|DSCP 1 | BE 2 | QoS
MRT-A oif|DSCP 1 | BE 2 | QoS
MRT-A oif|DSCP 1 | QoS 2 | QoS
MRT-B oif|DSCP 1 | BE 2 | -
MRT-B oif|DSCP 1 | BE 2 | BE
MRT-B oif|DSCP 1 | BE 2 | QoS
MRT-C oif|DSCP 1 | - 2 | -
MRT-C oif|DSCP 1 | BE 2 | -
MRT-C oif|DSCP 1 | BE 2 | QoS
RP: Rendez-vous Point
A,B,C: Multicast routers
R: Non multicast routers
DR: Designated Router
oif: output interface

figure 60 – Example network scenario

Following [15], we also assume that each MRT stores, for each router output interface (oif), an additional entry. This entry represents the DSCP value, which is used to mark replicated packets that are forwarded along the considered interface. The figure 60 shows, in the leftmost column, labeled "Only H_1+H_2", the MRT states for routers A, B, and C, in the assumption that only hosts H_1 (QoS-enabled) and H_2 (Best-Effort) are members of the multicast group. In the figure we adopted the label "BE" to denote a best effort service, and the label "QoS" to denote a service with pre-defined QoS requirements. For the sake of simplicity, in what follows we assume that the network provides a single QoS level, in addition to the best effort service. Again, by looking at figure 60, we see that the MRT of router A marks packets directed to host H_1 (interface oif 2) with a QoS marking, while it labels packets forwarded to interface 1 as BE. In the MRT of router B, clearly, only the interface 1 is active, while router C has no multicast state, at the moment. Let us now discuss what happens when, firstly the BE host H_3, and then the QoS host H_4, join the multicast group.

In the case of H_3, we adopt a standard PIM join operation. The Designated Router of H_3, DR_3, sends a PIM Join(G) message towards the RP. When a multicast router receives this message, it adds a new entry to its multicast routing table (MRT), with a BE value, for the associated DSCP marking field. This state will indicate that future datagrams destined for group G must be marked as BE, and forwarded on the interface where the join message came from. The Join(G) message is regenerated upstream along what we define as the Branching Path. This is the path from the requesting DR to the first router that already has a Join(G) state for that group, which is called Branching Router. Eventually the Branching Router could coincide with the RP itself. In the specific case of figure 60, the Branching Path goes from DR_3 to router B, which is the Branching Router, since it already has a Join(G) state created by the Join message coming from host H_2, by way of DR_2. The second column of figure 60, labeled "H_3 joins", shows the modified multicast routing tables in routers A, B and C after a successful join of the BE host H_3. In the case of QoS-host H_4, a fundamental difference is the extent of the Branching Path. If the Designated Router DR_4 were to send a standard Join(G) message, the Branching Router would have been router C, since it already has a Join(G) state. However, when a QoS host wants to join the group, it needs to send a modified[24] join message, hereafter referred to as QJoin(G) message, which explicitly carries the QoS level the host is requesting. This QJoin(G) message travels upstream until it either reaches the RP, or a router with a QJoin(G) state (i.e., a state that instructs a router to send QoS marked traffic over at least one of its interfaces). We define QoS Branching Path the path from the Designated Router to the first router that already has a QJoin(G) state for that group. We name such a router as QoS Branching Router (note that it may be the RP itself). In figure 60, the QoS Branching Router for host H_4 results to be router A. The rightmost column of figure 60, labeled "H_4 joins", shows the modified multicast routing tables in routers A, B and C after a successful join of the QoS host H_4.

As a side comment, note that now the distribution tree has a QoS path from RP to DR_4. Therefore, hosts H_2 and H_3 will receive a BE service only on the last hop of their path. In other words, the fact that some members of the multicast groups require QoS, indirectly brings benefits also to BE users, even if they are not paying for such benefits.

§4.2.3 Integration of GRIP-like operation

Joining a QoS group requires a QoS-aware procedure to be run over the path where the QoS service has to be extended, i.e., the new QoS Branching Path. Provisioning of QoS in DiffServ networks is frequently assumed to be accomplished via static over-provisioning (i.e., over-dimensioning of core network links with respect to the expected offered traffic). When dynamic provisioning of DiffServ domains is considered, the traditional approach is to rely on

[24] The new PIM messages considered, QJoin(G), Confirm(G), QPrune(G), can be built upon the PIM message format specifications, as extensions. In fact, PIM defines only height different packets, while a four-bit packet type field (i.e., sixteen possibilities) is available.

centralized control entities, often referred to as Bandwidth Brokers (BB). A BB is an agent that has sufficient knowledge of resource availability and network topology to make admission control decisions. Border routers use control protocols to interact with this agent. The existence of BBs has been assumed in [15], [107] to manage multicast traffic handling strategies. Conversely, we have shown that per-flow admission control can be supported on stateless DiffServ domains, with no need of managing per-flow states within each core router.

With reference to the example discussed in figure 60, in our presentation we focus on the establishment of QoS on the QoS Branching Path from router A as a starting point, to designated router DR_4 as a destination point[25]. Our proposal consists in adapting GRIP to the multicast case. Although we remark that the GRIP router operation can be extended to support several levels of QoS and different traffic classes, here we require that GRIP router support at least three different DSCP values: BE packets, QoS information packets, and QoS probing packets. Let us look back at figure 60 remembering that we want to establish QoS, considering the network router A as a starting point and the designated router DR_4 as a destination point. When the QoS Branching Router A firstly receives a QJoin(G) message, it sends a GRIP probe down on the multicast tree. As reviewed above, the probe is a normal packet, which is distinguished from information packets via a special DSCP marking. Specifically, if QoS packets are marked with DSCP value, say X_1, GRIP probes are marked with a different DSCP value, say X_2, which is logically associated to X_1. For each QoS level, a different pair of DSCPs should be reserved. All network routers support a DiffServ PHB which consists in letting packets marked as X_2 go through only, for instance, if the load run-time measured on X_1-marked packets is lower than a given threshold. The result of this operation is that a GRIP probe is received at the destination DR_4 only if all the routers along the path are found in an non-congested state, defined with a suitable criterion. When the GRIP probe arrives at DR_4, a Confirm(G) message is sent back as a feedback packet, to notify the sender node A that all the routers along the QoS Branching Path can accept a QoS connection. When the Confirm(G) message is finally received at router A, the multicast data can be forwarded on the new path with QoS guarantees. As discussed in the previous section, subsequent replicated packets are marked according to the DSCP value specified in the Multicast Routing Table.

[25] We assume that QoS on the last hop $DR_4 \rightarrow H_4$ is provided by some layer-2 mechanism, or, in other words, that when the host H_4 wants to join the multicast tree with QoS, a local admission control function on the last hop segment has been already performed with positive answer.

This basic operation appears a straightforward extension of GRIP. However, there are several details than need to be clarified. A first problem is that, although the aim of the GRIP probe is to check whether the point-to-point communication from A to DR_4 can support QoS, in practice it is necessary to route the GRIP probe as a multicast packet. In fact, the GRIP probe needs to follow the same path of the multicast data packets. This problem can be solved by using a multicast packet type, for the GRIP probe, and by updating the multicast state within each multicast node so as to route the GRIP probe down to the appropriate output interfaces only. This is accomplished by extending the PIM-SM state machine associated to each multicast router interface, to manage some additional states per QoS class. A simplified version of the extended PIM-SM finite state machine is represented in figure 61, where the four relevant states are reported. The semantic of the plot reporting the PIM-SM finite state machine is explained as follows:

- *Idle state*: this is the default state for a service, and it denotes the absence of joined users in the branching path departing from the router itself. If not refreshed, any other state is switched to Idle after a timer expires. However, explicit transitions to Idle state can be enforced by means of pruning messages.
- *Join state*: if one or more users join a multicast group in the path which the router belongs to, the router switches in this state. Information packets are replicated and forwarded by the router in the appropriate interface, in a BE fashion. The Join state is reached upon the reception of a Join message, or if a probing attempt does not succeed, or also if a QPrune message asks for the downgrade of a QoS group.
- *Probing state*: when a multicast router output interface is in the Probing state, it means that a QJoin message has been received, but no Confirm message has been received yet. The PIM-SM also enters this state upon an already joined user asks for QoS upgrade: the probing procedure is performed as in the Probing sate, while going on experiencing a BE multicast service; if a Confirm is received within a timeout, the multicast delivery is enhanced to the requested QoS level, i.e., the PIM-SM enters the QoS Join state described below. Otherwise, the PIM-SM switches back to the Join state.
- *QoS Join state*: this state is activated after a Confirm message, and it implies that the router output interface is QoS-enabled. When in the QoS Join state, the multicast routing table is set as described in the previous section.

It is worth noting that, upon probing phase is started and not yet confirmed nor aborted, multicast traffic packets are forwarded according to the following rules:

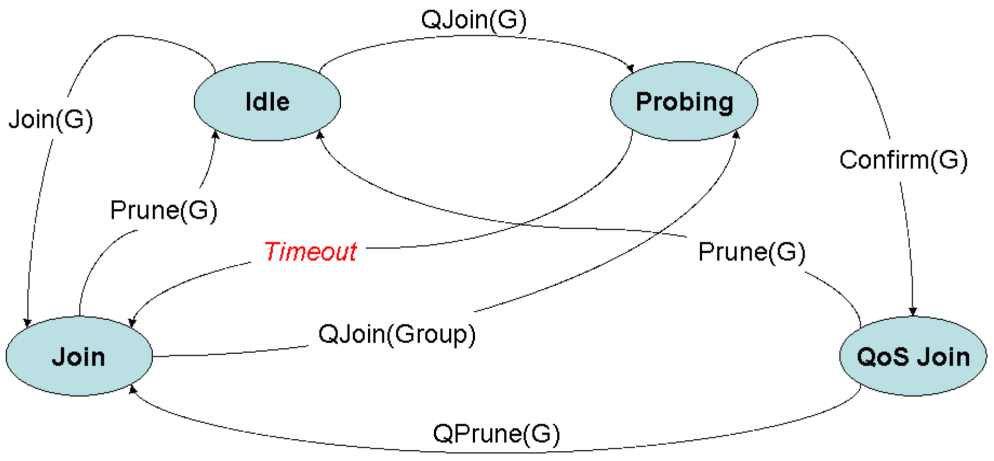


figure 61 – PIM-SM extended finite state machine

- Packets labeled as information packets are replicated and forwarded according to the existing Multicast Routing Table. With reference to the example of figure 60 while host H_4 is joining the multicast group (i.e., before a Confirm(G) has arrived), routers A and B continue to operate according to the previous MRT (central column in figure 60), i.e., they replicate and forward packets labeled as BE, while router C does not replicate multicast data packets on the output interface 2.
- Probes are recognized based on their special DSCP value. They are forwarded only toward interfaces whose state is Probing. With reference to the considered example, a probe packet is generated at router A and forwarded only on interface 1, while the same probe packet is forwarded only on interface 2 at both routers B and C.

The extended PIM-SM finite state machine is illustrated in figure 61, in the assumption of a single QoS class. In case of more QoS classes, each class will have its own Probing and QoS Join state. Moreover, for convenience of illustration, we have omitted the PIM-SM Prune Pending state, and a similar QoS Prune Pending state necessary to handle the QoS class. Entrance in this state occurs when a Prune(G) message or, in the case of QoS, a QPrune(G) message is received. The system remains in this state until a timeout expires, or a Join(G), or QJoin(G) in the case of QoS, is received to refresh the multicast state. In fact, we recall that multicast states are "soft" i.e., as long as a group G is active on a router interface, the router will periodically receive Join(G) or QJoin(G) messages in order to refresh the relevant states. When information transfer for this group is no longer desired, the requiring entity should stop sending joins for that group, so that the relevant state expire. In alternative, an explicit PIM Prune(G) messages can be sent towards the RP for that multicast group. Furthermore, any Join or QoS Join state is absorbed in the Idle state if not timely refreshed.

§4.2.4 A real test bed implementation

A real implementation of the theoretically proposed QoS-aware multicast scenario has been provided[26], in accordance with the description given in the sections above. The implementation of multicast protocols and QoS modules over DiffServ routers is performed by means of PCs equipped with Linux Red Hat 9.0 operating system, and by tackling four main points:

- managing QoS-joining/pruning procedures in a dynamic way - this requires a new joining method for QoS users;
- enforcing admission control over those branches of the multicast tree that do not accommodate QoS traffic, but that are requested to accommodate QoS traffic for a new user - this requires the introduction of a GRIP-like mechanism in the multicast routers;
- extending PIM-SM signaling and messages, in order to distinguish BE and QoS requests - in doing this, we have to introduce new PIM-SM messages, without modifying the existing ones; furthermore, the PIM-SM finite state machine has to be upgraded, including pending state while probing the path;
- marking packets and router interfaces in order to differentiate between BE and QoS traffic and output router interfaces where packets have to be accordingly marked.

In this scope, the PIM-SM has been enhanced with a new module for QoS, whose purpose is the dynamic creation and maintenance of the QoS-aware interfaces in the multicast tree. This Q-module, as depicted in figure 62, operates in parallel with legacy PIM-SM modules. The Q-module creates a multicast tree for QoS transmission, also referred to as QTree, while the PIM-SM creates the legacy multicast tree for best effort services. The Q-module handles some additional messages, for QoS joining and pruning, plus the probing message:

- *Probe*: the GRIP-like message tagged with a low priority DSCP, and injected in the multicast tree by a QoS branching router;
- *QJoin*: this message is generated by a DR, which is a leaf in the multicast tree, and it climbs up the multicast tree until a QoS-enabled router is found (the QoS branching router); QJoin signals the will of a DR to begin part of the QTree;
- *QPrune*: as for QJoin, but for pruning purposes; it is meant to downgrade the service from the QoS to BE level.

[26] The test bed has been realized in collaboration between University of Palermo, Italy and CRES (Monreale, Palermo – Italy) end it was partially founded by the Italian FIRB VICOM project

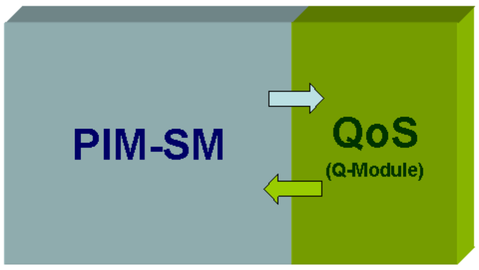


figure 62 - A schematic representation of the extended PIM software

Additionally, the Q-module introduces some new internal states, that are combined with the existing PIM-SM states; thus the states previously mentioned and depicted in figure 61 are generated. The PIM-SM module soft states are:

- *Idle state*: no users are joined to the multicast group;
- *Join state*: one or more users joined a multicast group, and a Join(G) message crossed the router;

The additional Q-module soft states are:

- *QoS request*: a probe has been generated or has crossed the router, and the probing phase is still active; this state corresponds with the Probing state of the overall system;
- *Probe waiting*: when a DR generates a QJoin(G) message, it starts waiting for a probe from the nearest QoS branching router; this is also embedded in the Probing state of the overall system, since the router has been crossed by a QJoin(G) message (the one that the DR router generated in order to contact a QoS branching router);
- *Confirm waiting*: this is the state in which the QoS branching router switches upon the reception of a QJoin (G) and a probe has been sent back to the DR that originated the joining message; Again, this state is embedded in the Probing state of the extended PIM-SM finite state machine, since it is generated upon the reception of a QJoin(G) message;
- *QoS Join*: this corresponds to the Join state, but data belonging to the group are handled in a higher priority class with respect to BE packets; the QoS Join state is reached after a Confirm message crosses the node while in Probing state. The Join and QoS Join state are differentiated by means of a boolean flag, which name is QOS_SET: if the flag is true, the state is QJoin, otherwise it is simply Join.

PIM-SM modifications and integrations are possible thanks to the adoption of an open source software from University of Southern California: Pimd v.2.1.0. This is a standalone implementation of PIM-SM v.2.0 protocol, in compliance with [35]. Pimd installs in a Linux router a daemon interacting with the Linux kernel through system calls, and it is in charge of creating, managing and maintaining the multicast tree, both in the case of source-based tree and shared multicast tree. PIM and IGMP messages are managed by the Pimd software, as well as routing mechanisms, multicast routing tables and virtual interfaces.

Some schemes representing different configurations of our test bed are proposed in figure 63 to figure 65. There are: a Rendezvous point (RP), representing the root of the multicast tree; two multicast router (R), operating as branching points in the middle of the multicast tree; two designated routers (DR), each collecting joining/pruning requests from a LAN and relying multicast traffic in the same LAN. Finally, three hosts (H_1, H_2 and H_3) represent the multicast user population. Additionally, not reported in the figures, a multicast server is connected to the RP router and generates multicast traffic for hosts H_1, H_2 and H_3. Each router in the test bed is provided with Pimd software. Hosts require to join a multicast group using a common IGMP message reaching the DR the host is attached to. BE and QoS-aware multicast command are provided with a suitable DiffServ marking. In particular two DSCP are used (plus one for probing), with different priority in the DiffServ semantic. Packets belonging to QoS sessions are labeled with the high priority label. In the same fashion, QoS requests in each LAN are differentiated from BE requests with the same mechanisms. In what follows we comment the operation of each element in the test bed, while reporting some operating examples.

Firstly, the correspondence between the multicast tree and the QTree built up by legacy PIM-SM modules and Q-module respectively, is depicted by means of an example in figure 63. The nodes involved in the two trees are the same, but in the QTree some node can be temporarily deactivated.

With reference to the multicast tree of figure 63, in figure 64 the three phases of the QoS-aware joining process is reported. At the beginning the multicast topology is the one proposed in the leftmost side of figure 63. In the first window of figure 64, the one labeled as “QoS request messages”, a user under the scope of DR_1 asks for the joining to the G group. It additionally asks for QoS by suitably marking the IGMP packet conveying the request with the IP header field TOS = 40, which is the TOS value we selected for QoS purposes. Then, the router DR_1 switches to the Probing state, and in particular in the Probe waiting mode. In the meantime, DR_1 creates a QJoin(G) message and sends it upstream in the multicast tree. While crossing multicast routers, the QJoin(G) message enforces the Probing state in each multicast node. The QJoin(G) message is eventually sunk by the first router that is QoS-enabled, i.e., the QoS branching router, which in the example corresponds to the Rendezvous point.

Then the second window in figure 64 comes into play. In fact, upon the QJoin has fulfilled its task, a probing phase is originated by the QoS branching node RP in the downstream direction. Meanwhile, RP enters the Confirm waiting mode. The case in which the probe reaches the destination (DR_1) is shown in the figure. If it is not the case, the setup procedure aborts after a timeout, and each node crossed by the QJoin(G) transits in the Join state for group G, on the opportune interface. When the probe is received by DR_1 before the timeout expires, this router switches from the Probe waiting mode to the QoS Join state, since the request is locally accepted. Furthermore, as shown in the third windows of figure 64, DR_1 sends a Confirm(G) to the QoS branching router, and each multicast node that sees the Confirm(G) (before a local timeout expires) switches to the QoS Join state. This message is eventually received by the QoS branching router who sinks the message and removes the Confirm waiting mode.

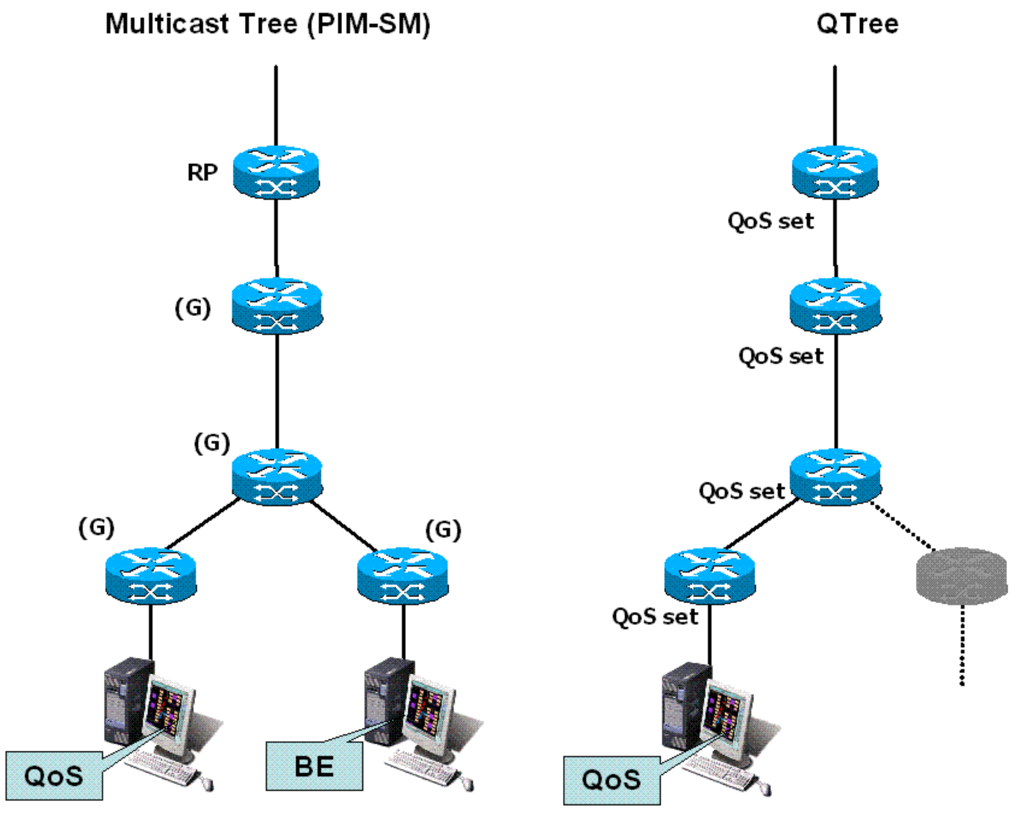


figure 63 - Logical splitting of normal multicast tree and its QoS subset

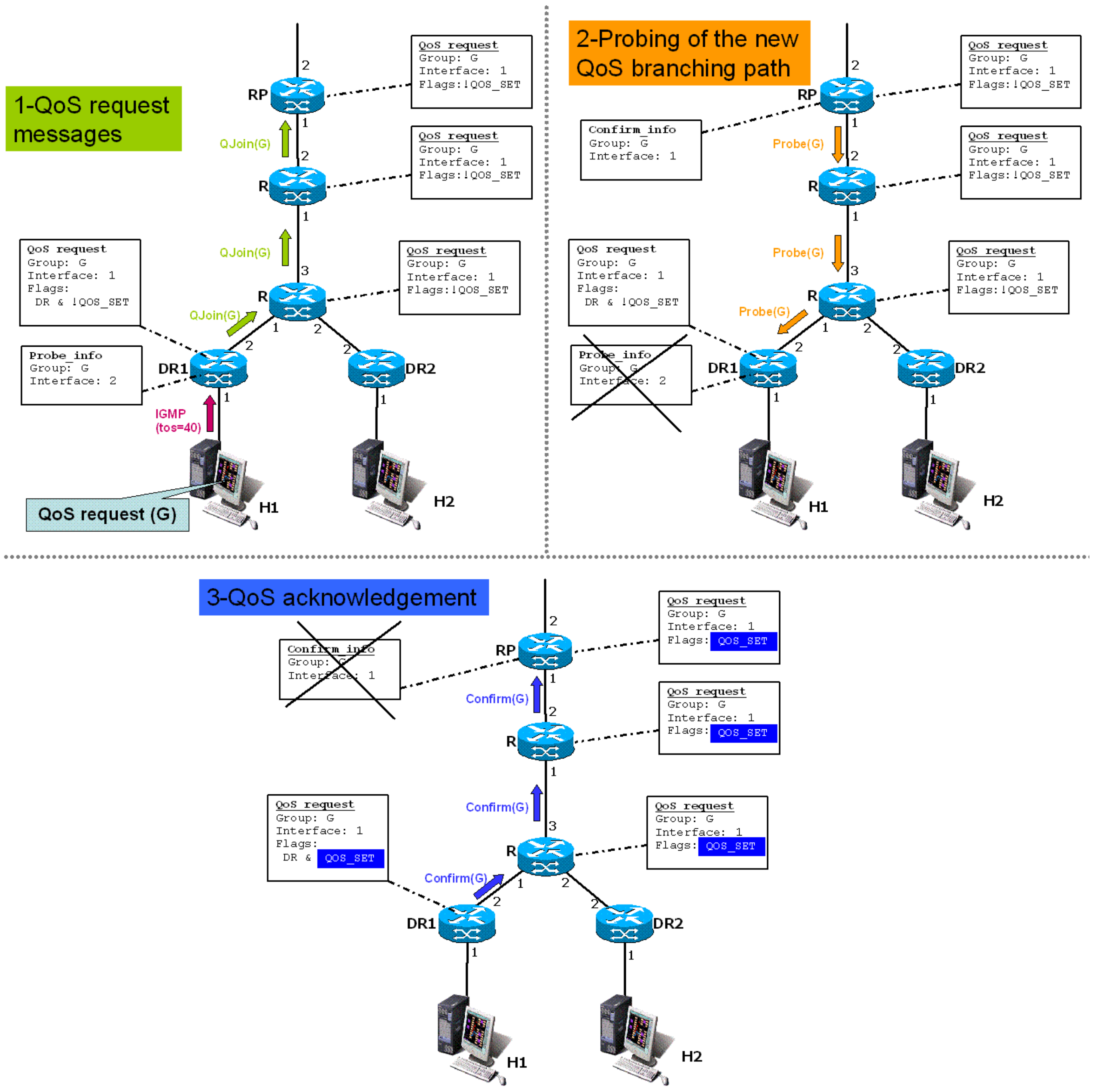

1-QoS request messages
2-Probing of the new QoS branching path
3-QoS acknowledgement
RP
R
DR1
DR2
H1
H2
QJoin(G)
Probe(G)
Confirm(G)
IGMP (tos=40)
QoS request (G)
QoS request Group: G Interface: 1 Flags: !QOS_SET
QoS request Group: G Interface: 1 Flags: DR & !QOS_SET
Probe_info Group: G Interface: 2
Confirm_info Group: G Interface: 1
QoS request Group: G Interface: 1 Flags: QOS_SET
QoS request Group: G Interface: 1 Flags: DR & QOS_SET

figure 64 - Messages used for an accepted QoS request: 1)QoS IGMP request and QJoin; 2) probing phase; 3) the confirm messages

Finally, a pruning example is proposed in figure 65 in case of group G1 has to be removed in a branch of the QTree. The leftmost part of the plot represents a multicast tree with QoS enabled groups G1 and G2, in the branch including routers RP to DR1. Suppose that user H_1 is the only participant in group G1, and that H_1 wants to leave the QoS multicast reception, and switch to a BE service (e.g., this could be done to save money in a traffic based billing framework, or in the case that H_1 has not enough resources available for the handling of a QoS stream). In the rightmost part of the picture it is shown how H_1 sends an IGMP packet conveying a QoS prune message (a prune IGMP that is suitably labeled in accordance with the DiffServ paradigm). Upon the reception of the IGMP request, the Q-module in DR_1 turns into "false" the QOS_SET flag and sends a QPrune(G1) message upstream. Thus, each multicast router encountered by this message sets QOS_SET = FALSE. At the end, the QPrune(G1 message reaches the QoS branching router for group G1 and the QoS multicast transmission is definitely stopped over the entire branch (RP to DR_1, in the example). Note that the multicast state for other groups than G1 is not modified in any router.

§4.3 Conclusions

In this chapter, we have defined a framework to extend per-flow signaling across a stateless domain, by means of a scalable connection blocking service (section §4.1). Our approach requires the stateless domain (e.g., DiffServ) border routers to accomplish the following tasks: i) assign a mark set to incoming requests, based on its QoS specification and the QoS provisioned within the domain (for DiffServ, assign PHBs); ii) tunnel signaling packets through the stateless domain. Our solution does not require the presence of centralized entities as bandwidth brokers, but the edge-to-edge adaptation of the GRIP mechanisms. It is worth noting that, being based on pure data-plane operation (versus control-plane operation), our solution does not violate the DiffServ paradigm. Thus, to provide a more detailed presentation, the investigation has specifically considered the interoperation between RSVP and DiffServ. However, the key ideas of our proposal can be easily applied to any other hop-by-hop signaling which can be considered in the ongoing IETF activity about signaling (NSIS).

In the second part of the chapter (section §4.2), we proposed a modular approach to develop a QoS multicast solution for DiffServ domains, starting from DiffServ and GRIP. Our multicast solution is built on top of existing multicast protocols (PIM) and allows support for both Best Effort and QoS multicast streams The possibility to upgrade or downgrade the service while running is additionally provided. In order to deal with QoS-related issues, we defined a complementary module for PIM-SM, and we called it Q-module. The details of the resulting finite state machine was provided, jointly with the new procedures and messages needed by our solution. However, we remark that our solution could be deployed starting also from other existing multicast models, such as Core Based Tree and Source Specific Multicast. Eventually, a real system has been deployed and tested, based on Linux operating system and Pimd software. A number of possible issues deserving future research include the extension to: i) the support of multiple QoS levels, especially those appearing when different PHBs are simultaneously operating on the same link; ii) the support of the so-called Limited Effort (LE) PHB, to protect already existing traffic (including BE) from new users joining the multicast tree; iii) end-to-end, inter-domain issues, in accordance with the signaling extension procedure reported in section §4.1; iv) fault tolerance.

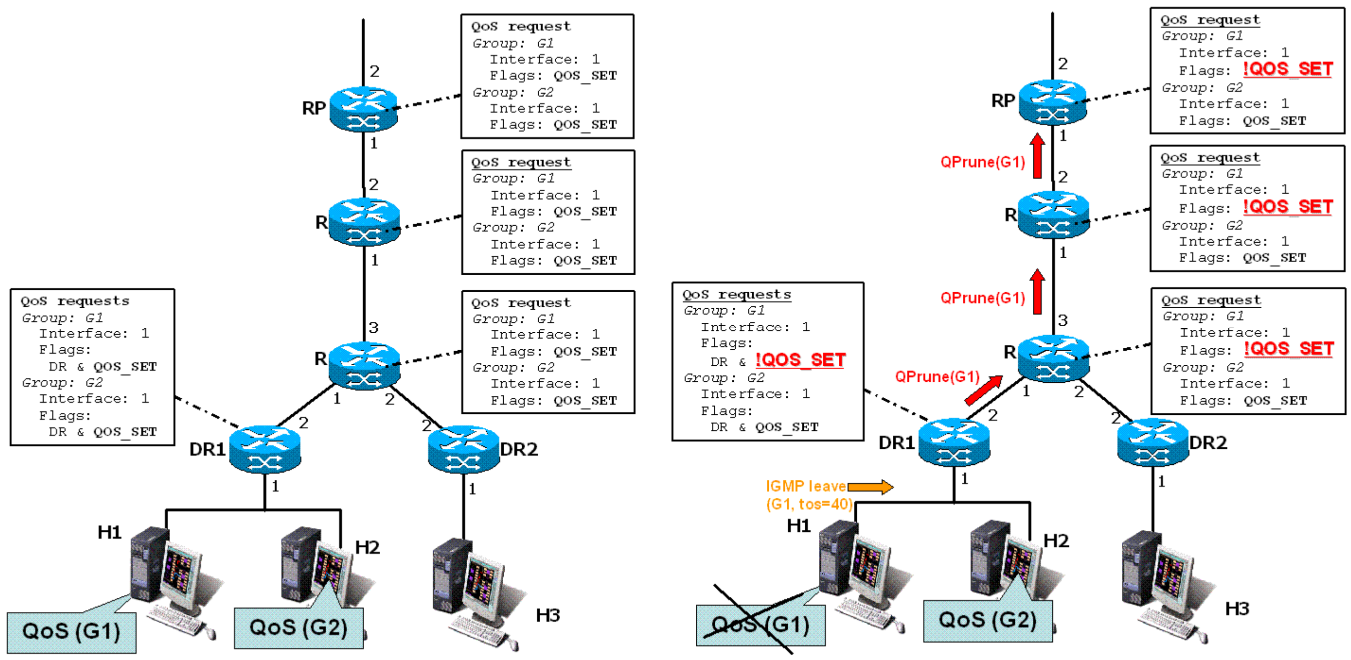

RP
R
R
DR1
DR2
H1
H2
H3
QoS (G1)
QoS (G2)
QoS request Group: G1 Interface: 1 Flags: QOS_SET Group: G2 Interface: 1 Flags: QOS_SET
QoS requests Group: G1 Interface: 1 Flags: DR & QOS_SET Group: G2 Interface: 1 Flags: DR & QOS_SET
QoS request Group: G1 Interface: 1 Flags: !QOS_SET Group: G2 Interface: 1 Flags: QOS_SET
QoS requests Group: G1 Interface: 1 Flags: DR & !QOS_SET Group: G2 Interface: 1 Flags: DR & QOS_SET
QPrune(G1)
IGMP leave (G1, tos=40)

figure 65 - Example of pruning from a QoS Group G1

Chapter V

Streaming and interactive traffic in Vehicular Area Networks

In this chapter we change the perspective from which QoS issues are analyzed. We switch from the admission control point of view to the resource sharing one. In the chapters above we dealt with the problem to decide whether a new service can be admitted to the network, now we focus on the way an already admitted service has to be supported by the network. In fact, active services has different requirements and can suffer in different ways the possibly changing connection conditions. Additionally, admission control schemes can hardly deal with fluctuations in the available bandwidth, as in the case that wireless connections come into play. Thus, an important role is played by adaptive techniques that allow active applications to fairly share the overall amount of resource available at a given time instant. Consider, for instance, a mobile user that accesses two IP network services in parallel, let's say that the user tries to retrieve a file from a repository, and in the meantime it wants to listen to its favorite radio program which is broadcast also over IP. If the wireless access network are enough to accommodate both services, the mobile user can start to receive the requested file and the broadcast program. However, if the user moves to an area in which the wireless connectivity degrades, it can happen that one or both the services could experience bad performance. But, while the file download could go on with a reduced amount of bandwidth, the real time service significantly declines its performance if a minimum amount of bandwidth is not guaranteed. Of course, the problem becomes harder and harder as the amount of data to be dealt with grows, as in the case of (high definition) TV broadcast. In this occurrence it would be fine to exploit the potentialities of a bandwidth manager located close to the user, or in the user itself, that allows to redistribute the resource share ratio between active applications, also taking into account the nature and the priority of concurrent services. In the previous examples, a solution should be to reduce as much as possible the bandwidth of the file transfer process, while leaving to the real time stream at least the minimum bandwidth required to avoid annoying noise effects in playing the received stream. Hence, in this chapter we will analyze the effect of bandwidth sharing schemes on service performance. In particular we will focus on the improvements due to the adoption of a proxy-based VAN scenario, where network resources are handled by proxies which are also endowed with caching and pre-fetching mechanisms that permit to decouple the service in two segment: the user asks for a service to the proxy that, upon having collected any user requests, asks for data to the remote server, through a wireless access network. Additionally, the proxy is in charge of enforcing an elastic packet buffering before delivering data to users: the effects of the elastic buffer, as explained in what follows, go well beyond the simple delay smoothing envisioned in traditional schemes.

The chapter is organized as follows. In section §5.1 we introduce to vehicular area network issues and to our proxy-based networking proposal. Then, in section §5.2 a presentation of service to be supported over VANs is given, together with the introduction of a new bandwidth management scheme to be adopted by proxies located in the VANs. The third part of the chapter (section §5.3) explains how we tested the proposed approach, in a simulative fashion, while section §5.4 is devoted to a thorough performance evaluation of the proposed approach. Eventually, conclusions are given in section §5.5

§5.1 VAN networking

Mobile networking not only tackles the problem of providing a networking infrastructure for mobile customers, but also includes the ability of managing moving networks. This is the case of Vehicular Area Networks (VANs). These networks are formed by customers located on the same moving vehicle, e.g. a moving train or bus. The moving network is interconnected to the terrestrial network via one or more wireless links. Satellite links are frequently considered as an important candidate technology in practical deployment scenarios (see for example [2], [26], [44], [79], and [104], encompassing public trains, ships, airplanes, buses, etc. But emerging wireless technologies for metropolitan high speed networks, such as IEEE 802.16 [53], [56] and IEEE 802.20 [55], may become in the near future important technologies in the vehicular area networking arena.

Consider that wireless technologies and multimedia world tend to be integrated in a worldwide mobile telecommunication environment, in which mobile customers want to experience multimedia service even while moving or traveling. On the other side, it is possible to recognize that a greater and greater number of entertainment systems are based on multimedia and network content retrieval. Thus it can be easy to foresee the interest of entities like railway operators, or air companies, in the development of effective mobile multimedia servers, able to entertain passengers while reaching their destinations.

A suitable architectural solution is proposed here, consisting in a satellite content server that relays files to a local proxy, which in turn is responsible of storing data and forwarding the requested files to the right customer. The scenario considered is that of a VAN in which users connectivity to the rest of the network is managed by a specialized on-board gateway. The role of this gateway is to provide internetworking between the internal network and the wireless technology adopted for connecting to the terrestrial network. For sake of simplicity, in this paper we assume that such a connection is accomplished through a single high-capacity wireless link (e.g., a satellite link or a WiMax [102] connection), hereafter referred to as wireless backbone. The network architecture deployed within the VAN can be built either on wireless or wired local area networking technologies, its implementation details being out of the service support issues addressed in this thesis, and depending on the considered vehicular network application scenario. In addition to internetworking functions, the gateway may act as proxy server. As such, it may provide local storage capabilities to support caching and/or pre-fetching algorithms. These mechanisms are meant to support video and interactive stream [85], [90], as well as multimedia with quality of service [86], [98]. These mechanisms are also devised to maximize the probability that a customer requesting an object to download (e.g., a web page or a video segment) may find it stored into a repository associated to the proxy, thus improving the retrieval performance, and reducing the traffic load on the wireless backbone. Effective caching and pre-fetching mechanisms for streaming services, which may be applied with minimal or null modification to the VAN scenario considered in this work, have been thoroughly studied in [30], [99] and [66], also considering pre-fetching located on the edge of the backbone network, in edge routers, where micro-cell or overlying network are present, as in [61].

A problem typical of the VAN scenario is the possible outage of the wireless backbone. Such an outage, hereafter referred to as “channel outage”, may occur while the vehicle crosses areas characterized by severe fading conditions, as in the case of going through tunnels. This is a very critical issue when dealing with streaming services, which experience possibly long (order of several seconds) interruptions, thus causing highly negative performance impairments (service disruption) in terms of the end customer point of view. In what follows we well refer to the event of service interruption as “connection outage”. The proxy solution is also suitable in order to avoid the end user dealing with handover issues, since the mobile proxy could be made able to take care of eventually needed handover (this is still not true in the case of satellite, but in general it could be true if one uses wireless MAN connectivity standards [53], [54], [102]); for example proxy could implement techniques devised in [92], where pre-fetching issues are also addressed. Now, we argue that, other than traditional caching and pre fetching mechanisms, a further role of the proxy is to support mechanisms devised to hide channel outage periods to the final user, thus reducing the impact of channel outages in terms of resulting connection outage. This can be accomplished by decoupling, from a service level point of view, the delivery service provided inside the moving network, from that enforced on the wireless backbone. Specifically, the presence of an on-board proxy enables the network manager to split the streaming service into two independent parts. Within the moving network, the stream is delivered to the end user at its natural playout speed. For convenience of presentation, in what follows, we well assume a constant playout rate[27]. Conversely, the multimedia information may be retrieved on the wireless backbone at a variable speed, eventually higher than the playout rate: the resulting excess information retrieved is accumulated into a proxy buffer for future playout. Hence, the per-stream proxy buffer can be seen as an "elastic" buffer that empties at constant rate and is filled at variable rate: although during a channel outage no information can be retrieved, this does not automatically lead to connection outage, as long as sufficient information has been previously stored in the buffer.

§5.1.1 Proxies and elastic buffers

A number of works dealing with file download issues can be found in the specialized literature. Basically the problem can be separated into two parts: the retrieval of a file and its physical download. The former problem is related to content network architectures deployment in terms of content discovery algorithms, and it is out of the scope of this work, while the latter part is related to capacity allocation issues, local storage and replicas management and hardware performance like memory access speed. Since the aim of the present work is the minimization of connection outage probability while downloading a multimedia stream, this work deals with local storage issues and optimized server-proxy connections.

Proxy is a well know approach, meant to mitigate problems like high packet loss rates and long delays, as shown in [66], where different caching and pre-fetching algorithms are tested. These techniques consist in caching on the proxy data that are frequently required, even if there is not any user actually requesting those data. The pre-fetching approach allows a higher

[27] In fact, the basic ideas proposed here do not require a constant playout rate. However, the detailed design of the resource management algorithm on the wireless backbone may slightly differ in the case of variable rate streaming services, as it might be optimized to further take advantage from the knowledge of the traffic flow statistical description.

bandwidth utilization efficiency. In fact, pre-fetched data can be cached only once in correspondence with many download requests. Moreover different caching and pre-fetching strategies has been proposed in order to optimize other performance parameters as: network bandwidth utilization [99] and delay playout, as in [90] where only a file prefix is pre-fetched on the proxy memory; in [86] and [72] layered-encoded multimedia are also considered in pre-fetching algorithms, where incremental layers can be timely stored on the proxy; the hit ratio, i.e., the probability that a requested file matches an object stored on the local proxy, is the aim of the work presented by Reissline et al. in [85]. The role of a proxy-assisted delivery schemes is focused by B. Wang et al. in [98], where the authors show how a relative small amount of data is needed to be pre-fetched in order to experience very low start-up delays with relative low network costs. Finally an insight in cost-based pre-fetching replacement schemes can be found in [106] and [97]. We face the proxy and pre-fetching problem from a different point of view, that is from the perspective of QoS in terms of connection outage probability. The main contribution of this chapter consists in the definition of a new bandwidth allocation algorithm for the wireless backbone link. The proxy operates a dynamical allocation of the satellite bandwidth in order to maintain a certain minimum buffer level for each active connection. In fact we argue that the proxy is able to minimize the connection outage probability by simply control the buffer size associated to each connection.

Moreover, a well known solution to speed-up any client download, is the connection splitting between the content server and the client, by interposing a proxy node able to quickly download files from server and also acting as a speed-buffer for the client. In fact, client applications typically need a reduced amount of bandwidth with respect to the server capacity. Splitting the download connection, using elastic buffering strategies on a proxy, allows network provider to better use available resources, decoupling the transmission speed [12] and it also prevents the client applications to be stopped due to a temporary server outage.

Elastic buffering is a proxy management technique devised to decouple the multimedia information retrieval rate on the network backbone from the playout streaming rate at the user terminal. It has been shown in the past that the application of elastic buffering mechanisms in terrestrial networks brings significant advantages in terms of network effectiveness [13]. Here we show that elastic buffering is an extremely effective mean to reduce, or even eliminate, streaming service outages due to intermittent backbone connectivity. Moreover, we show that elastic buffering is not only a technique suitable for multimedia information retrieval services, but it can be effectively applied to artificially delayed real time services.

Before tackling the issue of moving networks, let us first review the fundamental performance advantages provided by the adoption of elastic buffering mechanisms in the support of streaming services. This theme is quite new, and it was firstly proposed in our early works [12] and in diffusive terms in [71]. The basic idea that we propose is very simple. Without proxy, a multimedia streaming session is setup directly between the end user and a remote streaming server. In the assumption of limited buffering capabilities at the receiver, the delivery rate of the multimedia information is of course equal to its playout rate at the receiver. Let's now insert, in this delivery process, an intermediate dynamic storage capability, i.e., a buffer, placed on a proxy between the receiver and the video server. It is thus possible to split the streaming connection in two parts. From the proxy to the end user, the multimedia information will be delivered at its natural playout speed. But from the server to the proxy, it is possible to retrieve the multimedia information at a rate higher than the playout rate. The excess information downloaded from the satellite is in fact temporarily buffered in the storage resources available at the proxy. This operation leads to a significant improvement in terms of network utilization. As shown in Section V of [12], such a performance advantage can be analytically quantified in the assumption of infinite buffer size, and by considering a simplified network scenario composed of a single network link, in what follows named backbone, connecting the proxy to the server. In fact, let b bit per second be the playout speed of the multimedia streaming sessions (assumed homogeneous). Let B bit per second be the backbone capacity. For convenience, let B be multiple of b. Hence $K=B/b$ represents the maximum number of simultaneous streaming sessions that can be supported on the backbone. Assume Poisson arrival process of the streaming session requests, with rate λ requests/s, and let $E[T]$ be the mean playout duration of the multimedia streams (what follows hold for general distribution of the stream duration, depending only on the mean value $E[T]$]). Let $r = \lambda E[T]$. Then (in stable conditions, i.e., $r < K$):

- without proxy buffer, the number of simultaneous streams carried on the backbone obviously follows a Poisson distribution, i.e. the probability of finding n streams on the backbone is:

$$(21) \qquad P(n) = e^{-r}\frac{r^n}{n!}$$

- with the intermediate proxy buffer, the backbone can instead be modeled as an M/G/1-PS processor sharing queuing system, where the probability of finding *n* streams on the backbone is:

$$\textit{(22)} \qquad P(n) = \left(1 - \frac{r}{K}\right)\left(\frac{r}{K}\right)^{n}$$

In [12] a comparison between these two cases is given in terms of backbone blocking performance[28], assuming that an admission control rule is deployed on the backbone so that no more than *K* concurrent sessions can be simultaneously admitted. Without proxy buffer, the request blocking probability is readily computed via the Erlang-B formula *B(r,K)*. With proxy buffer, the blocking probability is computed as the drop ratio of an M/M/1/K finite size queuing system:

$$\textit{(23)} \qquad P_b = \left(1 - \frac{r}{K}\right)\frac{\left(\frac{r}{K}\right)^{K}}{1 - \left(\frac{r}{K}\right)^{K+1}}$$

The elastic buffer solution outperforms the standard streaming approach (for a numerical example, the reader may easily compute that, when $r = 85$ and $K = 100$, the blocking probability experienced by the users reduces from about 10^{-2} to about 10^{-8} in the case of elastic buffer. Simulation results obtained with finite buffer size [13] show that significant performance advantages can be obtained even with fairly limited storage availability. Finally, more efficient buffer management policies, integrating elastic buffering with traditional caching, are presented in [12].

§5.2 Services

To date, vehicular network users suffer for a very limited access to network resources. They can use local wireless connectivity provided by GPRS, EDGE and UMTS, and they could experience highly variable link conditions, also relative to the speed of the vehicle, and

[28] We remark that the discussion carried out in that paper was focused on an ATM transport network scenario, being such a technology extremely actual at the moment of writing that paper. In the ATM framework, it was natural to deploy the proposed approach on the Available Bit Rate (ABR) service, set the Minimum Cell Rate (MCR) to the playout rate of the stream, and evacuate the performance in terms of connection blocking. The performance insights given in this section suggest to push towards the deployment of elastic buffering mechanisms also for streaming service support over the Internet, by using TCP on the network core rather than constant-rate streaming over RTP/UDP. Of course, unlike the case of ATM, an admission control rule suitable for this scenario and, therefore, acting on TCP flows is harder to deploy.

to the number of connected users. However, the bandwidth available can range from few units to some hundreds of *Kbps*, and it is often not enough to support application such as video streaming. Even though new technologies provided with higher bandwidth will be soon available, and UMTS systems will be more efficiently deployed, the adoption of a bandwidth sharing strategy remains the mostly important issue to be addressed. In fact, network providers aim at furnishing video telephony, video and multimedia on demand, real time applications and, in general, a set of bandwidth consuming products. Additionally, the offered service will include traditional Internet applications such as e-mail, file transfer and web browsing, plus new location services to support vehicular mobility and to entertain users with information about local attractions and opportunities.

The majority of bandwidth consuming applications can be classified into Multimedia-on-Demand (MoD) and Real Time (RT) applications. We can further distinguish inside MoD and RT group. As regards MoDs, they include various types of streaming services, like video and audio, and the mostly employed MoD application is Video-on-Demand (VoD). As to real time services, they are commonly thought as direct streams of audio and video associated to an event to be broadcast. We also include in the RT group those applications that introduce an artificial playout delay before sending the stream or before playing it. These are the so called Delayed Real Time (DRT) services. The adoption of a playout delay has implication on the reliability of the application. In fact, the gap between the real time and the playing time allows the system to recover some transmission errors and, mostly common, to compensate the delay jitter experienced by data packets. It is worth noting that real time applications, and a number of video streaming applications, generally adopt UDP transport protocols, or another connectionless transport protocol. Thus no segment retransmissions occur after a delivery failure or upon a not recoverable error in the bit stream. This approach works fine when the failure rate is very low and no consecutive losses are experienced by the receiver. Unfortunately, in a VAN scenario the failure probability is highly variable and it can assume relevant values. In particular, VAN connectivity tends to behave as an *On/Off* process, where *On* and *Off* periods are strictly related to the location of the vehicle. Thus we will show that it makes sense to enforce some elastic buffering mechanisms also for real time application, and adopt DRT services instead of normal RT streaming. The artificial gap introduced in real time transmission should be long enough to allow users to asks for and obtain the retransmission of long sequences of lost data, e.g., after having gone through a tunnel.

In following subsections (§5.2.1 and §5.2.2) we describe the requisite of services similar to Video-on-Demand and Delayed Real Time, while in subsections §5.2.3 and §5.2.4 we will present the solution we envision for the management of concurrent VoD and DRT applications in the same VAN.

§5.2.1 VoD Services

Since Multimedia-on-Demand services are easily illustrated by the Video-on-Demand services subset, in the following we will refer to the behavior of VoD-like services. The basic idea of VoD functionalities is very simple. Each active client has a dedicated buffer on-board of proxy, in which client data are stored. The goal is to make the system robust to the uncertain behavior of the server-to-proxy link. In fact, the satellite-to-proxy link is characterized by a wideband, high-performing channel, but it requires a line-of-sight (LOS) connection. When no line-of-sight is available, the proxy stops receiving data from server, while clients connected to the proxy continue to receive data until their buffer, located on the proxy, goes empty too. Similar considerations apply if the satellite link is replaced with a different wireless access network; in that case, a more generic shadowing effect has to be considered instead of lack of LOS. The key of our approach consists in a suitable algorithm, namely A2M, for the allocation and sharing of the available bandwidth in the link between server and proxy. In this scenario, the file transfer is split into two sub-connections:

- the server to the proxy, where files can be relayed with a high rate, but the actual connection rate is an *On/Off* function of wireless backbone connectivity[29];
- the proxy to the client, where the file transfer occurs with tangible rate reduction, but the link (either radio or wired) shows very high performance.

Data pertaining to a VoD flow are transferred within a sequence order, and as a consequence of packet losses, application and/or transport protocols use retransmission techniques. The adopted mechanism is similar to TCP, where out-of-order packets can be received within a specific transmission window; this means that the occurrence of long term outages implies a pause in the transmission of new segments while lost data are retransmitted. In our study, we consider an application layer control that enforces the retransmission of lost data before going on with the file transfer, even in the case of not reactive and connectionless

[29] This consideration especially applies when satellite access networks come into play, while for terrestrial wireless connectivity, the bandwidth available in the access network can be at least modeled as a process with a finite number of states ranging from zero to the maximum supported bandwidth of the system.

transport protocols (e.g., UDP). In conclusion, a very important role is played by the strategy adopted to transfer the requested files from the content server to the proxy, and by the value added services offered by the proxy, which can be able to retain in its cache memory a certain amount of highly requested data (on accordance with the an empirical computed hit ratio).

§5.2.2 DRT Services

The Delayed Real Time services functionalities are more complex then the VoD functionalities because in real time streaming any piece of data is strictly connected with the time in which it will be delivered to users. In what follows, the locution "Real Time Stream" is referred to a multimedia content that is delivered to users in an exact time. Moreover, real time data considered here are delayed before to be delivered, so that we talk about Delayed Real Time services. It is similar to recording, but the content will be delivered exactly in a date and not when users ask for it. Nonetheless, users can look for real time transmission and are made able to join an ongoing broadcast delivery. It works exactly as in the case of TV broadcast channels, that users can select but cannot control. The difference is that users receive data after an additional delay, and they can ask for selected retransmission within a limited time interval.

As in the case of VoD flows, each active Real Time content (named "channel") has a dedicated buffer on-board of proxy. This buffer will store data when the proxy receives it; the data delivery takes place with a fixed delay with respect to the real time broadcast (the so-called playout delay). When the proxy is unable to receive data it will leave a blank space and continue to receive following data. In fact, original data are broadcast, and they are out of the user control. When a consistent packet loss occurs, the proxy is made able to ask for particular retransmissions, on behalf of users. The recovery procedure has to be completed within an interval time not greater than the playout delay, that is the time at which delayed data are delivered to the user. In other words, the system can receive a packet within a playout interval after the original transmission, otherwise the packet id definitely lost. Thus the recovery procedure can last, at most, as much as the artificial playout delay, and if the system does not add any delay, no packet loss will be recovered. This kind of application requires the logical division of available bandwidth in two parts:

- a fixed rate will be statically assigned to the retrieval of the latest data (i.e., the broadcast);
- the remaining bandwidth will be used to fill windows of not received data.

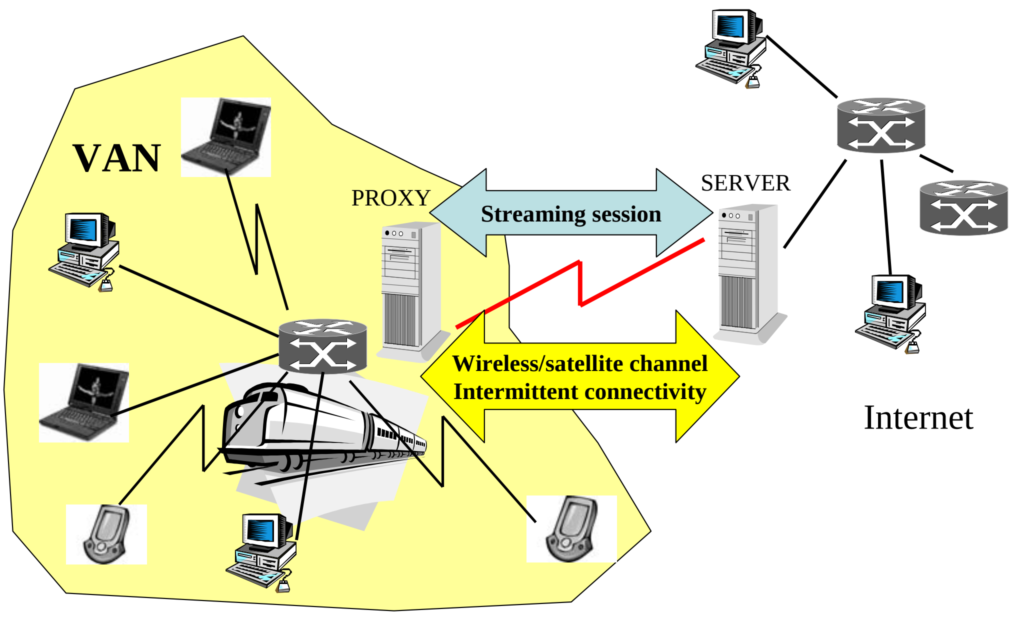


figure 66 - Vehicular Area Network scenario

Studies will be focused on the definition of an optimal restoring algorithm and on the definition of analytic criteria to decide the number of supported channels and the length of the buffer associated to channels.

§5.2.3 Adaptation of elastic buffering to moving networks

In addition to the role, discussed in the previous section, of improving the network performance, in the rest of the section we show that, in a moving network scenario, elastic buffering brings additional significant advantages. In figure 66 it is illustrated the concept of Vehicular Area Network (VAN). In traditional wireless networks, each user sitting on a moving vehicle connects independently to the terrestrial network. Conversely, in a VAN scenario, a network - typically a LAN or Wireless LAN - is deployed inside a moving vehicle. End-user connectivity to the terrestrial network (i.e., the Internet) is managed through a specialized on-board gateway that we propose to endow proxy functionalities. In what follows, we illustrate how a VAN can take advantage of an elastic buffer mechanism implemented on the on-board proxy. As long as some minimum requirements are met, the described ideas do not specifically depend on the networking technology employed inside the VAN, on the wireless backbone technology, and on the applications scenario. However, to provide a more concrete presentation, we will expose our basic ideas with reference to a particular application scenario, namely a train network connected to the terrestrial network through a satellite link. The goal of our approach is to make the system robust to the uncertain behavior of the satellite link. In fact the satellite link is characterized by a wideband, high-performing channel, but it requires a LOS connection. During a satellite link outage, which may last for several seconds, all communications are interrupted since no information can be received at the on-board gateway and, in turn, routed toward the relevant end user[30].

Let's first focus on VoD services (the extension to DRT services will be tackled in the next section §5.2.4). By enforcing an elastic buffer management in the on-board proxy, multimedia streaming sessions may remain active even during satellite outage periods, provided that a sufficient amount of information has been buffered in the proxy local storage. The streaming session presents outage (*connection* outage) only when both the satellite link is in outage and the buffered information exhausts. We well show in the following subsection that a very important role is played by the strategy adopted in managing the satellite link capacity. Specifically, a uniform allocation of the extra available bandwidth to the concurrent downloads, while effective in the case of wired networks results to be a poor strategy in the VAN scenario (as reviewed in section §5.4).

§5.2.3.1 The All-To-Min resource sharing algorithm

The described operation gives raise to an elastic buffer, which is filled during the periods in which the satellite link is active, and whose buffered data is consumed at a fixed rate given by the playout speed for each streaming session, multiplied by the number of concurrent streaming sessions. For the sake of simplicity, assume that the multimedia information repository is directly placed at the terrestrial satellite gateway. Let us consider the eventuality that an on-board customer requests a file. Once the streaming request is received at the proxy, it firstly checks if the requested stream is locally cached. If this is not the case, the request is forwarded to the remote server, provided that satellite resources are available (i.e., after a positive response of an admission control decision based on the number of already active streaming sessions on the satellite link). All the satellite link capacity is shared between all active sessions so that the multimedia information is retrieved on the satellite link at the maximum possible rate. The figure 67 describes two different resource sharing policies that can be enforced on the satellite link. The simplest approach is to fairly share the satellite resources among all the concurrent multimedia information retrievals. This policy is referred

[30] Indeed, this is the application scenario tackled in an European Community funded IST project, called FIFTH - Fast Internet for Fast Train Hosts. Other VAN application scenarios are nowadays being considered, either in research projects (e.g., ships are tackled in the IST project MOBILITY) or in commercial deployment (airplanes and buses). But the case of trains and buses is the one that presents major challenges at the data link and network layer, due to the occurrence of tunnel crossing - which is not applicable in the cases of ships and airplanes.

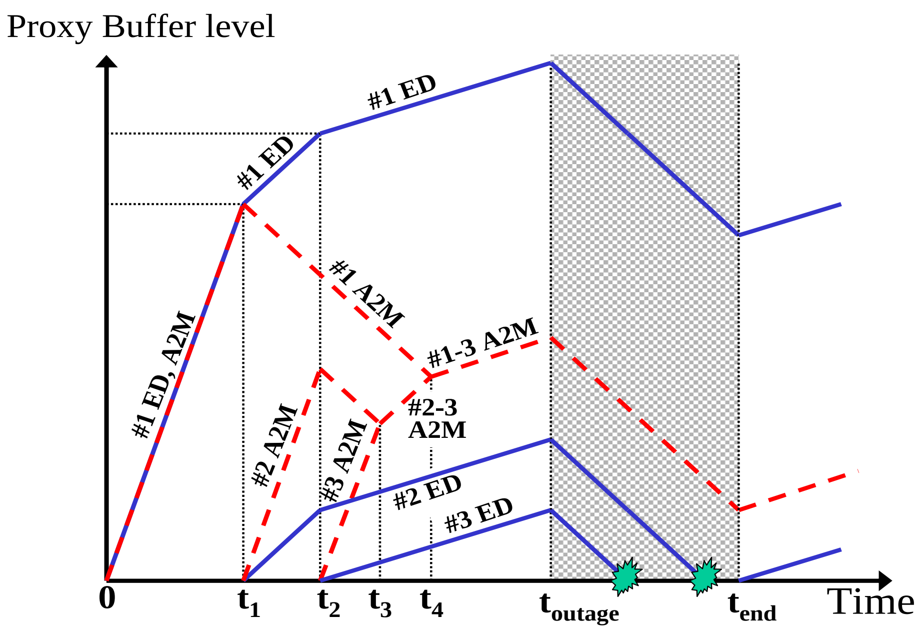


figure 67 - A2M vs. ED Operation

to as "Equally-Distributed" (ED). figure 67 illustrates the ED operation (solid lines) by reporting the per-session proxy buffer occupancy level versus time. It is assumed that at time 0, a new connection, session *#1*, starts. Let *C* be the satellite channel capacity, and let *R* the streaming playout rate (supposed equal for all sessions). Since a single session is offered to the satellite link, the channel capacity *C* is reserved to download information. This information is, in turn, played-out at rate *R*. We conclude that the buffer-level grows linearly with time, being

$$(24) \quad B_1(t_1)=(C-R)t_1 \ [bits]$$

the buffer-level at time t_1. Assume now that at time t_1 session *#2* starts. At time t_2 the buffered data for this latter session will amount to

$$(25) \quad B_2(t_2)=(C/2-R)(t_2-t_1) \ [bits]$$

while the level of the buffer occupation for session *#1* will have reached

$$(26) \quad B_1(t_2)=(C-R)t_1+(C/2-R)(t_2-t_1) \ [bits]$$

In general, when *n* session share the link, each is granted a retrieval rate equal to *C/n*. If no channel outage is assumed to occur, a reasonable admission control rule is to admit a new session as long as *C/n* remains greater or equal than *R* (ED is in fact the policy assumed through the discussion carried out in section §5.1, and this was the related admission control rule). A closer look to figure 67 allows us to underline a major limit of the ED policy. In fact, whenever a channel outage occurs (in the figure, from time t_{outage} to time t_{end}), the most recently admitted sessions are the first ones for which connection outage occurs. In turn, the buffer level reached by the first admitted flow is unnecessarily high. In other words, a fair resource sharing on the channel yields an unfair share of the proxy buffer space, and increases the probability that connection outage occurs during a channel outage period. In the case of channel outage, the buffer level represents the margin before an outage occurs. For this reason we will use also the term "outage margin". More generically, the outage margin represents the time before an outage occurs if connectivity goes down. It naturally comes out that an effective resource sharing approach is to devise a policy targeted to converge as fast as possible to a situation in which all sessions have the same outage margin.

We refer to such a new policy with the name "All to Minimum" (A2M). The A2M operation is designed to dynamically reserve all the channel resources to the session(s) with lower buffer level. This operation is implemented at the proxy, which can access the information regarding the per-flow buffered data. In turn, the proxy dynamically signals (through Layer 2 signaling mechanisms - e.g., satellite-specific mechanisms) to the terrestrial gateway the identity of the sessions which suffer of minimum buffer level, so that this latter gateway is able to properly schedule the information download. The A2M operation is exemplified in figure 67 (dashed lines). At time 0, the A2M algorithm operates as the ED one, since a single session is admitted on the channel. The differences start from the instant of time t_1 in which session *#2* starts. In fact, from time t_2, all the channel capacity is reserved to session *#2*. Hence, its buffer-level grows at rate C-R. Conversely, since no channel resources are assigned to session *#1*, its buffer level decreases at rate R. This would last until the two buffer levels would become the same. If, in the meantime, a new session *#3* starts (time t_2 in the figure), it will receive all the channel resources until it reaches the same buffer level of session *#2* (this occurs at time t_3 in the figure). Then, the channel capacity C will be equally shared between sessions *#2* and *#3*, so that their buffer level grows with rate $C/2$-R while the buffer level of session *#1* still decreases with rate R. Once all the n (three in the example) per-session buffered data have reached the same level (time t_4), a uniform share C/n of the channel capacity will be again enforced. As apparent from the figure, the A2M mechanism minimizes the probability that a connection outage occurs during a channel outage period. The price to pay is that, in the case a connection outage occurs, this is likely to simultaneously involve all the admitted sessions.

The A2M algorithm can be easily extended to a framework where streaming sessions are characterized by different data rates. The only difference is that the per-session buffer level has to be measured in terms of playout time, i.e., in terms of time to reproduce a content on a user terminal. The following Figure 68 and figure 69 depict the flow chard diagrams that explain ED and A2M operation upon a connection request arrives. In table 8 the two algorithm are reported in pseudo-code.

ED	A2M
1 - client requests a file to proxy; 2 - proxy checks if file is pre-fetched; 3 - if(file not pre-fetched) 3a - proxy requests the file to the server; 3b - *n* = # active connections 3c - proxy starts receiving file from server at rate *C/n* && proxy starts the delivery phase at rate *R*; else 3d - proxy starts the delivery phase; 4 - if(*n* changes to *n'*) 4a – set all download rate to C/*n'*	1 - client requests a file to proxy; 2 - proxy checks if file is pre-fetched; 3 - if(file not pre-fetched) 3a - proxy asks for the file to the server; 3b - proxy starts receiving file from server at a full rate *C*; else 3c - proxy starts the delivery phase; 4 - while (file not completely on proxy): 4a - proxy searches stream(s) with lower outage margin; 4b - all bandwidth resources assigned to stream(s) with lower outage margin;

table 8 – ED and A2M algorithms

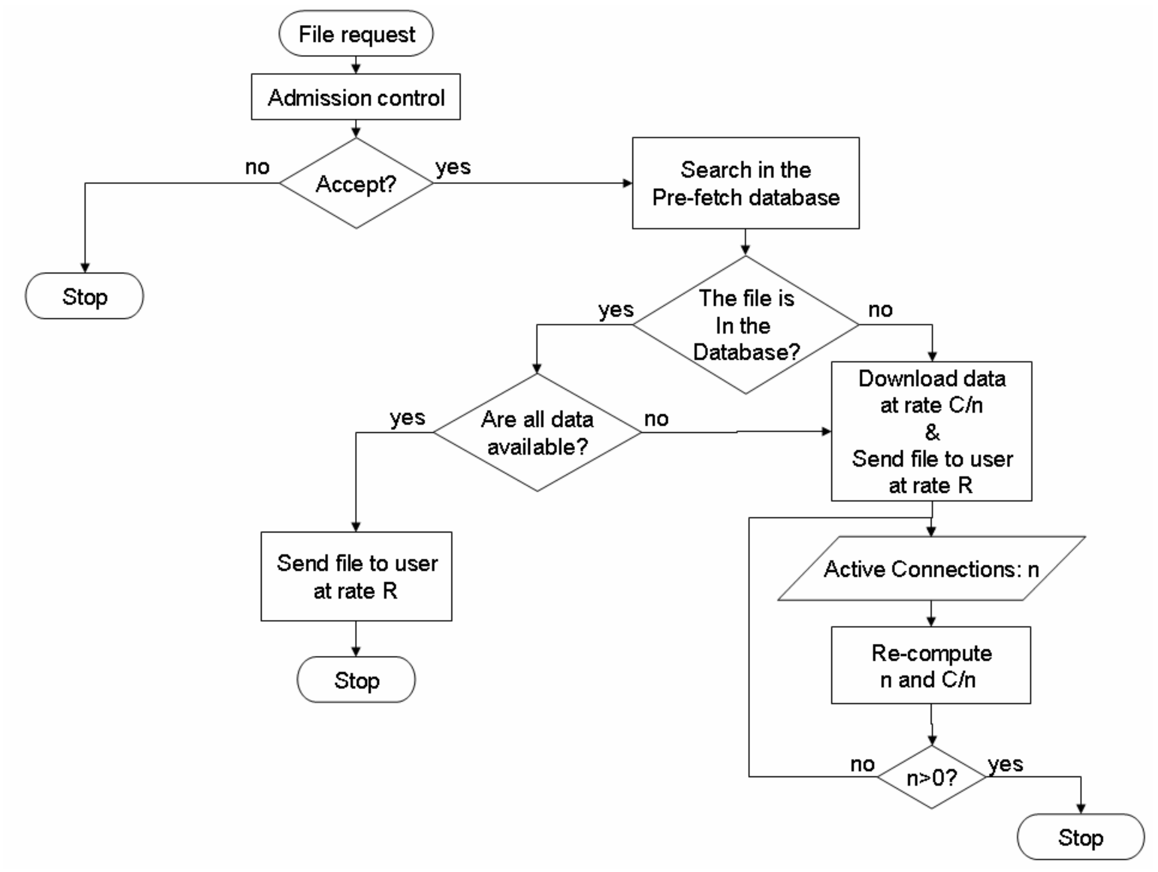

File request
Admission control
Accept?
no
yes
Stop
Search in the Pre-fetch database
The file is In the Database?
yes
no
Are all data available?
yes
no
Download data at rate C/n & Send file to user at rate R
Send file to user at rate R
Stop
Active Connections: n
Re-compute n and C/n
n>0?
no
yes
Stop

Figure 68 – Simplified ED flow chart diagram

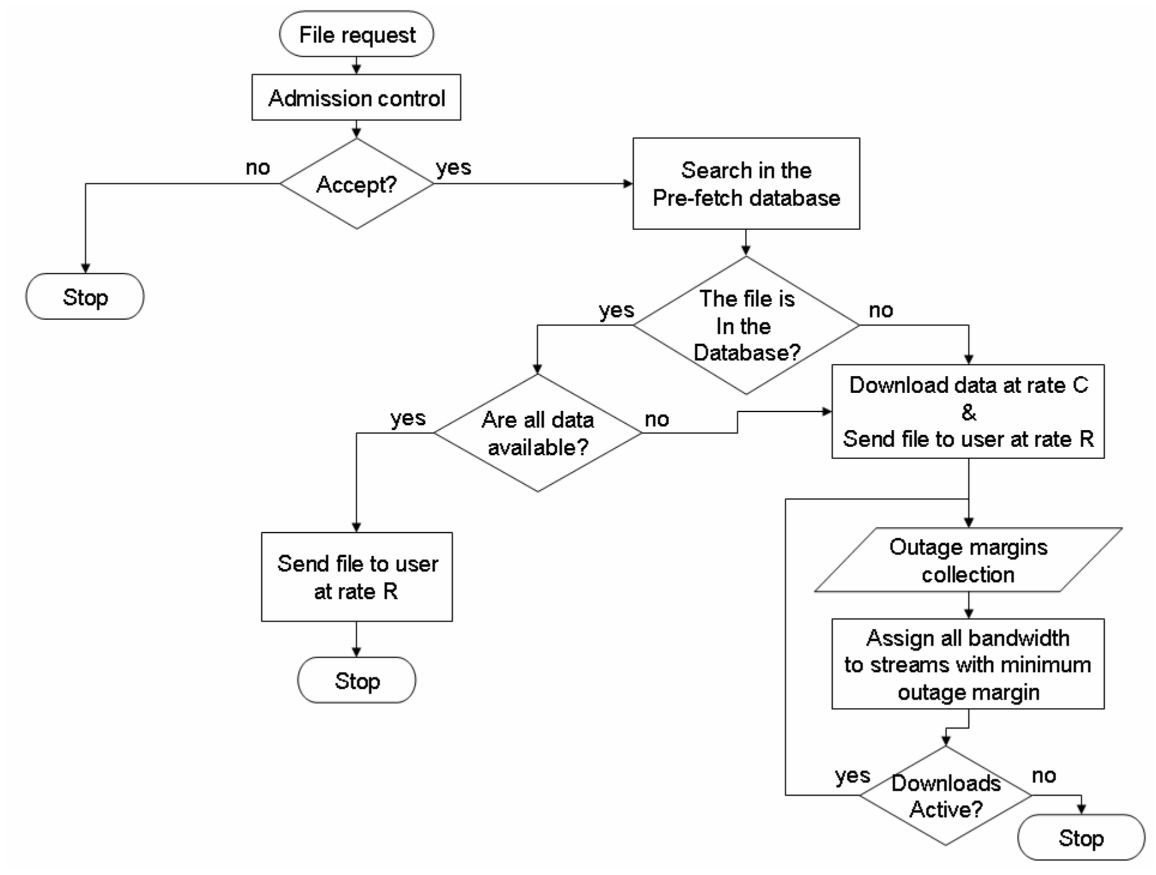

File request
Admission control
Accept?
no
yes
Stop
Search in the Pre-fetch database
The file is In the Database?
yes
no
Are all data available?
yes
no
Download data at rate C & Send file to user at rate R
Send file to user at rate R
Stop
Outage margins collection
Assign all bandwidth to streams with minimum outage margin
Downloads Active?
yes
no
Stop

figure 69 – Simplified A2M flow chart datagram

§5.2.4 Adaptation to Delayed Real Time streaming services

VoD and MoD services are hardly supported in a scenario characterized by a very large number of customers, such as the train passengers, and a fairly limited wireless backbone bandwidth, such as the satellite link. In fact, the satellite link may support only a limited amount of simultaneous streams, and thus only very few customers may receive service. Conversely, the seamless support of broadcast streaming services (such as digital television channels) on moving networks is an extremely appealing challenge for a public vehicle operator. Of course, due to the presence of tunnels, it is possible to compensate channel outage for real time streams only if their delivery inside the train is delayed of a time greater than the crossing time of the longest tunnel along the train path. In other words, if an event is scheduled to be broadcast at a given time t, the on-board delivery will start at time $t+D$, and the proxy will buffer all the information related to the D seconds of delay introduced. Delayed playout can be accomplished by simply using fixed buffers whose size is exactly equal to the one needed to store D seconds of broadcast video. Of course, when an outage lasting $\delta>D$ seconds occurs on the satellite link, the proxy will be able to playout only the first D

seconds of the considered stream, while connection outage will occur for the remaining δ-D seconds. However, a closer look to this problem reveals that an initial playout delay greater than the crossing time of the longest tunnel along the train path is not sufficient by itself to avoid outage. This is evident considering the example depicted in figure 70. This figure reports the real time streaming as received from the satellite (solid line) and the DRT streaming delivery occurring on-board (dashed line). The x-axis reports the elapsed time. As shown in the figure, the delayed playout starts with an initial offset equal to D. Two channel outage periods are illustrated in the figure: one occurring in the range [t_1, t_2] and the other in the range [t_3, t_4]. If no action is taken, it is evident that, during an outage period, a fraction of the original stream (referred to as sub-stream *#1* and *#2* in the figure) will not be stored in the proxy buffer. Hence, after a delay D, this outage will be experienced also within the train, and specifically in the time ranges [D+t_1, D+t_2] and [D+t_3, D+t_4]. In other words, the difference with respect to the multimedia information retrieval scenario stays in the fact that, during a channel outage, the satellite broadcasting goes on and hence the missing (not received) sub-streams cannot be temporarily stored in the buffer for subsequent playout. In this case, the outage margin defined for VoD-like services can be re-defined as the length of the first continuous block available in the buffer, i.e., the time distance from the first "hole" to be filled.

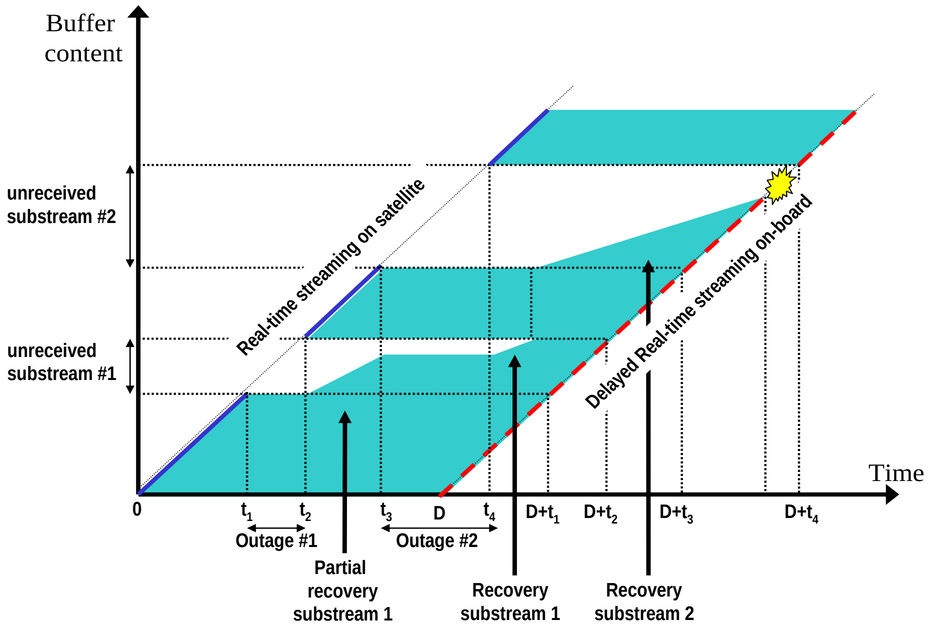


figure 70 - Connection outage after multiple channel outages

We propose to solve the above problem by treating each sub-stream as an independent special case of multimedia information retrieval session. As soon as an outage period ends, the proxy buffer uses extra bandwidth available on the satellite channel to recover the sub-stream, and thus refill the buffer “hole” caused by the channel outage. This implies that, to effectively support DRT services, the channel capacity must be split in two parts: one used to transmit the real time streams, and the other used for the described sub-stream recovery procedure. It will be shown in the numerical results section that the extra bandwidth made available to DRT services must be quite large. Since, in addition, this extra bandwidth is used only at the end of a channel outage, and generally for a short time (i.e., in a very bursty mode), it is convenient to let the sub-stream recovery procedures share the same bandwidth reserved to Video/Multimedia-on-Demand streaming sessions. In doing this, it is worth noting that the A2M algorithm can be adopted to manage this extra bandwidth, i.e., no distinction is in principle needed between sub-stream recovery procedures and normal on-demand multimedia streaming support.

To better understand how connection outage may arise, consider again the example illustrated in figure 70. We have here reported a scenario in which recovery of multiple not received sub-streams may occur. In fact, the recovery phase for sub-stream *#1* starts at time t_2, right after the end of the relevant outage period. However, due to scarce extra bandwidth availability in the time interval $[t_2, t_3]$, and/or too short time range $[t_2, t_3]$ elapsing before outage *#2* occurs, it is possible that only a part of sub-stream *#1* is recovered. As illustrated in the figure, at time t_4, the recovery procedure for the sub-stream *#1* restarts (in fact, following the A2M rules defined in section §5.2.3.1, the outage margin for the sub-stream *#1* is lower than the margin for sub-stream *#2*). Hence, all the available bandwidth is assigned to sub-stream *#1* until its complete recovery. But, in turn, this delays the start for the recovery procedure of sub-stream *#2*. In other words, even if the outage margin after channel outage *#2* was large enough to allow full recovery of the sub-stream *#2*, the presence of an additional sub-stream to be recovered leads to connection outage.

An important parameter is the ratio K between the spare bandwidth made available for recovery and the delayed streaming service rate. This parameter can be intended as the amount of video seconds that will be recovered in a second. In practice, when $K>1$, connection outage can occur only when the train is in a tunnel. But $K>1$ means that more than 50% of total bandwidth is allotted to recovery operations; this is a poor efficiency in link utilization. When $K<1$, the time needed to recover a sub-stream is longer than the sub-stream itself; this means that it cannot be recovered while delivering it, and an outage occur. Suppose *K<1*, and let *M(t_x)* be the outage margin at time t_x, $T_{s1}(t_x)$ the duration of the first sub-stream to be recovered at the same instant, and *R* the normal download rate[31]. Then, the given sub-stream can be recovered if and only if the following conditions are satisfied:

- the outer network is reachable in the time interval

$$\textit{(27)} \qquad t \in I(t_x) \equiv \left[t_x ; t_x + \frac{T_{s1}(t_x)}{K} \right];$$

- the margin limit remains greater or equal to zero in *I(t_x):*

$$\textit{(28)} \qquad M(t_x) + (K-1) \cdot R \cdot (t - t_x) \geq 0 \quad \forall t \in I(t_x)$$

This yields to the following equivalent condition:

$$\textit{(29)} \qquad \min_{t \in I(t_x)} \left[M(t_x) + (K-1) \cdot R \cdot (t - t_x) \right] \geq 0$$

$$\textit{(30)} \qquad \Rightarrow M(t_x) + (K-1) \cdot R \cdot \frac{T_{s1}(t_x)}{K} \geq 0$$

Since our aim is to define an admission control rule, we are interested in the minimum value of *K* that as to be provided in the interval *I(t_x)*:

$$\textit{(31)} \qquad \frac{1}{1 + \dfrac{M(t_x)}{R \cdot T_{s1}(t_x)}} \leq K < 1$$

Under these conditions, the system assures the recovery of the first sub-stream. If multiple sub-stream need to be recovered, the procedure can be iterated over each time interval that relates to a single sub-stream, considering that each sub-stream is characterized by a potentially different rate R_i (since a different number of transmission can be defeated in each different outage period), and assuming as starting time

$$\textit{(32)} \qquad t_i = t_{i-1} + \frac{T_{si-1}(t_{i-1})}{K(I(t_{i-1}))},$$

where *K(I(t_{i-1}))* is the value of *K* adopted in the previous time interval, and as outage margin

$$\textit{(33)} \qquad M(t_i) = M(t_{i-1}) + \frac{K-1}{K} \cdot R_{i-1} \cdot T_{si-1} + T_r$$

[31] I.e., for sake of simplicity, we suppose a constant playout rate, *R*.

where the last term T_r, is the time elapsed between outage *#(i-1)* and outage *#i*, thus representing normally received data from the broadcast channel.

§5.3 Simulative approach

A great effort was carried out, in order to solve the problem of coverage of shadowed areas like indoor railway stations or tunnels. In this framework we propose a solution able to overcome the problem without deploying additional repeaters and ground antennae. Our approach was thought with tunnel occurrence in mind, and pure traffic engineering mechanisms have been considered. The evaluation of proposed solutions, as discussed before, has been carried via simulation. Specifically, VoD-like services and DRT-like services are taken into account in order to develop an ad-hoc, event-driven, fluidic C++ simulator.

The proposal is based on the use of a few elements: a proxy server located on the vehicle, a smart buffer handling aboard the proxy server, a service-oriented resource sharing management. The proxy is in charge of decoupling users connection into two parts:

- An inner link, used in order to connect user terminal to the gateway. This could be a wired link or a WiFi service available on trains or other vehicles.
- An outer link, used by the proxy in order to connect to the satellite and hence to the rest of the outer network (Internet).

The proxy server is provided with a storage area available for both offline pre-fetching of files and streams, and temporary caching as well as stream buffering. This storage use is of fundamental impact, since it allows to decouple the speed retrieval of files between inner and outer links: in the inner part, users are served at the natural playout speed of a specific stream, while in the outer part, the proxy is made able to take advantage of the higher network speed; the excess information retrieved can be temporary cached in the proxy, and network resources can be fully devoted to other operations. Finally, it is also needed a resource sharing mechanisms, since proxy has to manage streams in a smart way, since the proxy is in charge of dynamically distribute satellite bandwidth between active connection. That is the simulation framework adopted.

The rationale of the proposed approach is that outer network resources could be suitable employed to fill in advance proxy buffers, as long as it is possible before forwarding data to the final users, and so eventually long outer link outages should be made transparent to the traveling user. The elastic buffering solution and the resource sharing algorithm, has been taken into account in the definition and implementation of A2M. In the simulation work, we focused on the evaluation of such a mechanism, also performing a comparison with a traditional approach that we named ED. Objective of the simulation was to test and explore the functionalities of A2M, evaluating its pros and eventually stressing its cons with respect to traditional operation. Moreover, each simulation is segmented in a transient phase, in which a simulation of network "warm-up" is performed, and a certain number of simulation runs (typically 50 or 100, but higher value are used when collected statistic do not show appropriate confidence level). Overall statistics are refreshed each time a simulation run exhausts, while data collected in the transient are discarded. The request process is modeled as a Poisson arrival processes for both VoD and DRT. Frequency of requests ranges in a wide spectrum, from very low load condition to infinite request rate. In particular, in a great number of simulation runs, we adopted very severe load condition, namely a "saturation" load was offered to the VoD server and/or to the real time video broadcast source. For saturation load we mean that connection requests are ready to be delivered each time an old connection expires. As to real time services, the system was stressed by considering a single everlasting real time flow to be delivered to vehicular users.

VoD streams are characterized by length (fixed or exponentially distributed) and natural rate (randomly selected in a user-given range). Date pertaining to VoD streams are sent in a strict temporal sequence, if losses arise, retransmissions are suddenly requested by the proxy, and no new data are sent until recovery procedures are complete. Real time streams are broadcast over a dedicated channel, by means of a user defined rate. Thus these data are received by the proxy in temporal sequence. During recovery procedure, real time services shares the same resources with VoD active streams.

We have modeled the satellite connectivity with an *On*/*Off* process with both *On* and *Off* periods exponentially distributed. The simulator user can set the average duration of *On* and *Off* periods. The only source of transmission errors in the simulation is the occurrence of an *Off* period, since no effect of attenuation, distortion in the propagation path nor elaboration errors at terminal level are considered. In other words, we are issuing only very long burst error, due to temporary lack of outer connectivity, and there is not any Forward Error Correction (FEC) technique that could suitably fit with such a long loss .

We assumed that satellite resources can be managed in a dynamic way, and very fast management procedure are used in comparison with outage durations. Finally, streams to be retrieved by the proxy and to be forwarded to final users, are modeled as fluidic flows that flush in the network links. No packet/message issues, transport layer features nor data link behavior were addressed in the current work.

§5.4 Performance evaluation

The evaluation of the effectiveness introduced by the A2M approach, as compared to the ED mechanism, is evaluated in this section. Specifically, sections §5.4.1 and §5.4.2 deal with VoD services, while the case of DRT services is discussed in §5.4.3. Additionally, simulation results taken from a train scenario currently under deployment are presented in section §5.4.4. Eventually, subsection §5.4.5 shows some effects of the concurrency of VoD and DRT services.

Here we describe most important parameters and models considered in the evaluation process, as listed below:

- requests statistics;
- file size distribution;
- admission control scheme;
- download protocol adopted;
- channel characterization;
- proxy buffering;
- proxy pre-fetching strategy.

Firstly it is relevant the statistical characterization of the way an object is requested by a client. By analyzing proxy server tracks and content distribution network statistics, it is possible to see that web objects are requested according to a Zipf-like distribution [108]. The Zipf law was originally formulated in terms of popularity of words in a text, and it states that ranking by decreasing popularity each object, the frequency of the ith object is:

$$(34) \qquad p_i = \frac{k}{i} \quad for\ \ i=1,2,..N$$

where

$$(35) \qquad k = p_1 = \left(\sum_{i=1}^{N} \frac{1}{i} \right)^{-1}$$

Zipf law has been generalized as follows:

$$(36) \qquad p_i = \frac{k}{i^{\alpha}}$$

where α assumes the value 0.986 for web object popularity[32]. Moreover we assume a file request process is a Poisson arrival process, i.e., the inter-request time is exponentially distributed. A second aspect is related to the file size distribution. It has been shown that the distribution of file size for web downloads fits a heavy-tailed distribution function (see [30], in which lognormal and Pareto distributions are used). Since we consider essentially a video retrieval scenario, a mostly uniform file size distribution can be a reasonable adoption. Admission control schemes and transport protocol selection affect QoS performance in terms of reliability and download times. In fact, according to the selected admission control scheme it is possible to prevent congestions and optimize the channel utilization, while the transport protocol determines the effective file download rate. For instance a standard TCP protocol could severely affect the transmission over a satellite link, while an UDP protocol should maximize the data rate at the expense of reliability, since no delivery checks are performed. In a VoD scenario the client download rate is almost fixed, while the proxy can store bytes at a variable rate. This implies that different protocols could be used on the server-to-proxy connection and on the proxy-to-client one; this is a further advantage of using a split download connection. As regards the channel characterization, the satellite transmits data over a reliable channel, using a suitable transmission power and adopting an appropriate coding scheme for error correction. Anyway the satellite transmission requires a line-of-sight between sender and receiver. When server and receiver (i.e., the proxy) are out-of-sight a channel outage occurs. The statistic of such events is related to the mobile behavior of the proxy. In particular a channel outage occurs when the train, with proxy on-board, enters a tunnel. Thus the channel outage probability is route-dependent. For the sake of simplicity we consider the line-of-sight/out-of-sight alternation as an *On/Off* process with both *On* and *Off* periods exponentially distributed. Finally the proxy caching and buffering strategy is responsible for channel usage optimization and for client connections outage prevention, as previously shown in section §5.2.3. Note that the channel capacity C and the user data rate R are not significant parameters as to their standalone meaning. In fact, due to the fluidic approximation adopted here, performance results will vary in accordance with the ratio C/R that represents the number of connections that can be admitted in parallel.

[32] In our work we have mainly adopted a Zipf-like model for file requests; moreover we tested also different distribution laws, in particular the uniform and geometric distributions. It is clear that the distribution law adopted determines the behavior of the system especially for what regards the optimal amount of data to be pre-fetched, but this is out of the scope of this work and no comparisons are reported here.

§5.4.1 Video-on-Demand services

We have considered a scenario characterized by homogeneous VoD streaming sessions, each requiring a constant playout rate equal to $R = 2$ *Mbps*. The wireless backbone capacity has been set to $C = 32$ *Mbps*, so the *C/R* parameters is equal o *16* connections. We assume that each streaming session lasts for an amount of time fixed to 1800 *s* (this is the time needed for internal network to complete the stream delivery, i.e., 1800 *s* is the stream duration at its playout time). To stress the wireless backbone, we have assumed a load regime. In what follows, this regime is also referred to as “saturation load”. Specifically, the number of simultaneous retrievals on the wireless backbone has been enforced not to overcome a maximum threshold (eight retrievals, in the simulation runs considered in this paragraph). As soon as a retrieval terminates, we assume that a new streaming request is immediately available. We remark that this is a special case of dynamic network operation, where new requests arrive at a very high (infinite) arrival rate. Numerical results obtained for Poisson arrival of requests and finite rate, are dealt with in the following subsection §5.4.2 for convenience of presentation, and will show that the saturation load represents the worst-case condition in terms of wireless backbone performance. We also remark that the number of users served, in average, in the saturation load conditions is actually greater than the above mentioned threshold, as a new streaming session starts as long as the previous streaming session has completed the information retrieval (but this can happen well before the stream delivery in the internal network ends). We have assumed outage periods (tunnels) to have an exponentially distributed duration, with mean value 180 *s*. The duration of the *On* periods is set depending on the channel outage probability target

$$(37) \qquad P_{ch} = \frac{T_{on}}{T_{on} + T_{off}}$$

In the simulation, resource management on the wireless backbone is assumed to be instantaneous. The proxy buffer space is assumed to be infinite (though, in practice, its occupation level always remains bounded). In a satellite network scenario, resource allocation will be managed at the data link layer, and additional management delays arise. However, we remark that, even for geostationary satellites, such signaling delays are negligible when compared with the time needed to cross a tunnel. Firstly, figure 71 shows the ED and A2M performance in terms of connection outage probability versus channel outage probability. In particular, we recall that the connection outage probability ($P_{co\text{-}out}$) is defined as the probability that a connection outage occurs for at least one time during the stream duration.

As shown in the figure, if no proxy were employed, this probability would rapidly converge to 100% as the channel outage probability grows. We see that the improvement provided by elastic buffering is remarkable. A2M significantly outperforms ED in all cases. As to buffer consumption, it is interesting, and perhaps not intuitive, to note that the buffer space needed on the proxy is very limited. Numerical results obtained during the 50% channel outage probability simulation run show an average buffer occupancy of about 2500 *s* (i.e., less than 1 and a half stream size), while this value decreases to as low as 500 *s* in the 75% channel outage probability case (clearly, the more severe the outage, the less loaded the buffers is).

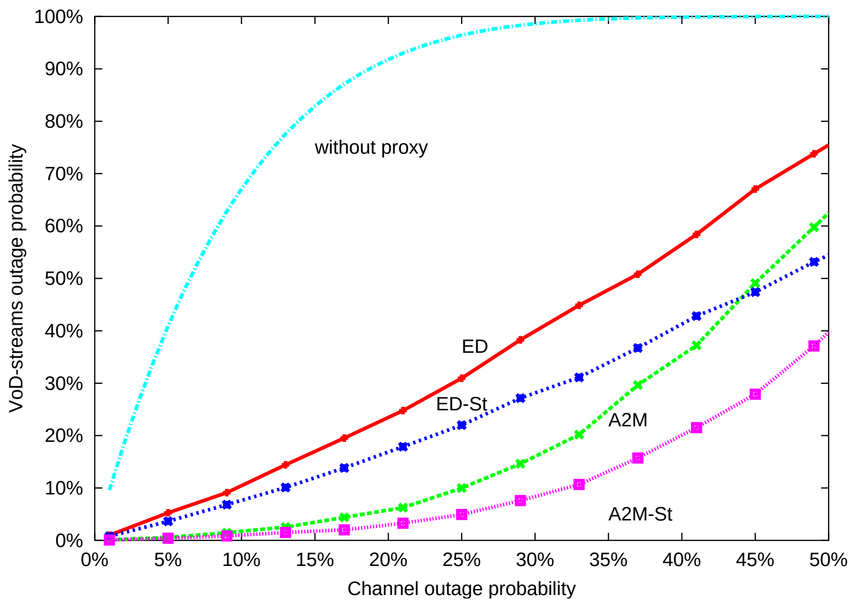


figure 71 - Video-on-Demand: connection outage probability vs. channel outage probability

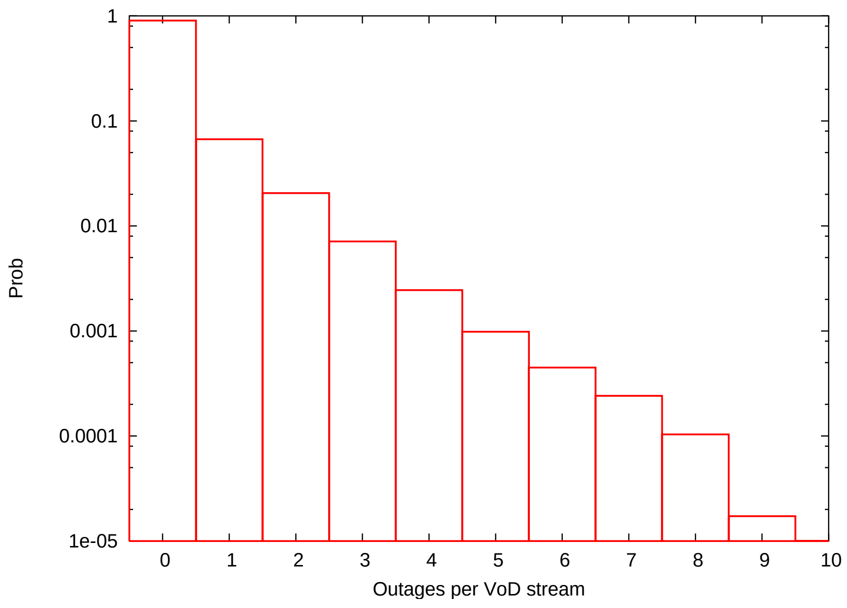


figure 72 - Probability that a Video-on-Demand stream experiences #n outages, with A2M

To demonstrate that the proposed proxy management scheme can be efficiently run in conjunction with a caching or pre-fetching mechanism, the figure reports the performance of the ED and A2M schemes (labeled as ED-St and A2M-St) in the assumption that 30% of the incoming requests match files preloaded in the proxy memory for half of their size (see [66], [90] for insights in partial pre-fetching schemes). As expected, pre-fetching leads to a performance improvement, but what is interesting from our point of view is that A2M shows a further relative advantage over ED.

Drawings in figure 71 are limited to present the probability than at least one outage occurs. To complement these results, figure 72 reports the probability distribution of the number of outages experienced by a single stream. A 25% channel outage probability, and the A2M mechanism, are considered in this figure. As shown by the figure (note the y-axis log scale), the probability that a stream experiences two or more outages is very low.

§5.4.2 Further results for VoD-like services

As in the previous subsection, for the sake of simplicity, simulations considered here do not take into account transport protocols and downloading files simply behaves as constant bit rate (CBR) application, eventually stopping and resuming after a failure. A lot of the reported performance results have been obtained for Poisson modeled offered traffic, in very high load conditions, i.e., with a very high request rate, similar to "continuous load" when the Poisson arrival rate diverges ($\lambda \to \infty$). This turns in a framework in which the accepted load is always to the maximum admissible by the admission control algorithm. Even if this is not the real case, the high load assumption is intended to stress the A2M mechanism in order to show its robustness to traffic load. Moreover, the continuous load assumption allows the study of the intrinsic beneficial effects of A2M, since the $P_{co\text{-}out}$ performance is a function of the caching scheme instead of both proxy caching and variable accepted load conditions. In any case, this section also report some performance figures attained with a reduced value of λ.

As a starting point, figure 73 shows the relation between download failure (connection outage probability, $P_{co\text{-}out}$), the channel outage probability (P_{ch}), and the max number of admitted connections in high overload conditions (λ=100 requests/s) when A2M algorithm is adopted. No pre-fetching is considered and a uniform distributed frequency request rate is adopted. In the same simulation conditions, figure 74 shows how the connection outage probability reacts to the variations of the accepted load, in terms of maximum number of admitted connections. It is remarked that with a small channel outage probability (5%), the connection outage probability becomes intolerable as the number of admitted connections grows and approaches the maximum (i.e., *C/R* = 16 connections). In terms of overall download time experienced by clients, figure 75 shows that downloading a file takes a time ranging between a minimum time (equal to the ratio between the file size *S* = 10^5 *Kbyte,* and the user rate *R* = 2 *Mbps*, so that *S/R* = 409 *s*) for low load and low channel outage probability, and almost twice that value, for download in high load and outage probability.

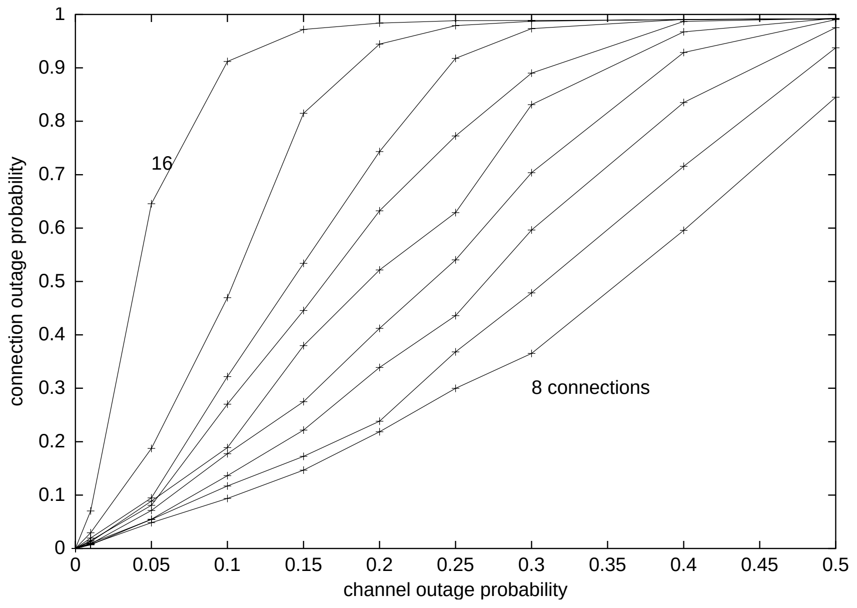


figure 73- Connection outage probability vs. channel outage probability with uniform content requests distribution and without pre-fetching

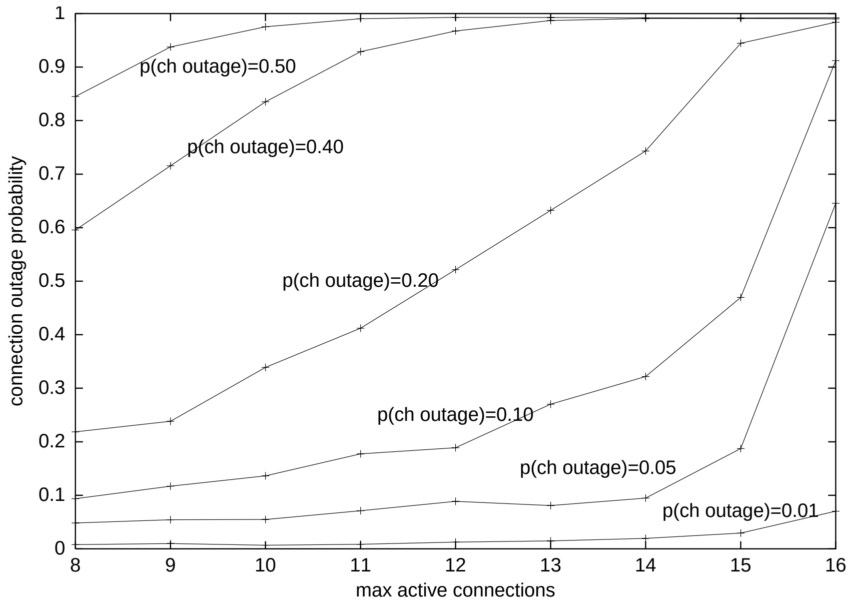


figure 74 - Connection outage probability vs. proxy utilization with uniform content requests distribution and without pre-fetching

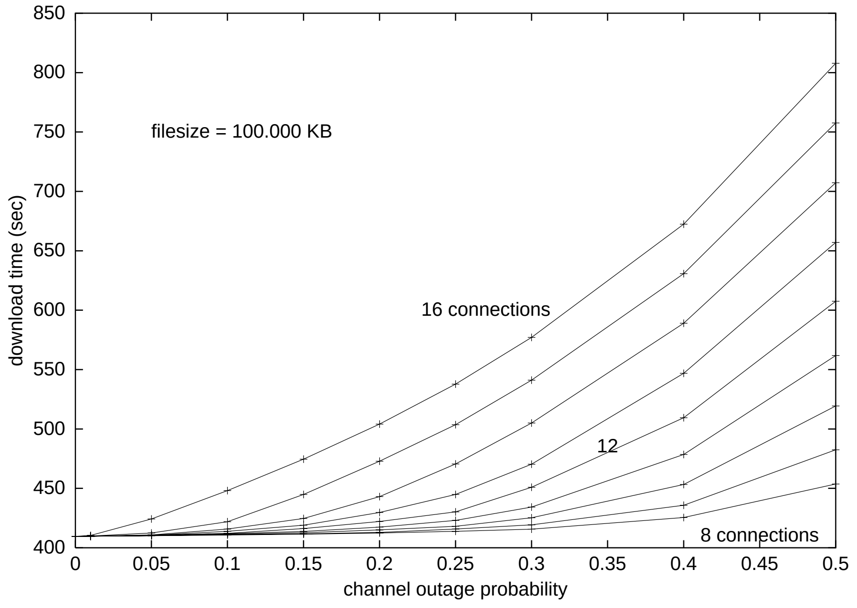


figure 75 - Behavior of normalized client download time as a function of the channel outage probability for several target utilizations

In the remaining figures a Zipf-like distribution law is adopted and pre-fetching effects are also evaluated. The file population is set to 10000 units, and α=0.986. Both figure 76 and figure 77 give a performance comparison between A2M and ED operation effects when a first half of the files characterized by the highest hit ratio is cached. As reported in the figure keys, a fraction of the mostly requested 100 files is pre-fetched, representing a cumulative request probability of about 51.3%. In particular figure 76 is related to low request rate conditions (λ = 0.02 requests/s), while figure 77 reports simulation results obtained in an quasi-continuous load framework (λ = 100 requests/s). Each figure plots four curves, two for A2M and two for ED simulations; for each bandwidth allocation scheme, the two curves represent simulations with and without pre-fetching.

It is obvious that pre-fetching algorithms improve outage performance, due to the probability that a client request matches a (partially) pre-fetched file. Moreover simulation results show that using exactly the same pre-fetching algorithm, A2M performance looks better than ED. A2M performance is well below the ED one, especially in low load conditions (figure 76), but this is still true also in high load conditions (figure 77). It is stressed that, using A2M, it is possible to obtain a connection outage that is less than a half of the probability experienced in traditional ED framework.

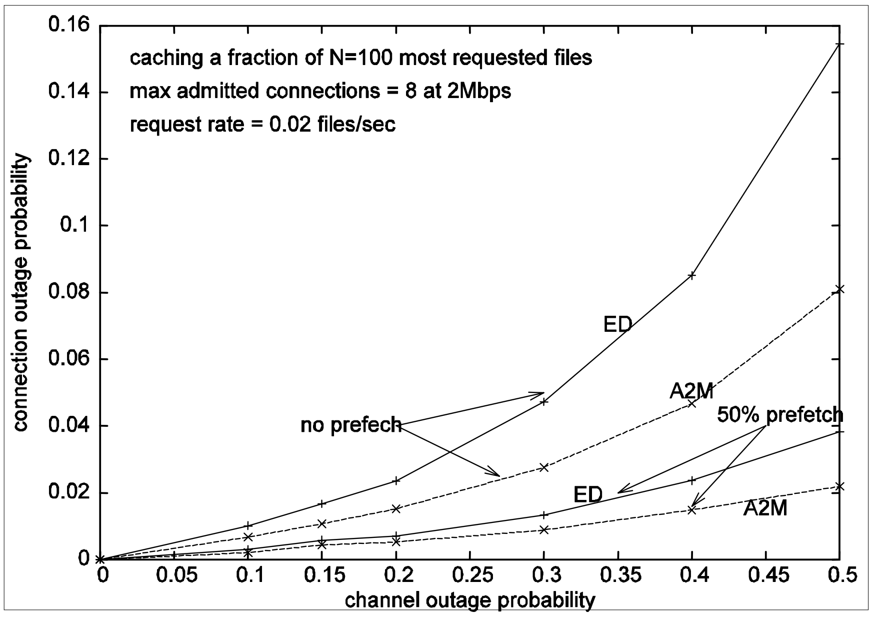


figure 76 - Connection vs. channel outage probability with max 8 connections in low load

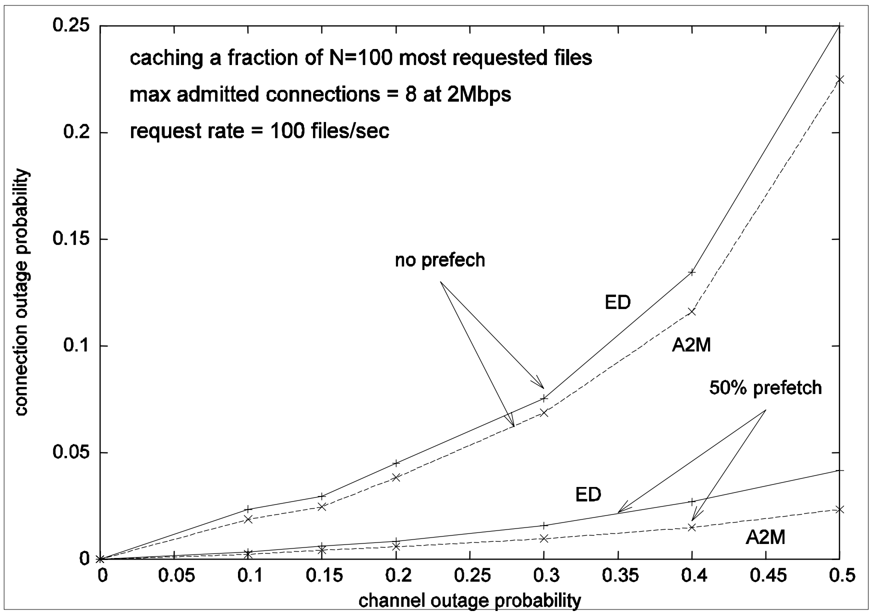


figure 77- Connection vs. channel outage probability with max 8 connections in high load

The pre-fetching gain is also clearly shown in figure 78 to figure 80. In these figures a partial pre-fetching scheme is proposed, as before. Different curves have been obtained by varying the pre-fetching fraction, the load and the accepted load. Firstly, in figure 78 the case that only eight connections are simultaneously admitted is considered, while the overall request rate is quite high. A small pre-fetching fraction as large as 10% of 100 among the most frequently requested files, allows a notable reduction of connection outage probability of one order of magnitude. Additional pre-fetch (as to the 50%, shown in the third curve of figure 78) further improve the performance, but with higher costs in terms of memory requirements. Note also that the benefic effect due to pre-fetched data, saturates as the pre-fetched grows, so that the advantage coming from additionally augmenting the pre-fetching rate rapidly disappears. On the other side, considering figure 79, using all the possible proxy-client connections (i.e., 16) yields a smaller gain using pre-fetching, i.e. a greater pre-fetching fraction is needed in order to obtain the same performance as before, even if the offered load is well reduced with respect to the case shown in figure 80. Obviously, the more the proxy is loaded, the more storage resources are required. This is clear from the consideration of curves depicted in figure 80, where a $\lambda = 100$ request/s is adopted, instead of $\lambda = 0.02$ request/s used in figure 79.

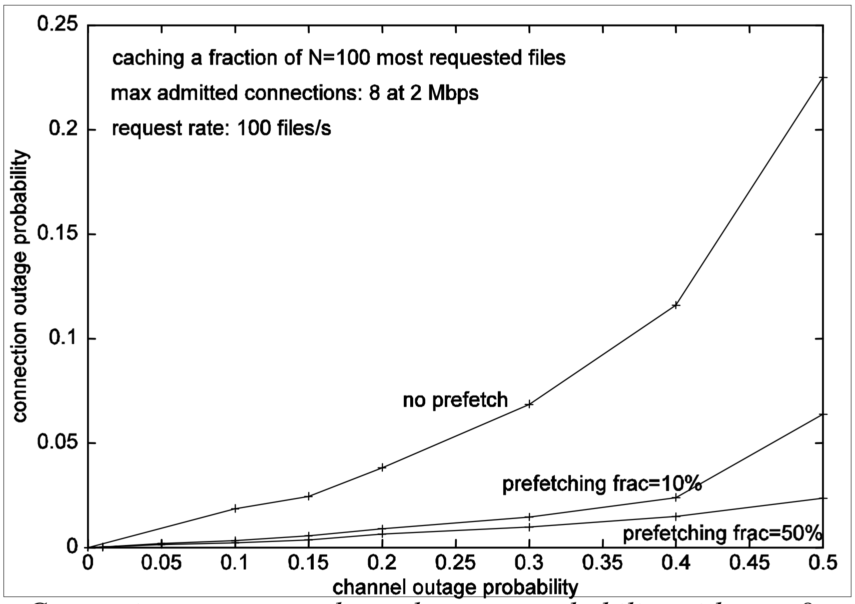


figure 78 - Connection outage vs. channel outage probability with max 8 connections admitted and high offered load

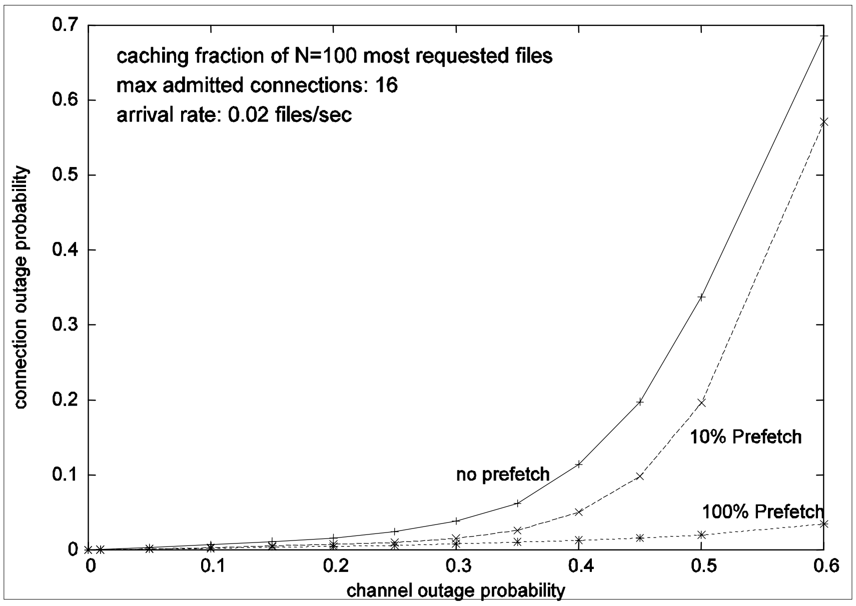


figure 79 - Pre-fetching gain in low request rate conditions

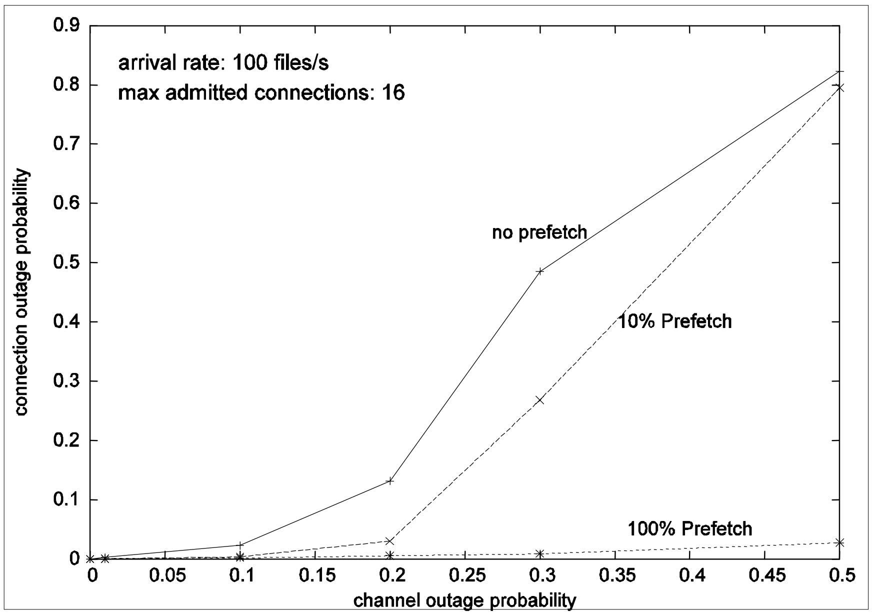


figure 80 - Pre-fetching gain in very high request rate conditions

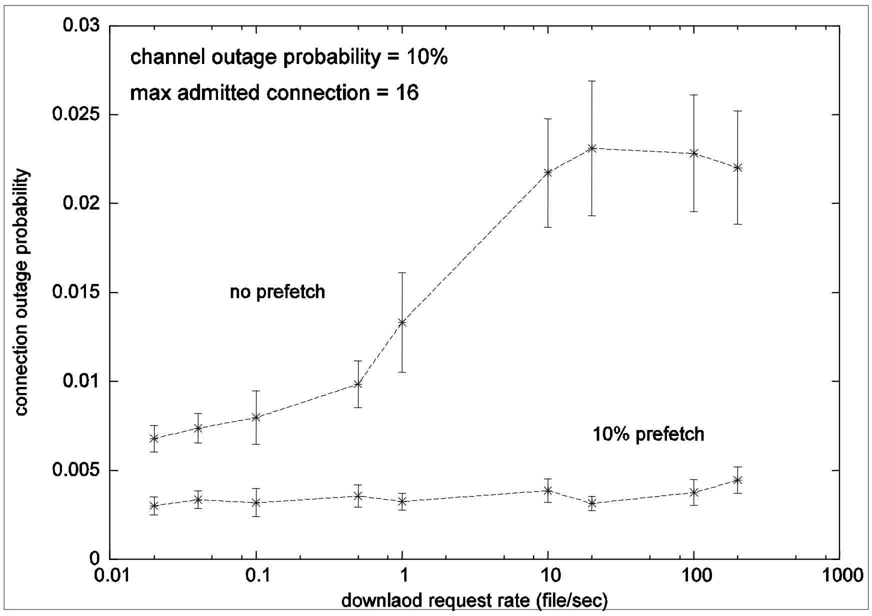


figure 81 - Connection outage probability as a function of request rate and pre-fetching gain for a 10% channel outage probability.

Finally figure 81 and figure 82 give a direct insight of the dependence between offered load, in terms of λ, and the connection outage probability $P_{co\text{-}out}$, given a pre-fetching setting and a channel outage probability. When the channel outage probability is reasonable (see figure 81) the A2M approach, with a light form of pre-fetching, appears to be robust to λ fluctuations. With a full loaded proxy (16 connections), and a 10% pre-fetch scheme as described above, a satellite channel outage probability equal to 10% yields a connection outage probability of about 0.3% whatever the value of λ. Even though large error bars are reported in figure 81, the gain obtained through the pre-fetch approach is remarkable but not indispensable, since the system adopting A2M seems to be robust to relative low channel outage probabilities. Conversely, if we consider significant values of P_{ch}, it arises the necessity to use pre-fetch schemes as longer as possible. Curves plotted in figure 82 demonstrates that, in high loss condition, a small amount of pre-fetched data are scarcely useful.

§5.4.3 Delayed Real Time services

Performance results concerning the support of DRT services are reported in figure 83. Results were obtained using DRT streams instead of the VoD ones. The wireless backbone was loaded with a single everlasting broadcast transmission at *2 Mbps*. In order to present cleaner results, we have not included VoD sessions in this scenario. The two major design parameters to be considered in such simulations are:

- the playout delay to be adopted before delivering a stream to the end user,
- the amount of extra bandwidth resources that are reserved to recovery procedures.

The percentage of real time stream lost as a consequence of connection outage (not recovered data), versus the recovery bandwidth, i.e., the extra-bandwidth with respect to the natural real time broadcast rate, is reported in figure 83. Two initial playout delays are considered: 300 *s* and 720 *s*. The channel outage has been modeled by assuming exponentially distributed outage periods lasting, in average, 180 *s*, and exponentially distributed visibility periods with mean value 1620 *s* (i.e., a 10% channel outage probability).

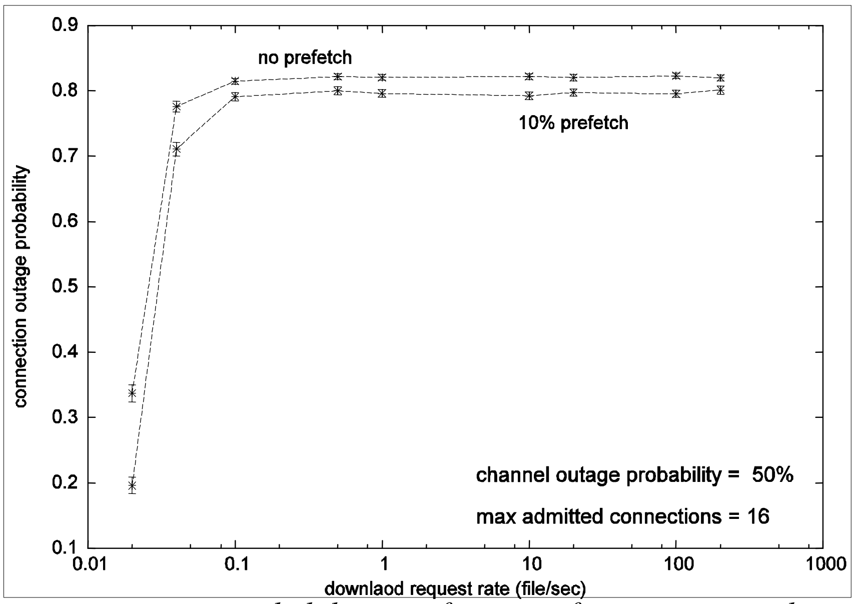


figure 82 - Connection outage probability as a function of request rate and pre-fetching gain for a 50% channel outage probability

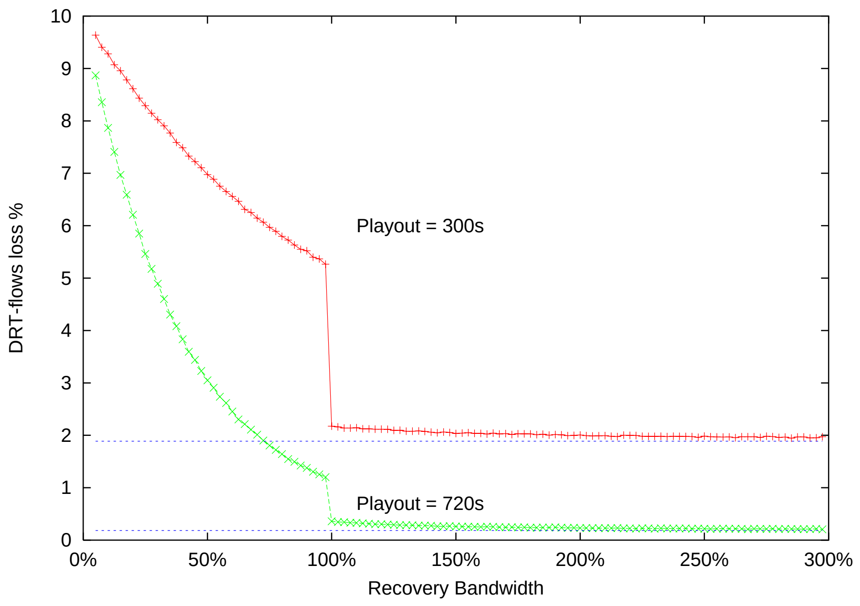


figure 83 - Delayed Real Time stream: fraction of not recovered (lost) data

The figure shows that as long as the recovery bandwidth increases, the performance improves. The amount of lost fraction of the stream converges to an asymptotic value which simply represents the probability that a single tunnel (i.e., an outage period) lasts more than the initial playout delay: in such a case, a fraction of the stream will be lost regardless of an eventually unlimited extra bandwidth available. A sharp improvement in the performance is suddenly encountered when the extra bandwidth available is equal to the stream rate. This operational point corresponds to $K=1$, as defined in section §5.2.4. In fact, for $K \geq 1$, the recovery bandwidth is greater or equal than the stream rate, and thus a connection outage can occur only inside a tunnel. Conversely, when $K<1$, the recovery rate (the rate at which the proxy buffer is filled) is lower than the stream rate (the rate at which the proxy buffer is emptied), and thus connection outage may occur outside a tunnel, as long as the buffered information exhausts (see also figure 70). Note that in figure 83, when the recovery bandwidth (normalized to the broadcast flow bandwidth) grows, DRT-flow loss decreases, and an asymptote is shown, that depends of the playout time. In fact the connection outage probability (with infinite recovery bandwidth) is equal to the probability of a channel outage longer than the playout time. Using exponential distribution with mean $\mu = T_{off}$, if D is the playout delay adopted, T the overall simulated time, and $P_{ch\text{-}out}$ the channel outage probability, then the mean number of tunnel per simulation is:

$$(38) \qquad N = \frac{T}{T_{off} + T_{on}} = \frac{T}{\mu + \frac{1 - P_{ch-out}}{P_{ch-out}} \mu} = P_{ch-out} \frac{T}{\mu}$$

While the mean outage time per tunnel is:

$$(39) \qquad L = \int_{D}^{\infty} (x - D) e^{-\frac{x}{\mu}} dx = \mu e^{-\frac{D}{\mu}} .$$

The average percentage loss is:

$$(40) \qquad L_{\%} = 100 \cdot \frac{L \cdot N}{T} = 100 \cdot P_{ch-out} \cdot e^{-\frac{D}{\mu}} .$$

In figure 83, horizontal asymptotes are computed according to the previous formula, and they only depends on the adopted playout, being μ = 180 *s* and $P_{ch\text{-}out}$ = 10%.

§5.4.4 FIFTH Train scenario: Italian railways

In this section, we present performance results taken from the reference deployment scenario tackled in the European Community IST project FIFTH (Fast Internet for Fast Train Hosts). The goal of this project was to deploy satellite communication for providing Internet and multimedia streaming services aboard of high speed trains. The following figures report results for a deterministic tunnel pattern, taken on the railways path between the Italian cities Rome and Florence (covered in about 1.5 hours by high speed trains). About 23% of this path is covered by 44 tunnels, the longest one lasting for about 180 *s*. The average crossing time for a tunnel is 24 s, although the tunnel sizes show a bursty pattern, and tunnels appear to cluster.

Results for VoD services are presented in figure 84. It assumes eight streams in saturation load, and shows the relationship between the extra-bandwidth to be made available on the satellite channel, versus the extra-time needed to complete the vision of the streams. We have reported such a performance figure as it provides a thorough insight in terms of exploitation: during a connection outage, advertising might be delivered to the customers, and the figure may be intended to quantify how much advertising overhead a customer shall suffer versus a given satellite bandwidth dimensioning. From the figure, it appears that a small extra-bandwidth assignment is able to drastically reduce the download time. For example, a 20% extra sizing of the satellite bandwidth leads to about an extra 13% of the stream delivery time. However, the figure shows that, to achieve a marginal completion time overhead, a considerable satellite bandwidth over-provisioning (up to 70% and more) is required.

Eventually, figure 85 reports results for a delayed broadcast (in real time) stream. Here, the obvious service requirement for deployment purposes is to provide a seamless support of the broadcast channel, i.e., design the system so that no outage periods would occur. The plot reported in the figure shows the initial playout delay (y-axis) and the extra bandwidth sizing (x-axis) necessary to achieve an uninterrupted service (zero-loss). The figure clearly shows that initial playout delay can be traded off with extra bandwidth. It also shows that a minimum initial playout delay equal to the longest involved tunnel (about 180 *s*) is necessary to provide an uninterrupted service support.

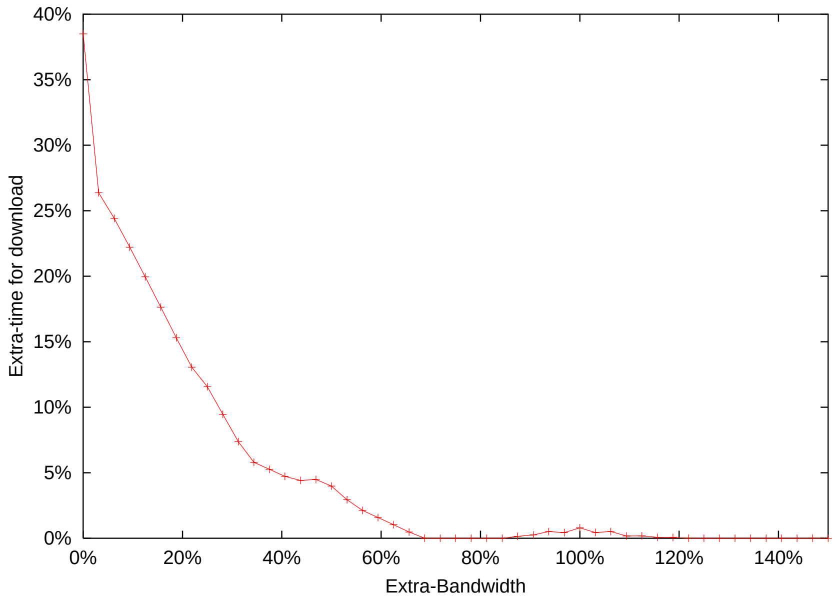


figure 84 - Rome-Florence railways path: Video-on-Demand performance

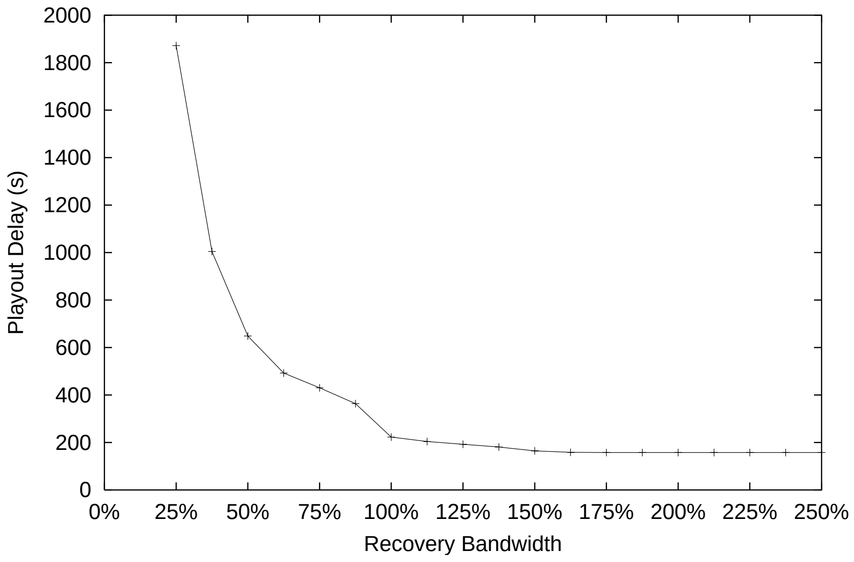


figure 85 - Rome-Florence railways path: Delayed Real Time zero-loss trade-offs

§5.4.5 Mixed VoD and DRT services evaluation

In this section we report simulated results about the coexistence of VoD and DRT services sharing the same bandwidth over the satellite link. Simulations was carried out mainly following two approaches: performance are considered as a function of the number of VoD flows, or as a function of the channel outage probability. In both cases, the overall satellite bandwidth was set to 2048 *Kbps*, each VoD flow requires 128 *Kbps* at user side, a single broadcast channel is considered at 1024 *Kbps*. Note that different *C/R* values apply for VoD and DRT services, and that is the reason why we specified the relative data rates. Playout delay simulated for DRT services was set to 300 *s*, while each VoD flow, if no interruptions occur, lasts exactly 1800 *s*. Tunnels average duration is 180 *s*. As performance figures we have selected: i) the probability that a VoD flow fails; ii) the extra-time to completely deliver a VoD flow to the final user, in percentage; iii) the amount of data, in percentage, pertaining to a broadcast transmission, and that the system was not able to deliver to the final user.

Firstly figure 86 depicts performance of the system reported in the assumption that an ED sharing scheme is adopted. Eight VoD flow are admitted simultaneously, and a saturation load is offered, thus the number of active VoD flows is, in practice, stuck to eight. This mean that half resources are requested by the VoD service, and 1024 *Kbps* left for DRT (i.e., *K*=1, in this specific case).

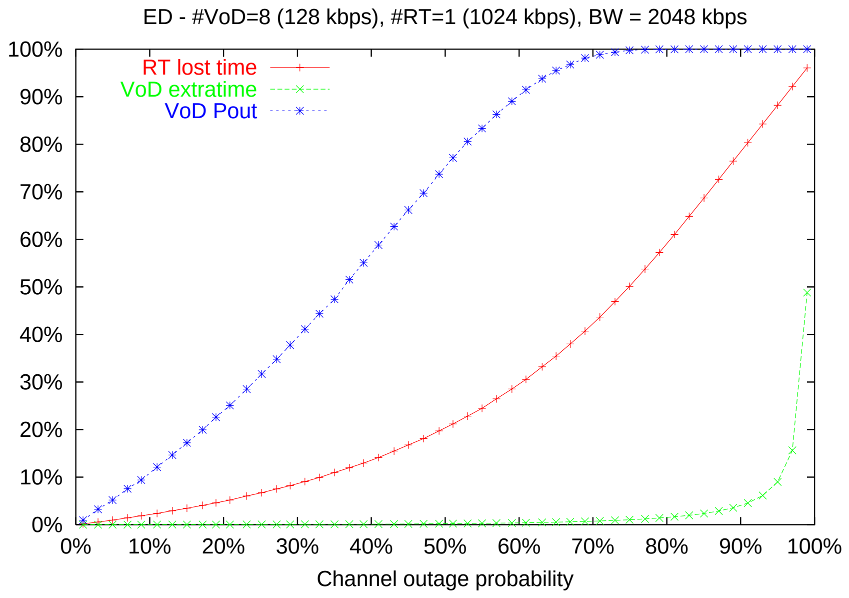


figure 86 - ED performance using both VoD and RT services vs. channel outage probability

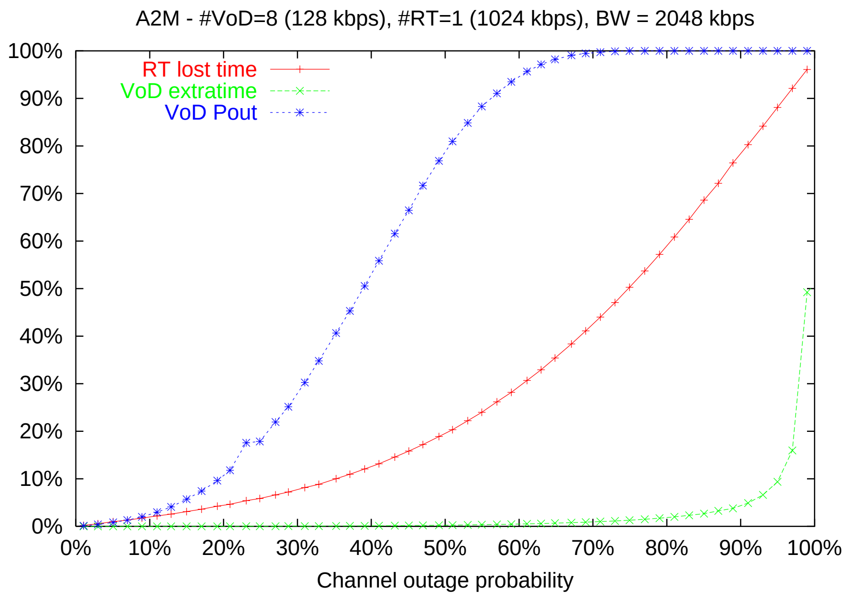


figure 87 - A2M performance using both VoD and RT services vs. channel outage probability

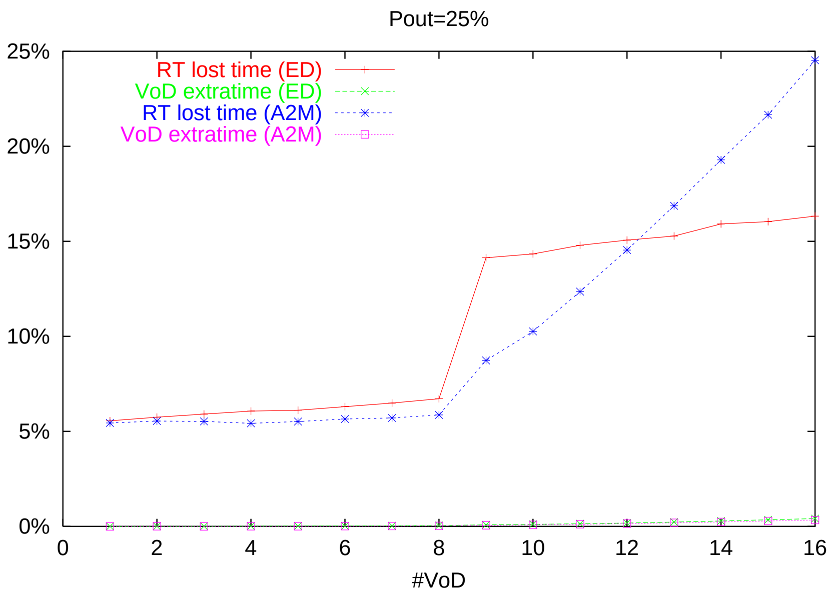


figure 88 - A2M and ED performance using VoD and RT services vs. number of allowed VoDs

DRT performance result reasonable for a large range of channel outage probability values: let say that a 30% of channel outage probability can be recovered by using a 300 *s* playout delay; when outages are more frequent, a greater initial delay is needed for DRT stream delivery. VoD performance looks pretty in terms of extra-time requested in order to complete the download, but if one has a look to the probability that a VoD flow stops during its execution, it is clear that a great number of interruptions occurs if the channel outage probability overcome a threshold of about 10%.

In comparison with performance illustrated in the previous figure, A2M is evaluated in figure 87. It is possible to say that DRT performance do not improve deeply, while VoD service enhancement is remarkable, and can be quantified in a 10% gain in terms of sustainable channel outage probability.

Afterwards, if we consider the relation between services performance and number of allowed flow, in the same condition of above, figure 88 shows a comparison between ED and A2M performance. As to the scenario simulated in such a figure, a 25% channel outage probability is enforced, and no particular differences arise in terms of overall time needed in order to definitely convey a video to the final user. It is not reported here, but, as to VoD services, the great gain due to the use of A2M consists in the reduction of outage occurrence, as shown in other cases reported before. On the other hand, as to the DRT services, a quite interesting results can be highlighted in figure 88: A2M is useful and provide better performance than ED, but this is not still true if the number of admitted VoD flows grows too much. In fact, since we adopted a saturation load, using *n* VoD flows implies that, in average, only 2048-128**n Kbps* can be available for DRT recovery procedures, and this, in turn, makes the equivalent value of *K* decrease. In the selected case, the maximum number of VoD connections should be set to 12 (i.e., $K = 0.5$) if A2M beneficial effects have to be granted. Note also that when resources allotted to VoD overcome one half of the overall satellite resources, DRT performance drastically falls both in case of ED and A2M usage. Nonetheless, in that transition the system behaves much more robustly to VoD load if A2M is adopted.

Finally, figures figure 89 and figure 90 illustrate the behavior of the system in case of contemporary usage of both VoD and DRT services when the real path between Rome and Florence is considered as in figure 84 and figure 85. In particular, both pictures refer to a scenario with a single everlasting broadcast stream at 1024 *Kbps* and eight VoD flows at 128 *Kbps*, 1800 *s* lasting, with saturation load. In the case of VoD performance, a 720 *s* playout was fixed for DRT services. In those pictures, in correspondence with service whose performance is reported, the percentage in the x-axis represents the available resource in comparison, respectively, with the DRT (figure 89) and VoD (figure 90) requirements, once subtracted the nominal amount of resource that could be needed by the other service, i.e., VoD and DRT respectively. In other words, the point corresponding to 100% is that of a system that can accommodate twice the bandwidth of permitted VoDs or all VoDs plus a complete and continuous retransmission of the broadcast channel.

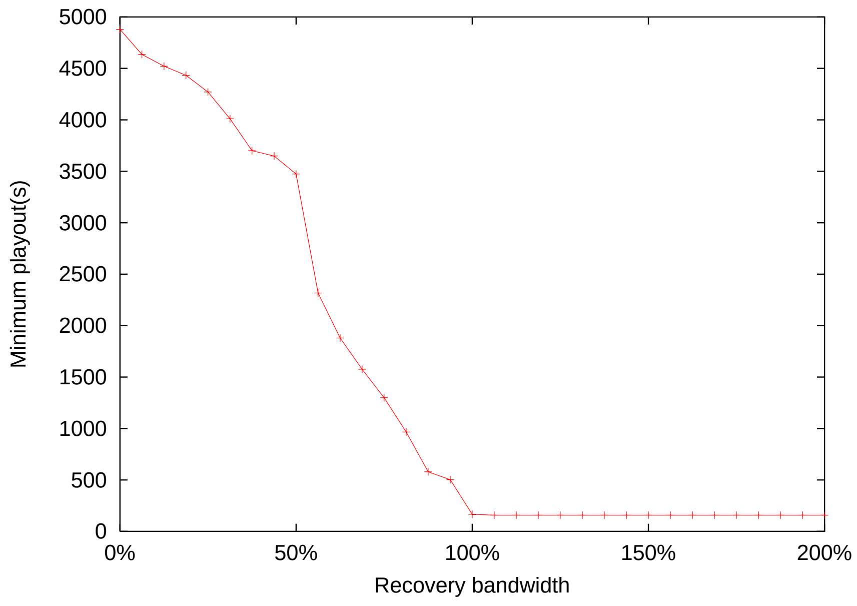


figure 89 - Rome-Florence path: zero-loss performance for RT services in presence of 8 VoD flows @128 Kbps

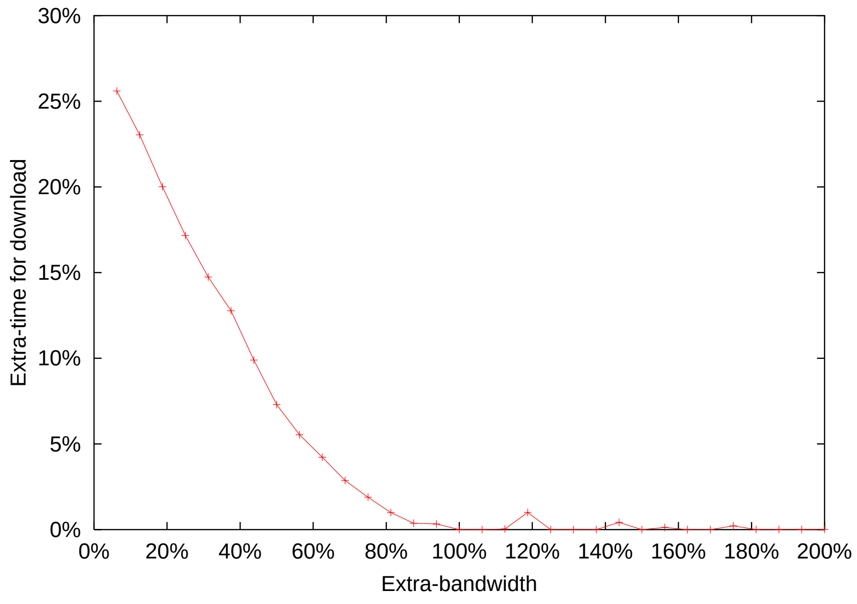


figure 90 - Rome-Florence: performance of VoD in presence of a RT stream @1024 Kbps, 720s playout

With respect to performance obtained using only one service at a time, a degradation can be noticed both for DRT and VoD, but a reduction in the overall bandwidth consumption can be obtained if we consider that the sum of services requires less than the sum of resources separately required by each service. As a matter of fact, the presence of a service acts as a reduction of resources available for the other service, but the same performance reported in figure 84 and figure 85 can be obtained by using less and less than twice the resources needed when using separate channels for DRT and VoD services. However, the system can be suitably tuned as before, and quality can be guaranteed by means of extra-bandwidth usage and reasonably long playout delay to be used for DRT services.

§5.5 Conclusions

In this chapter we have reviewed the advantages provided by the adoption of elastic buffering in the support of multimedia streaming. Elastic buffering relies on a proxy devised to split data flows into two parts. The proxy to client stream is delivered at the natural playout rate. Conversely, the stream is downloaded from the remote server to the proxy at a rate possibly higher than playout.

We considered VAN users attempting to download multimedia contents from the outer network, relying on a proxy. VoD/MoD and DRT services have been taken into account, and a suitable bandwidth sharing algorithm, A2M, has been proposed to confer reliability to both kinds of service. Excess retrieved information is temporarily stored in the proxy buffer for future delivery. Simulation results show the effectiveness of elastic buffering to compensate intermittent connectivity occurring during outage periods. Moreover, they show the superiority of the proposed A2M mechanism with respect to a traditional approach in which the bandwidth on the wireless backbone is equally shared among the active streams. This work leaves open the issue of how to deploy the proposed mechanisms over the Internet. Such a deployment requires the thorough investigation of signaling mechanisms between the proxy and the remote server for the dynamic support of play/still/resume functions implemented at the proxy side, and their application on top of a transport session between proxy and remote server to achieve information download at a speed higher than the natural streaming playout.

The basic ideas proposed in this chapter was developed within the frame of the European Community funded IST FIFTH project, which employs a bidirectional communication between a GEO satellite and a traveling train equipped with an auto-seeking on-board parabolic antenna. The investigation carried out for this scenario has assumed a dedicated satellite link for the moving train, as this was the project demonstration target. Other wireless access frameworks were also addressed by the Italian VICOM project. Next steps in research mainly relate to the transport protocol influence, and the implementation of an admission control scheme, taking advantage from the amount of pre-fetched data in the proxy. Further performance insights are needed to understand the effectiveness of the proposed approach in a scenario where a fleet of trains, sharing the same satellite link, needs to be networked.

Chapter VI

QoS using interactive DVB satellite systems

In the first part of Chapter IV we gave some insight in the interoperation of different signaling protocols, and in particular on the interaction between GRIP and RSVP deployed, respectively, in a DiffServ and an IntServ domain. Signaling extensions represent a key factor for the spreading of IP networking capabilities that allow to support QoS-aware applications. Hence we have shown that heterogeneous systems can be made compatible for end-to-end QoS-aware cooperation. This turns in the possibility to deploy integrated scenarios with heterogeneous value added features to be harmonized in a shared fashion. As long as resources managed issues are represented by bandwidth consumption, queue length design, dropping algorithm implementation, scheduling approach to priorities, packet marking, network policing, and so on, network devices can be configured to handle IP traffic in the proper way, e.g., following the IntServ approach, the DiffServ strategy or the legacy Best Effort scheme. The network provider could experience the necessity to upgrade its devices with new additional features for the management of low level mechanisms such as AQM and marking, as discussed in Chapter III and Chapter IV. It could be also needed to introduce some smart bandwidth and resources handling tools, as discussed in Chapter V.

However, the envisioned IP-layer (or network-layer, in general) mechanisms rely on heterogeneous data link technologies over definitely different physical infrastructures. Thus an important point for the provision of QoS support in a network, is the interaction between protocols that run in the network-layer or above, and lower layers protocols. Note that this is not a novel approach to QoS, since other approaches to reservation, like RSVP, are envisioned to collaborate with QoS-aware Layer 2 devices. The point is that lower layer capabilities sometimes can hardly be summarized in terms of bandwidth and processing delay, since they could additionally include different kind of elaborations as packet encapsulation, fragmenting or packing, and also data link-level priority schemes. Therefore a mapping of Layer 3 services to Layer 2 services is often required, even though a complete study of this mapping has not been carried out for every kind of data link technologies, in particular for satellite, where gateways are commonly envisioned between terrestrial and flying nodes, and satellite segments have been considered standalone networks. Satellite technologies are going to be particularly relevant in this days, because of the novel possibility to use them as bidirectional gates to remote destination, also suitable for interactive multicast applications.

The knowledge of underlying means and physics details permits a finer tuning of network, transport and application parameter. One of the mostly relevant case is precisely represented by satellite networks, where considerable bandwidth resources are typically available, but severe impairments are due to remarkable delays in the transmission path. For instance, the support of reactive applications (e.g., using TCP) over geostationary satellites has been thoroughly investigated, and a number of proposals has been produced in order to modify the behavior of TCP and other transport/application protocols to cope with the satellite link characteristics as in [2], [33], [44], [79] and [101]. The same is for IP interoperability with satellite transport mechanisms [62]. In this chapter we conclude the discussion about the network support for QoS applications by analyzing the impact of geostationary satellite network nodes on DiffServ and IntServ IP architectures. The DVB-RCS satellite transmission system is adopted as a reference scenario, where GRIP for admission control and signaling extension issues will be reconsidered, as well as DiffServ and IntServ capabilities. The problem of providing QoS support in DVB-RCS satellite networks is also addressed and observed from an integration perspective with the deployed IP architecture.

Hence the present chapter is organized in the following way: section §6.1 describes a scenario in which DVB-enabled satellites are integrated in a network with IntServ and DiffServ nodes. In section §6.2 it is shown the interoperability between QoS models provided by IntServ, DiffServ and DVB-RCS. Section §6.3 describes a concrete case and provides a mapping between IntServ services and DiffServ classes, and, in turn, between DiffServ classes and DVB-RCS Profile Classes. Eventually, §6.4 proposes some conclusive remarks. Further information about satellite releases of DVB systems can be found in ISO and ETSI standards, [36], [37], [38], [39], [59], [60] and therein reported references.

§6.1 DiffServ and IntServ in a satellite DVB-RCS system

In this section we consider a network in which an internal link is implemented via a GEO satellite hop provided with DVB-RCS technology. Thanks to the adoption of DVB-S and DVB-RCS standards, the satellite node can be used as a common asymmetric link, in particular the RCS standard allows any users to send data from ground to satellite using the same antenna adopted for the signal reception. To date, flying DVB-RCS satellites are provided with transparent payload, so that they simply act as bent pipes, particularly suited for long distance communications. The upload pipe is shared among different users by means of a frequency and time division multiple access technique. However, additional capabilities are envisioned for a short term evolution of satellite payload, and regenerative payload systems, endowed with packet switching facilities, will be soon available for common usage. Hence, the intelligence of actually deployed DVB-RCS satellites is concentrated on specialized ground stations:

- a Network Control Center (NCC) is responsible for the generation of a Terminal Burst Time Plan (TBTP) compatible with the on-board switching requirements and the signaling procedures for the resource allocation in a MF-TDMA shape; it is also responsible for satellite connections establishment;
- a Satellite Terminal (ST) is responsible for the selection and the fine scheduling of the data to be transmitted, the QoS support and the initiation of resource requests for each application.

Interworking between terrestrial and satellite QoS architectures is mostly a matter of Layer 3 to Layer 2 translation rules. The optional support to mesh connectivity in a multi spot beam communication scenario additionally requires mesh control signaling implemented by ground segment entities in the control-plane. Ad hoc interoperation between this satellite signaling and the Layer 3 IP signaling for QoS is required in addition to Layer 3 to Layer 2 translation rules; finally, a full flagged support for packet switching operation will require an enhanced satellite on-board processor, capable of several QoS related functions, such as QoS-aware switching, QoS-aware buffering and buffer management, QoS-aware scheduling. The requirements on the NCC and the ST will be somehow relaxed as far as the satellite air interface is concerned, but the overall capability in terms of end-to-end QoS support will increase also in view of strong cooperation between the overall QoS capabilities of the satellite system and those of terrestrial segments.

The optimization of terrestrial and satellite terminals interoperation is driven by the IP QoS component characterization. To this goal the following issues are worth being considered:

- the end user shall adopt a signaling scheme to notify its quantitative traffic and performance requirements, as derived by the invoked application. IntServ architecture with RSVP and its possible alternatives and evolutions represent a suitable references and the satellite system should be able to interoperate effectively with them, by means of a local admission control for satellite resources allotting. This is due to correctly managing the traffic packets in terms of DVB-RCS capacity categories at MAC level, and for a proper forwarding of the signaling packets, also considering the internal mesh connection control signaling.

- Different implementations exist of the DiffServ architecture, examples being a static provisioned approach, or a dynamic solution as the one proposed in this thesis by means of the GRIP mechanism. The satellite system should be able to interoperate effectively with each of them, by translating the relevant signaling (e.g., GRIP probes and similar implicitly conveyed signaling messages) and DSCP codes to the most appropriate DVB-RCS capacity category at MAC level.
- The end-to-end delivery system is a sequence of domains, each one presenting its own specific QoS architecture (if any). Adjacent domains interoperate on the basis of SLAs which adapt the QoS categories on the two sides of the sub-network edge.

A number of possible scenarios stem from the application of these concepts to the DVB-RCS link configurations. For the sake of simplicity we selected a particular scenario able to highlight the main configuration issues in such a kind of network. Hence we decided to focus on a network framework where IntServ is deployed as to access network, i.e., at the edges of the overall network. DiffServ is then employed in the core of the network, and a DVB-RCS-enabled satellite is located in the core, i.e., inside the DiffServ sub-network.

In this selected scenario, the satellite sub-network does not provide any QoS related function or capability in addition to standard DVB-RCS features supported by MAC and mapping/translation between the MAC and the IP QoS that applies to the domain which the satellite belongs to. The described scenario is depicted in figure 91: a hybrid IP QoS architecture is proposed, and a DiffServ sub-network is nested in an IntServ-to-IntServ as end-to-end service provisioning system. RSVP is adopted in IntServ, while the core DiffServ QoS issues are addressed by the GRIP mechanism. Note that RSVP messages have to be encapsulated in GRIP procedures, and, in turn, GRIP probes have to be handled by the DVB-RCS module in the satellite hop. This is a basic segment, which is unaware of network QoS, but it has local QoS mechanisms and it can be suitably configured to deal with GRIP probes in the proper way. The satellite is only requested to respect the GRIP semantic, so enforcing a performance downgrade in the service experienced by probing packets (with respect to the DiffServ class the label carried by the probe belongs to). The figure also reports DVB-RCS signaling between satellite terminals and network control center.

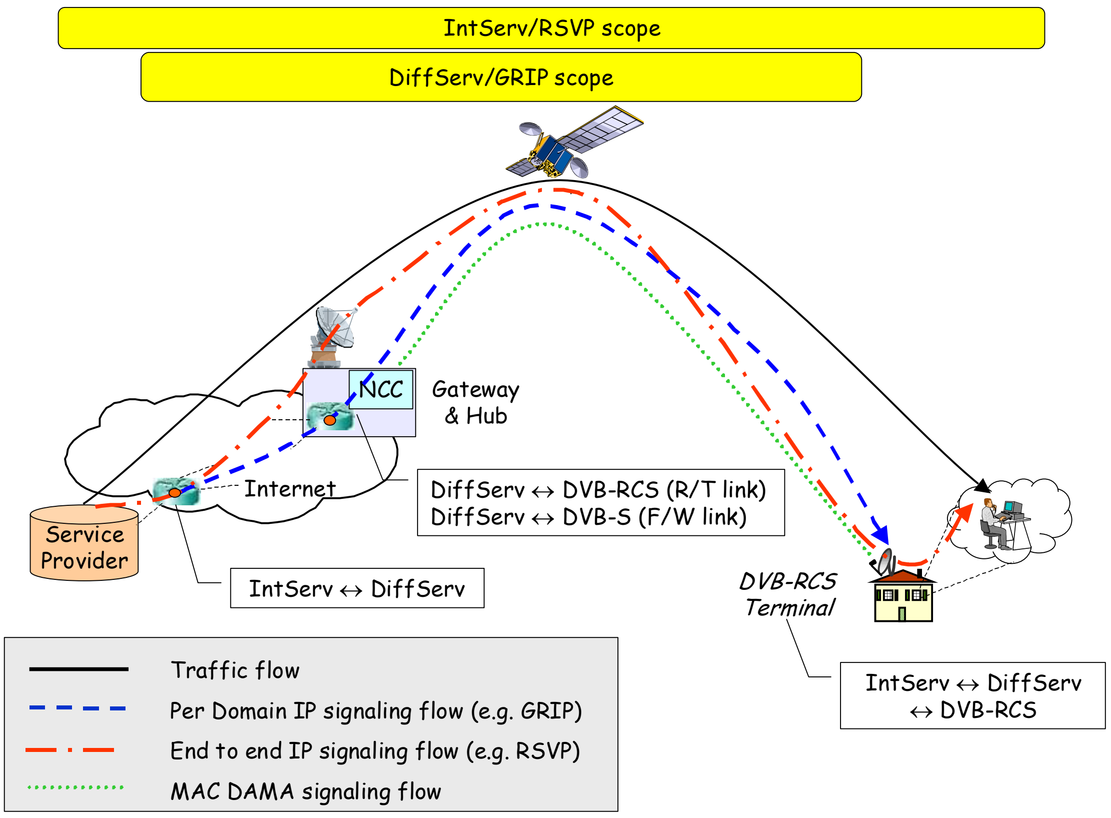

IntServ/RSVP scope
DiffServ/GRIP scope
NCC
Gateway & Hub
Internet
Service Provider
DiffServ ↔ DVB-RCS (R/T link)
DiffServ ↔ DVB-S (F/W link)
IntServ ↔ DiffServ
DVB-RCS Terminal
IntServ ↔ DiffServ ↔ DVB-RCS
Traffic flow
Per Domain IP signaling flow (e.g. GRIP)
End to end IP signaling flow (e.g. RSVP)
MAC DAMA signaling flow

figure 91 – The adopted end- to-end integrated scenario

More into details, the following considerations applies as to signaling messages,:

- end-to-end RSVP signaling is conveyed transparently in the payload and it is processed by the leaf routers only (one on the terrestrial side, which the service providers attaches to, the other one the satellite side, inside the DVB-RCS terminal);
- if the pure DiffServ QoS architecture is upgraded with GRIP support, per domain GRIP signaling is conveyed between the leaf router on the terrestrial side and the leaf router in the DVB-RCS terminal; this signaling is processed by these leaf routers and the router in the NCC, which operates similarly to a hub;
- the router in the hub maps between DiffServ/GRIP and DVB QoS;
- the router in the DVB-RCS terminal maps between i) RSVP and DiffServ/GRIP (cross-domain operation) and between ii) DiffServ/GRIP and DVB QoS (cross-layer operation);
- the NCC handles DVB-RCS MAC signaling for the on-demand assignment of capacity resources;
- satellite admission control is performed by the NCC in the hub on the basis of a measurement based solution;

- the mapping between RSVP and GRIP is performed by the ST as to end-to-end connection requests that ask to traverse the satellite hop in the satellite return channel direction (i.e., upstream from an ST to the NCC); conversely, in the opposite direction, RSVP enters the GRIP area in a common terrestrial router;
- a connection blocking service is provided by GRIP at the edge of the DiffServ are, hence, in the ST also;
- satellite connections, related to Layer 2 connection establishment, are dealt with by NCC, but they can be triggered by ST requests.

The figure above depicts the referenced scenario without pointing out to the specific on-board payload architecture. In fact, this issue is relevant in terms of the ST/NCC-to-satellite payload signaling to drive on-board QoS decisions, but it is not suitably depicted in the graphical form chosen in this section, which is instead oriented to stress the points of mapping/translation between QoS architectures and the types and boundaries of the signaling flows that traverse each single domain.

In the following of the section we report an overview of the QoS-related parameters of both the IntServ/RSVP model and DiffServ model and a brief presentation of the defined DVB-RCS network Profile Classes and the correspondent QoS parameters. Examples of mapping will be furthermore addressed in the following sections.

§6.1.1 IntServ service model

The IntServ/RSVP model includes two sorts of service targeted towards real time traffic applications: GS and CLS. In parallel with these two services the default BE service has to be considered for mapping purpose. GS is intended to support real time applications with tight delay requirements. The QoS provided by a GS implies: assured level of bandwidth (guaranteed throughput), mathematically bounded end-to-end delay (guaranteed maximum delay) and no queuing losses for conforming packets. The applications which require a GS specify both the traffic characteristics (TSpec) and reservation characteristics (RSpec); non conforming traffic is treated as BE. The CLS is intended to support a broad class of applications which have been developed for use in today's Internet but are highly sensitive to overload conditions. Important members of this service class are the adaptive real time applications with loose delay requirements. The QoS offered by a CLS is similar to those achievable by a BE service in an unloaded network. The CLS does not accept or make use of specific target values for control parameters such as delay or loss. The applications which require a CLS specify only the traffic characteristics (TSpec) while the reservation characteristics (RSpec) are not required; again, non conforming traffic is treated as BE.

§6.1.2 DiffServ service model

The DiffServ model presently defines three PHBs. The class selector PHB replaces the IP precedence field of the former TOS byte. DiffServ additionally offers two relative forwarding priorities:

- the EF PHB guarantees that packets will have a well-defined minimum departure rate which, if not exceeded, ensures that the associated queues are short or empty. EF is intended to support services that offer tightly bounded loss, delay and delay jitter;
- the AF PHB group offers different levels of forwarding assurances for packets belonging to an aggregated flow. Each AF group is independently allocated forwarding resources. Packets are marked with one of three drop precedences, the ones with highest drop precedence are dropped with greater probability than those marked with the lowest drop precedence. DSCPs are recommended for four independent AF groups, although a DiffServ domain can have more or fewer AF groups.

Eventually, the BE level is also considered as a default PHB with no priority at all (BE PHB).

§6.1.3 DVB-RCS service model

Traffic offered to a DVB-RCS is transported accordingly to specific Profile Classes (PCs) that perform differently as to delay and loss characteristics. Actually, this is the only QoS support offered by a DVB-RCS segment in addition to the admission control functions. The different PCs are meant to provide significantly different levels of service, and the envisioned applications range from e-mail to tight real time video and voice applications.

In particular, three QoS parameters are considered, and six PCs are identified for the DVB-RCS satellite network on the basis of significant combinations of the QoS parameters. The parameters are:

- the maximum packet transfer delay (MPTD), computed as the time due to move a packet between the extremities of the satellite link;
- the peak-to-peak delay variation (PtPDV), expressed as the maximum jitter experienced in the satellite link;
- the packet loss ratio (PLR).

Profile Class	MPTD (One Way)	PtPDV	PLR	Application example
1	highly sensitive (some hundreds ms)	highly sensitive (some tens ms)	Loosely sensitive ($\leq 10^{-3}$)	Voice-based
2	Sensitive (≤ 1 sec)	highly sensitive (some tens ms)	Sensitive ($\leq 10^{-4}$)	Real time TV-cast, Interactive TV
3	Sensitive (≤ 2 sec.)	Loosely or not sensitive	highly sensitive ($\leq 10^{-6}$)	Real Time Transaction Data
4	loosely sensitive (few seconds)	not sensitive (no upper bound)	Sensitive ($\leq 10^{-4}$)	Web Browsing, Interactive Games
5	loosely sensitive (some seconds)	not sensitive (no upper bound)	highly sensitive ($\leq 10^{-6}$)	File transfer
6	not sensitive (no upper bound)	not sensitive (no upper bound)	not sensitive (no upper bound)	e-mail, Fax

table 9 - DVB-RCS network Profile Classes

The six Profile Classes are characterized in table 9 following an approach related to the sensitiveness of each PC to the QoS parameters. Of course, delay are accounted for a duration of the order of hundreds of milliseconds or also seconds, since the adopted satellite is in a geostationary orbit, and it takes about 120 milliseconds to traverse the radio channel.

There are basically two high performance Profile Classes (PC1 and PC2) which are particularly suited for real time applications offering a great amount of traffic. PC number 3 offers intermediate performance for real time applications and interactive applications requiring a small amount of data to be transferred, but which need higher protection against packet loss. Pleasurable support for web surfing is offered by PC4, as well as for instant messaging applications with slow rate video and interactive games. PC5 is definitely not suited for interactions and real time needs, but it suits for a *better that best effort* service, useful for file transfer or slow rate Internet browsing. A BE service is actually provided by the PC6, without any kind of guarantees and bound for QoS parameters.

§6.2 QoS-aware interworking of DiffServ, IntServ and DVB system

This section is devoted to the mapping between QoS service model presented in the section above. Let’s consider the occurrence that an IP architecture endowed with QoS-aware mechanisms includes a satellite hop. With reference to the IP architecture envisioned in this thesis and the location of the satellite hop in the network, the issues to be addressed are essentially three: i) how to map IntServ services and RSVP on DVB-RCS Profile Classes; ii) how to map DiffServ classes and GRIP on DVB-RCS Profile Classes; iii) how to support network signaling extensions in accordance with rules defined in section §4.1 and therein reported sub-sections. The first point occurs when the satellite hop is under the direct control of the IntServ domain. The second point refers to a scenario in which the DiffServ sub-network also exploits the capabilities offered by the DVB-RCS connectivity. This point also applies to the integration scenario proposed in section §6.1, as long as the satellite segment is embedded in the DiffServ domain. The third issue, again, applies to the previously reported scenario. In fact, in section §6.1 we envisioned an end-to-end integrated scenario with IntServ at both network edges and DiffServ in the middle, covering the satellite area and much more. However, it is worth noting that the need for signaling extensions for GRIP and RSVP (or similar approaches) also requires the knowledge of mapping techniques for both IntServ to DVB-RCS and DiffServ to DVB-RCS QoS features, as soon as the implicit signaling mechanism coming into play (i.e., GRIP) is in charge of managing the satellite link. In fact, it could be hopeful to harmonize the QoS behavior along with the overall network, so that the DiffServ features have to be adapted to both IntServ services and DVB-RCS capabilities. Additionally, as a matter of fact, the DiffServ on DVB-RCS mapping should be driven by the hypothetical direct IntServ on DVB-RCS mapping that would take place in a all-IntServ network, instead of providing abstract service classes uncorrelated with the service running in its surrounding areas. Recall that DiffServ with GRIP capabilities for signaling extensions does not operate for itself, but it provides flexible means to extend the scope of some other powerful (but neither flexible, scalable nor stateless) protocols.

In definitive, as to our intents, the third issues listed above implicitly requires to preliminarily address the remaining two issues. That is the reason why we have selected exactly that scenario.

§6.2.1 QoS-related mapping between IntServ/RSVP and DVB-RCS

In this section, mappings in terms of Profile Classes (including QoS parameters) and traffic parameters between IntServ/RSVP service model and DVB-RCS service model are provided. There are two possible mappings for GS: Profile Class 1 and Profile Class 2, because of the real time support required by GS. In DVB-RCS domain, the main difference between Profile Class 1 and Profile Class 2 is the maximum packet transfer delay value assured by DVB-RCS bearer service, that is some hundred of *ms* for Profile Class 1 targeted towards conversational applications and less than 1 *s* for Profile Class 2 targeted towards streaming and interactive TV services. Consequently a GS is proposed to be mapped into the DVB-RCS Profile Class 1 or Profile Class 2 according to the requested maximum packet transfer delay: if the requested maximum packet transfer delay is in the order of some hundreds of *ms*, then the GS is mapped into the Profile Class 1, otherwise if the requested maximum packet transfer delay is less than 1 *s* the GS is mapped into the Profile Class 2. Anyway, PC1 could be enough for every GS services. Conversely, Profile Classes from 3 to 6 are not applicable matches for GS. These classes do not provide stringent delay bounds and cannot guarantee consistently low delay for every packet.

Guaranteed Service is mapped into DVB-RCS Profile Class 1 and Profile Class 2 which require that only PDR traffic parameter is specified. The latter will result from the following RSVP/DVB-RCS mapping formula:

$$(41) \qquad \boldsymbol{PDR_{DVB\text{-}RCS}} = \begin{cases} \text{Rate } R\text{, if the DVB-RCS ingress point has a burst buffer} \\ \\ \text{Max } (R, p)\text{, with little or no burst buffering}^{33} \end{cases}$$

where *R* is the rate specified in the RSpec parameters and selected to obtain bandwidth and delay guarantees and *p* is the Peak Data Rate specified in the TSpec parameters.

There are three possible mappings for CLS: Profile Class 3, Profile Class 4 and Profile Class 5. In DVB-RCS domain, the main difference between Profile Class 3 and Profile Class 4 and 5 is the maximum packet transfer delay requirement which is specified (even if it is not stringent) in the Profile Class 3 and not specified in Profile Class 4 and 5. Considering that the applications (with loose delay requirements but which are highly sensitive to overload conditions) which require a CLS specify only the traffic characteristics (TSpec) while the reservation characteristics (RSpec) are not required, the choice among Profile Class 3, 4 and 5 can be made on the basis of the requested TSpec, the available resources or according to the predefined correspondence rule which associates classes of applications to the Profile Classes. The Profile Class 6 does not provide enough capability for CLS as it does not envisage any delay or loss requirement, therefore it is used only to support BE traffic.

[33] The burst buffering refers to the token bucket burst parameter *b* specified in the TSpec parameters. The requirement - little or no burst buffering - resemble the zero-buffer case.

Controlled Load Service is mapped into DVB-RCS Profile Class 3, Profile Class 4 and Profile Class 5 which require that PDR, SDR and MBS traffic parameters are specified. These traffic parameters will result from the following RSVP/DVB-RCS mapping formulas:

$\mathbf{PDR_{DVB\text{-}RCS}} = p$

$\mathbf{SDR_{DVB\text{-}RCS}} = r$

$\mathbf{MBS_{DVB\text{-}RCS}} = b$

where p is the Peak Data Rate, r is the Sustainable Rate and b is the Token Bucket Size specified in the TSpec parameters.

Eventually, table 10 summarizes the proposed mapping between IntServ/RSVP classes and DVB-RCS Profile Classes. It can be noted that, even if all of the Profile Classes have the capability to carry Best Effort service, the natural Profile Class is the DVB-RCS Profile Class 6 which does not envisage any delay or loss bound. As to BE services, there is no traffic description and a default value is typically set by the network operator.

	IP Classes	**DVB-RCS Profile Classes**					
		1	**2**	**3**	**4**	**5**	**6**
IntServ Classes	**GS**	√	√				
	CLS			√	√	√	
	BE						√

table 10 - Mapping of IntServ/RSVP classes to DVB-RCS Profile Classes

§6.2.2 QoS-related mapping between DiffServ/GRIP and DVB-RCS

In this section, guidelines to mapping in terms of Profile Classes (including QoS parameters) and traffic parameters between DiffServ/GRIP service model and DVB-RCS service model are provided. As to traffic parameter mapping, it is worth noting that in a DiffServ domain, only traffic aggregates can be managed. As a consequence, predefined traffic parameters have to be considered, and the network operator furnishes default values. One solution consists in distributing the available bandwidth between DiffServ classes, following a predefined percentage repartition. This repartition depends on the number and the types of implemented DiffServ classes. In alternative, once a new connection has to be established, each terminal can use a different, but locally predefined, set of connection parameters. Subsequent parameter renegotiation should be allowed. In every cases, the only required parameter to signals is the bandwidth, and a one-to-one relationship is set between a DSCP and a DVB-RCS PC: a DSCP corresponds to one and only one PC.

EF PHB should be considered the equivalent of GS in IntServ. As a consequence, there are two possible mappings for EF PHB: Profile Class 1 and Profile Class 2, because of the real time support and stringent delay requirements. as already seen, in DVB-RCS domain, the main difference between Profile Class 1 and Profile Class 2 is that the first allows to obtain few hundreds of *ms* as to maximum packet transfer delay, while using the second a delay smaller than 1 *s* is guaranteed. Since in DiffServ is not allowed to distinguish between EF streams, we propose to map EF PHB on Profile Class 1. This adoption is also suitable, since it leaves more room to AF mapping, consisting of many different QoS levels.

AF PHB corresponds to CLS: one can think to use Profile Class 3, Profile Class 4 and Profile Class 5, plus Profile Class 2 not used by EF. In turn, this leaves room for a level differentiation between up to four AF classes, even though up to three levels inside each AF class should be required. This implies that up to twelve PC values should be needed instead of the four available values. Hence, two approaches, or a combination of both of them, are envisioned:

- operate a superposition of DSCPs over corresponding PCs;
- reduce the number of DSCP in use in the DiffServ domain.

Note that an AF aggregate could be mapped on a single PC, since internal AF differentiation occurs at network layer. However, a further differentiation at MAC layer could be useful, especially for implicit signaling purposes.

A further mapping between IntServ/RSVP and DiffServ is needed to specify the AFx class to be used in accordance with the IntServ operation in adjacent domains. In particular, one can determine the DiffServ-IntServ class correspondences off-line, using the following rules:

- map RSVP messages on EF PHB (or BE, but not AF, which requires a separate semantic);
- determine DVB-RCS PC from IntServ/RSVP to DVB-RCS mapping table, obtaining a given PC, say PCx;
- look-up at DiffServ to DVB-RCS mapping table and return nearest DSCP between EF, AF11, AF21, AF31, AF41, and BE; in this way a DiffServ class is detected between EF, AF1, AF2, AF3, AF4, and BE.

IntServ BE has to be mapped on DiffServ BE. In turn, even if all of the PCs could carry Best Effort service, the natural Profile Class for BE PHB is the DVB-RCS Profile Class 6 which does not envisage any delay or loss bound. Moreover, as for BE services, there is no traffic description and a default value is typically set by the network operator.

A "Terminal Interworking and Coordination function", proper of the ST, determines the IP QoS level of an incoming flow by analyzing the information contained in the DSCP field. After processing of the DSCP field, the ST knows the type pf service requested by a new flow and it may determine a suitable associated Profile Class. Some exemplificative DiffServ mapping are proposed in this paragraph. In table 11 and table 12 we take into account every possible AFxy DSCP, plus EF and BE DSCPs. Note that a single AF class is mapped onto two PCs: one for AFx1 and one for both AFx2 and AFx3. Using such a mapping, the network is provided for supporting:

- delay sensitive, low loss requiring streams in AF1 class;
- slightly delay sensitive, very low loss requiring streams in AF2 class;
- loosely delay sensitive, low loss requiring streams in AF3 class;
- very loosely delay sensitive, very low loss requiring streams in AF4 class.

The difference between mappings presented in table 11 and table 12 is that, in the latter table, AFx3 are not taken into account. Moreover, in table 13 an alternative mapping is proposed, and only two AF classes are dealt with:

DiffServ Classes		**DVB-RCS Profile Classes**					
Class	**Prec.**	**1**	**2**	**3**	**4**	**5**	**6**
EF		√					
AF1	**1**		√				
AF1	**2**			√			
AF1	**3**			√			
AF2	**1**			√			
AF2	**2**				√		
AF2	**3**				√		
AF3	**1**				√		
AF3	**2**					√	
AF3	**3**					√	
AF4	**1**					√	
AF4	**2**						√
AF4	**3**						√
BE							√

table 11 - A possible mapping of DiffServ classes to DVB-RCS Profile Classes

- very delay sensitive, low loss requiring streams in AF1 class;
- loosely delay sensitive, very low loss requiring streams in AF2 class;

As an example of relation between IntServ services and DiffServ classes, let's consider the mapping proposed in table 12 and a CLS service mapped as in table 10. There are three possible mappings onto PC3, PC4 or PC5. Suppose that a stream in CLS is not delay sensitive, but it is loss sensitive, since PC4 is selected. In table 12, AF22 and AF31 DSCPs correspond to PC4, thus AF3 class is suitable for carrying data pertaining to that specific stream. In conclusion, data will be carried as AF31 mapped onto PC4, while AF32 probes will be used (when needed) mapped on PC5. On the other hand, RSVP signaling, when to be forwarded transparently (see subsection §4.1.4), will be mapped onto EF PHB and PC1.

DiffServ Classes		**DVB-RCS Profile Classes**					
Class	**Prec.**	**1**	**2**	**3**	**4**	**5**	**6**
EF		√					
AF1	**Data/Ack**		√				
AF1	**Probe**			√			
AF2	**Data/Ack**			√			
AF2	**Probe**				√		
AF3	**Data/Ack**				√		
AF3	**Probe**					√	
AF4	**Data/Ack**					√	
AF4	**Probe**						√
BE							√

table 12 - A first alternative mapping of DiffServ/GRIP classes to DVB-RCS Profile Classes

DiffServ Classes		**DVB-RCS Profile Classes**					
Class	**Prec.**	**1**	**2**	**3**	**4**	**5**	**6**
EF		√					
AF1	**Data/Ack**		√				
AF1	**Probe**			√			
AF2	**Data/Ack**				√		
AF2	**Probe**					√	
BE							√

table 13 – An additional alternative mapping of DiffServ/GRIP classes to DVB-RCS Profile Classes, using a minimal DiffServ set

§6.2.3 Interoperation of DiffServ and DVB-RCS satellite networks

In this section we show Satellite Terminals, implementing DVB-RCS standards, and Satellite Gateway, including Hub and NCC, dealing with IP packets when the end-to-end integrated scenario of figure 91 is considered. The description is carried out using and commenting the flow chart diagrams reported in the following pages. DiffServ, GRIP and eventually RSVP/GRIP mapping are solely considered here.

Firstly, for the sake of clarity we recall some preliminary terms. By definition, given a ST (i.e., a DVB-RCS earth station), with the term *outgoing IP traffic* we will refer to IP datagrams leaving the ST and routed towards the satellite network (to be carried over satellite links), whereas with the term *incoming IP traffic* we will refer to IP datagrams entering ST and coming from the satellite network (carried over satellite links). Moreover, given a ST we will label as *IP layer* the Internet network layer interfacing terrestrial networks, whereas towards the satellite network a peer *S-IP layer* will enclose the necessary IP interworking functions dealing with satellite issues. GRIP interworking functions are managed by a *GRIP interworking layer* logically overlying both IP and S-IP layer, while DiffServ functions remains in the scope of Layer 3 (as well as IP and S-IP).

§6.2.3.1 Gateway/hub/NCC side

In this scope *outgoing* datagrams are datagrams directed to the satellite, while *incoming* datagrams are leaving the satellite air interface to enter the satellite terminal. As to the gateway terminal (a hub with a NCC aboard) the provided flow chart diagrams will show the logical operations flow for the handling of:

- Outgoing IP datagrams, classified as:
  - Outgoing GRIP probes,
  - Outgoing AFx1 and AFx3 datagrams,
  - Outgoing EF and BE datagrams.
- Incoming IP datagrams, classified as:
  - Incoming GRIP probes,
  - Incoming AFx1 and AFx3 datagrams,
  - Incoming EF and BE datagrams.

The provided diagrams are valid in the in the frame of the following hypotheses:

1. only point-to-point unidirectional terminal initiated connection control procedures will be taken into account;
2. only the connection setup phase of GRIP sessions and satellite connections will be considered; a complete finite state machine is necessary to describe the overall lifecycle of a satellite connection at Layer 2;

3. GRIP interworking function receives reassembled datagrams from the Layer 2;
4. for the sake if simplicity, we assume that once a Layer 2 connection is opened, no needs of parameters re-negotiation arises;
5. the RSVP/GRIP interworking function adopts the operation described in §4.1.4.2 (RSVP plus GRIP), i.e., the signaling extension is performed on the basis of the information conveyed by the RSVP Resv message.

Firstly, figure 92 to figure 98 illustrate the IP flow handling related to IP traffic pertaining to the satellite hub. In figure 92 it is shown that datagrams are classified in: i) S-IP datagrams, that is traffic offered from the satellite to the hub, and ii) IP datagrams, coming from the terrestrial link. IP and S-IP datagrams are classified by means of the SAP where datagrams came from. Both for outgoing and incoming datagrams, a DiffServ classification is operated in order to separate AF datagrams and other (EF and BE) traffic. Finally, AF traffic is further decomposed in GRIP probes (labeled as AFx2) and any other AF traffic (labeled as AFx1 and AFx3). The overall classification is performed by means of a single parameter, that is the DSCP label in IP datagrams. The rationale of such a classification come from the adopted end-to-end integrated scenario. In fact, the satellite hub represents a node in the core of a DiffServ domain, so that nor GRIP sources neither GRIP destination endpoints have to run over satellite hub, but the local GRIP admission control criterion has to be enforced. In other words, the satellite hub acts like a gate for GRIP probes, and the state of that gate (accept or reject) has to be computed based on the traffic measures locally performed. Thus the hub has to distinguish between three kind of datagrams:

- AFx2 datagrams, carrying GRIP probes and triggering MBAC;
- AFx1 and AFx3 datagrams, to be forwarded in the satellite air interface only after a AFx2 GRIP probing has been positively acknowledged;
- every other IP traffic (EF and BE DiffServ datagrams).

The gate behavior is shown in figure 93, as to the outgoing datagrams, and in figure 96, as to the incoming datagrams. In figure 93, after an outgoing GRIP probe is received, the hub checks whether the destination subnet is connected with an active AFx class or not:

- if yes, a measurement of the traffic generated in that class is taken, and the local GRIP admission control is run. At this moment:
    - the probe is accepted and forwarded, or
    - the probe is dropped.

- if not, a connection admission control (Layer 3) and an eventual setup procedures (Layer 2) are performed in order to setup AFx1, AFx2 and AFx3 connections towards the destination subnet.
    - In case of positive CAC outcome, the probe is forwarded;
    - In case of negative CAC outcome, the probe is discarded.

Note that the DVB-related CAC phase (i.e., the procedure to admit a ST to join the satellite network) has to be performed only the first time a probe has to be forwarded to a specific sub-network, since an AFx relation has a pre-determined amount of resource to be allotted in the output interface of a core DiffServ router. In fact, in this exemplificative approach, there is no need to resize the resources allotted to an AF connection. The DVB-related CAC has not to be confused with the per-flow based GRIP CAC procedure to be performed for each new flow in a given AF class.

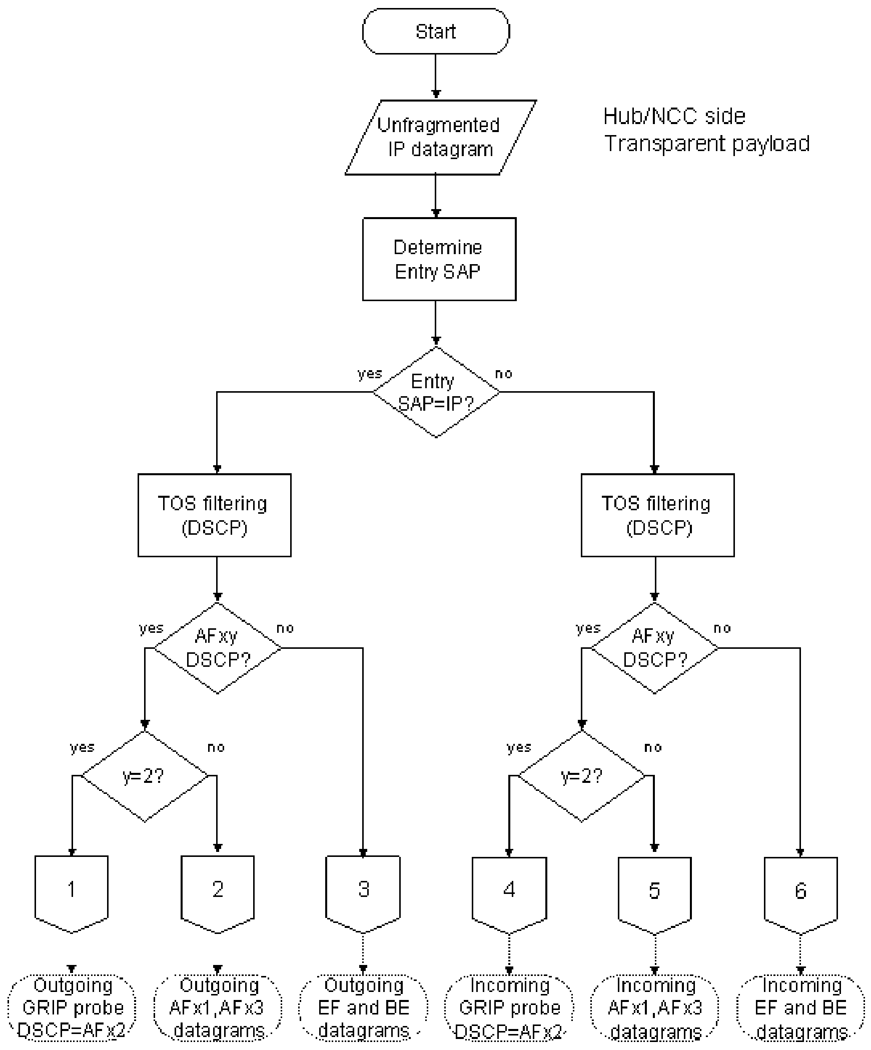


figure 92 - IP packet flow handling at Hub

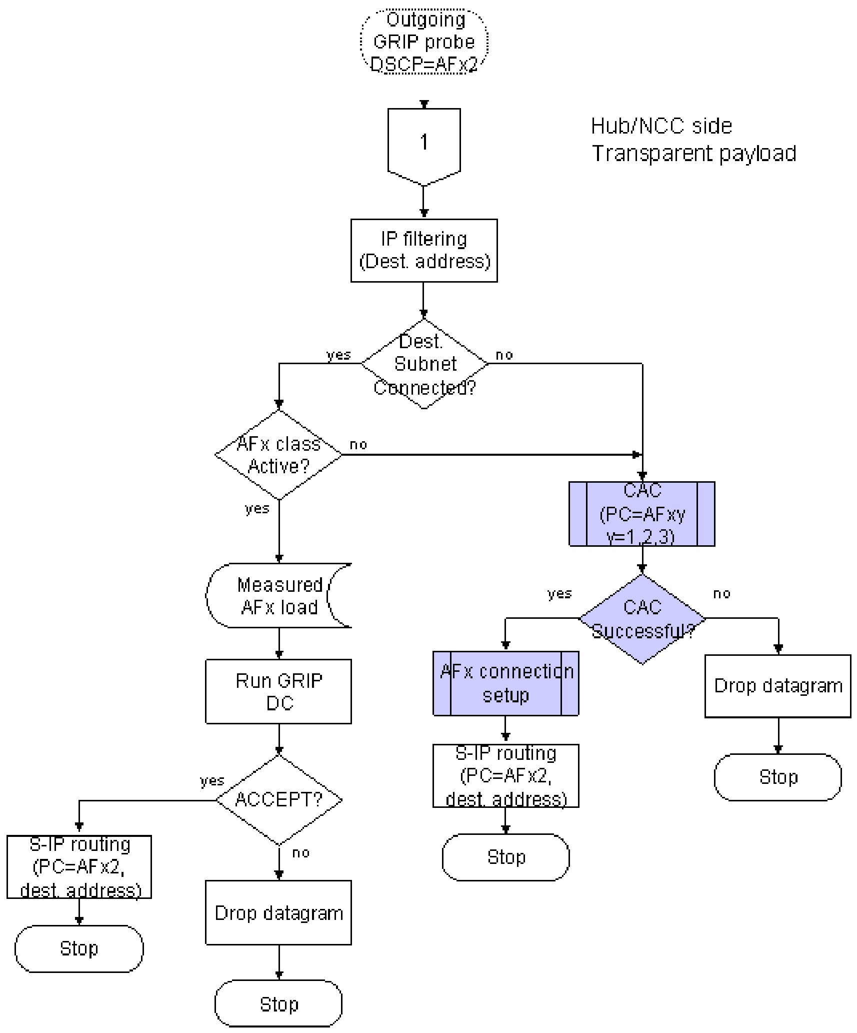
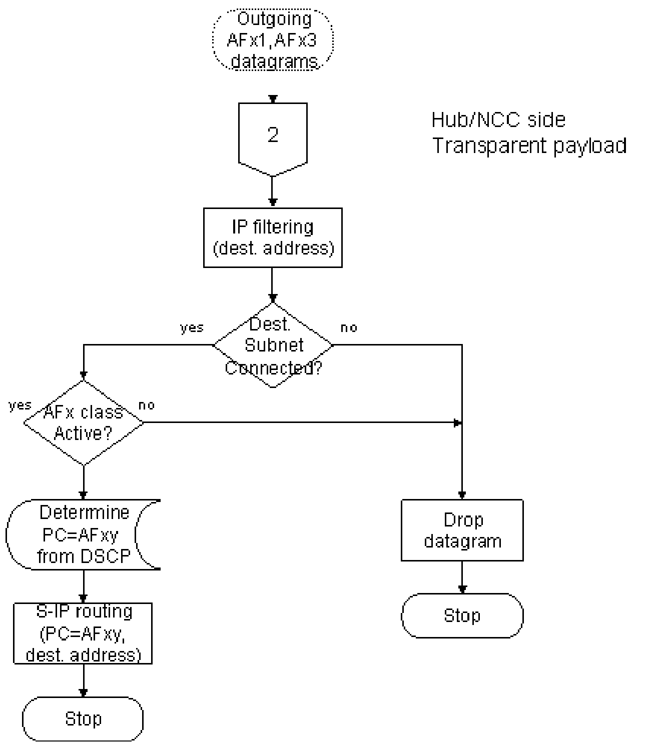


figure 93 - IP packet flow handling at Hub – GRIP probes to be sent to satellite

AFx1 and AFx3 outgoing datagrams are handled following the logic operation depicted in figure 94: firstly, the destination subnet is determined, and an eventually check is performed about the existence of an open AFx connection (that is a triple AFx1, AFx2, and AFx3 connection at Layer 2). Negative responses cause the datagram dropping, otherwise datagrams are forwarded to the satellite.

A difference can be noticed between AFx1/AFx3 and EF/BE outgoing-traffic handling. In figure 95, after a preliminary DVB PC computation, the same checking as above is performed over destination subnet connected with the determined PC. Where positive outcome is given, datagrams are forwarded; but if a connection already set is not available, the hub tries to open a new one, instead of simply dropping datagrams. As already mentioned when describing the AFx setup procedure, EF and BE connection need to be open only once per destination subnet, and no resource reallocation is needed. In figure 96, a simpler GRIP probe handling is shown as regards the incoming datagrams.



Outgoing AFx1,AFx3 datagrams
2
Hub/NCC side
Transparent payload
IP filtering (dest. address)
yes
Dest. Subnet Connected?
no
yes
AFx class Active?
no
Determine PC=AFxy from DSCP
Drop datagram
Stop
S-IP routing (PC=AFxy, dest. address)
Stop

figure 94 - IP packet flow handling at Hub – Other outgoing AF datagrams

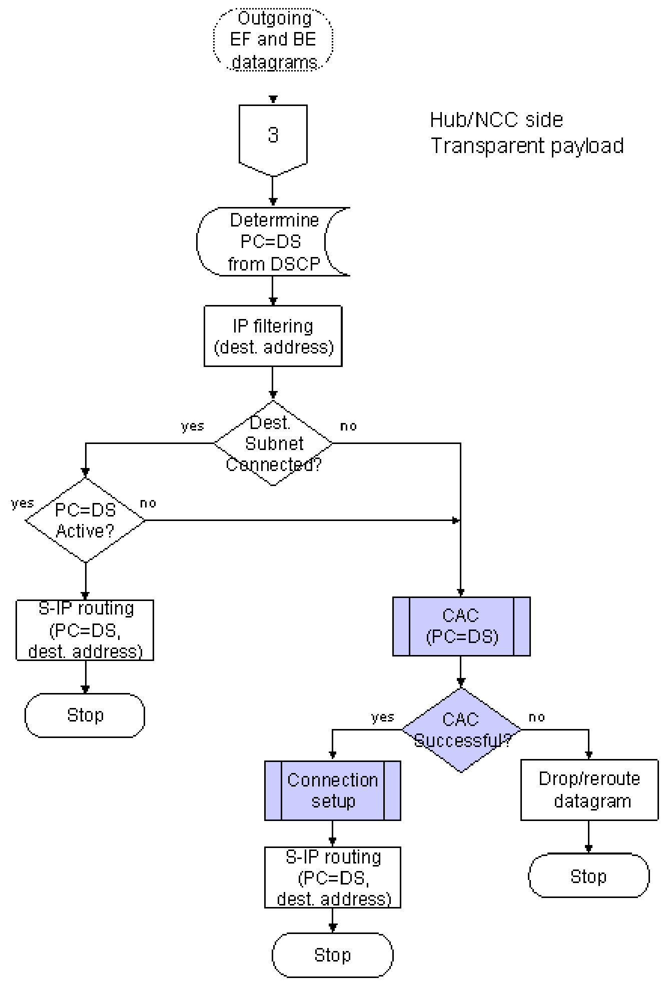


figure 95 - IP packet flow handling at Hub – EF and BE traffic going to satellite

Moreover, when an AFx connection is still active in the return link, a traffic measurement is taken over the incoming AFx traffic and a GRIP measurement based CAC is performed, affecting the behavior experienced by AFx2 probes in the upstream direction. However, if no open AFx-enabled Layer 2 connections are available in the return link, it has to open such a connection in order to accommodate GRIP acks in the return path. If this procedure fails, GRIP probe should be dropped, since no acknowledgement will be allowed.

Finally figure 97 and figure 98 report IP forwarding of AFx1, AFx3, EF and BE traffic coming from satellite and to be forwarded over a terrestrial DiffServ link, without any further action to be taken.

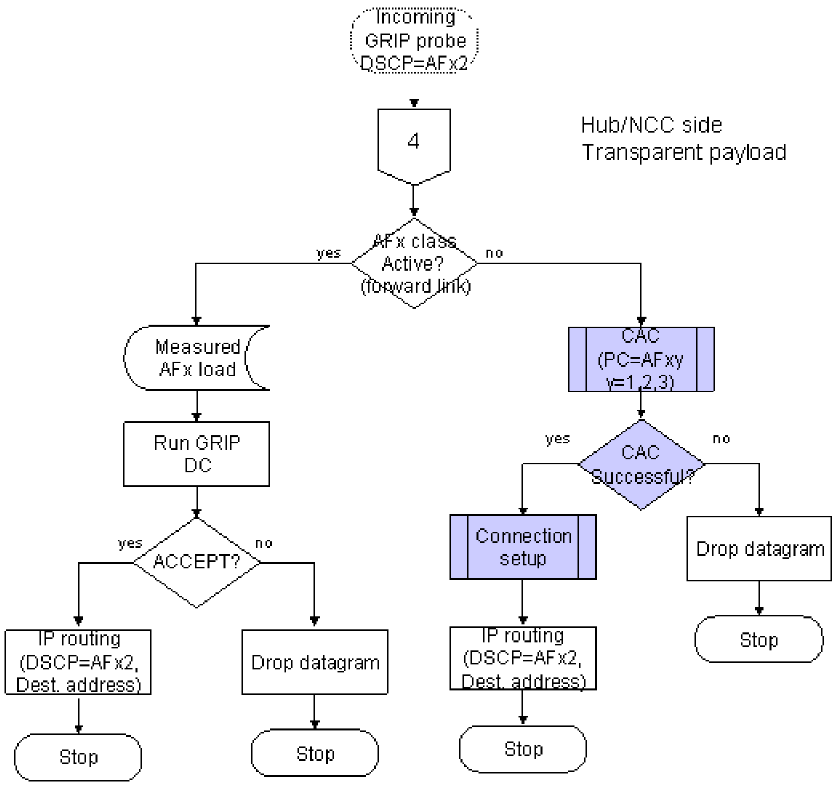


figure 96 - IP packet flow handling at Hub – Received GRIP probes

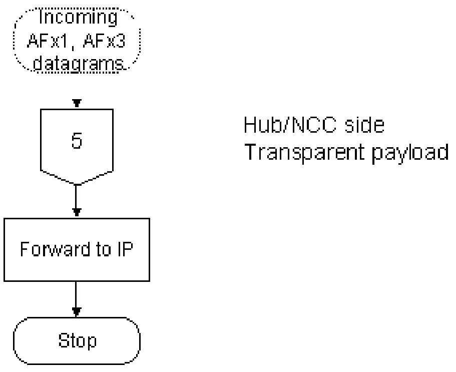


figure 97 – IP packet flow handling at Hub – AF non-GRIP- probes leaving satellite area

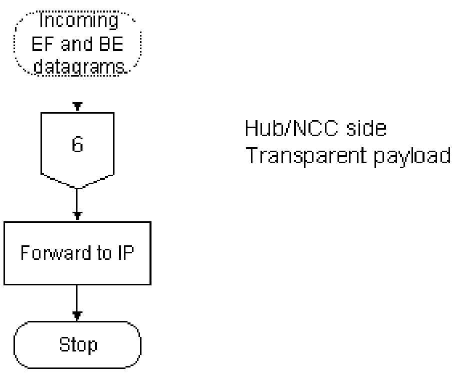


figure 98 – IP packet flow handling at Hub – EF and BE traffic leaving satellite area

§6.2.3.2 Satellite Terminal side

Now we consider a generic DVB-RCS satellite terminal working in the same scenario as above for the satellite hub/NCC. Again, recall that in this scope *outgoing* datagrams are datagrams directed to the satellite, while *incoming* datagrams are leaving the satellite air interface to enter the ST. Moreover, the entire DiffServ domain proposed in the scenario is accessible from the terminal side only through the satellite gateway. As a consequence, a satellite terminal has to connect with only one destination satellite node (the hub/gateway). On the contrary, a gateway should establish multiple connections with other satellite terminals. The provided flow chart diagrams will show the logical operations flow for the handling of:

- Outgoing IP datagrams:
  - Outgoing AF datagrams,
  - Outgoing EF and BE datagrams.
- Incoming IP datagrams:
  - Incoming RSVP Resv messages,
  - Incoming EF and BE datagrams,
  - Incoming GRIP acks,
  - Incoming GRIP probes.

The provided diagrams are valid in the in the frame of the following hypotheses:

1. only point-to-point unidirectional terminal initiated connection control procedures will be taken into account;
2. only the connection setup phase of GRIP sessions and Layer 2 satellite connection will be considered; a complete finite state machine is necessary to describe the complete lifecycle of a satellite connection;

3. diagrams refer to the owner side of a DVB-RCS link, that is they specify how the aforementioned kinds of datagrams have to be filtered and handled by the interworking functions of the terminal which controls connections the diagram pertains to;
4. the satellite terminal logon phase is assumed to be already performed;
5. the satellite terminal is provided a Probing Table, maintaining a list of user connections attempting to perform GRIP admission control and waiting for a GRIP ack; a separate process will be in charge of timely refresh such a table when GRIP timeouts exhaust;
6. reassembly of IP datagrams is assumed to be already performed at Layer 3 before the GRIP interworking function process the datagram;
7. for the sake if simplicity, we assume that once a Layer 2 connection is opened, no needs of parameters re-negotiation arises;
8. the RSVP/GRIP interworking function adopts the operation described in §4.1.4.2 (RSVP plus GRIP), i.e., the signaling extension is performed on the basis of the information conveyed by the RSVP Resv message.

In this scenario a ST is located at the edge of a DiffServ domain, so that the ST is in charge of performing IntServ/RSVP to DiffServ/GRIP mapping and vice versa. In this scope, a ST has to parse both GRIP messages and RSVP signaling crossing the border between DiffServ and IntServ domains. Furthermore, GRIP sender and receiver endpoint operation plus the local decision criterion for CAC purposes have to be implemented on-board of such a terminal.

The flow charts in figure 99 to figure 105 propose the IP flow handling related to IP traffic pertaining to a ST. Furthermore, figure 106 and figure 107 introduce two procedures called by the flow handling schemes. As to the logic operation to be performed at satellite terminal side, figure 99 shows that IP and S-IP datagrams are separated by means of the Service Access Point (SAP), and a following classification is also performed both in incoming and outgoing branches of the figure. As to outgoing datagrams, the first action taken by the satellite terminal consists of a mapping from IntServ to DiffServ classes, accordingly to a predefined SLA adopted at the edge of IntServ and DiffServ domains. Eventually, outgoing datagrams are classified in AF datagrams and EF/BE datagrams. On the right side of the flow chart diagram, the branch dealing with incoming traffic operates a first classification based on a DSCP filtering and AF datagrams are set apart from non-AF datagrams (BE and EF). AFx2 datagrams are finally separated from other AF traffic, while non-AF traffic is parsed in order to look for embedded RSVP Resv messages, to be also separated from any other datagrams.

The rationale of such a classification come from the adopted scenario. In fact, the satellite terminal represents an edge-node in the DiffServ domain, so that GRIP session initiator and the GRIP receiver entities have to be activated over satellite terminal. Furthermore, since the terminal is located at the edge of a DiffServ and an IntServ domain, semantic IntServ/DiffServ translations have to be managed and performed in both directions, while RSVP Resv messages have to be parsed in order to trigger GRIP admission control in the DiffServ domain.

In figure 100 outgoing AF datagrams are shown: the AFx class activation is checked and,

- in presence of a pre-established AFx connection at Layer 2, the right DVB PC is computed and datagrams are forwarded towards the satellite;
- if no AFx connections are available at Layer 2, datagrams are dropped.

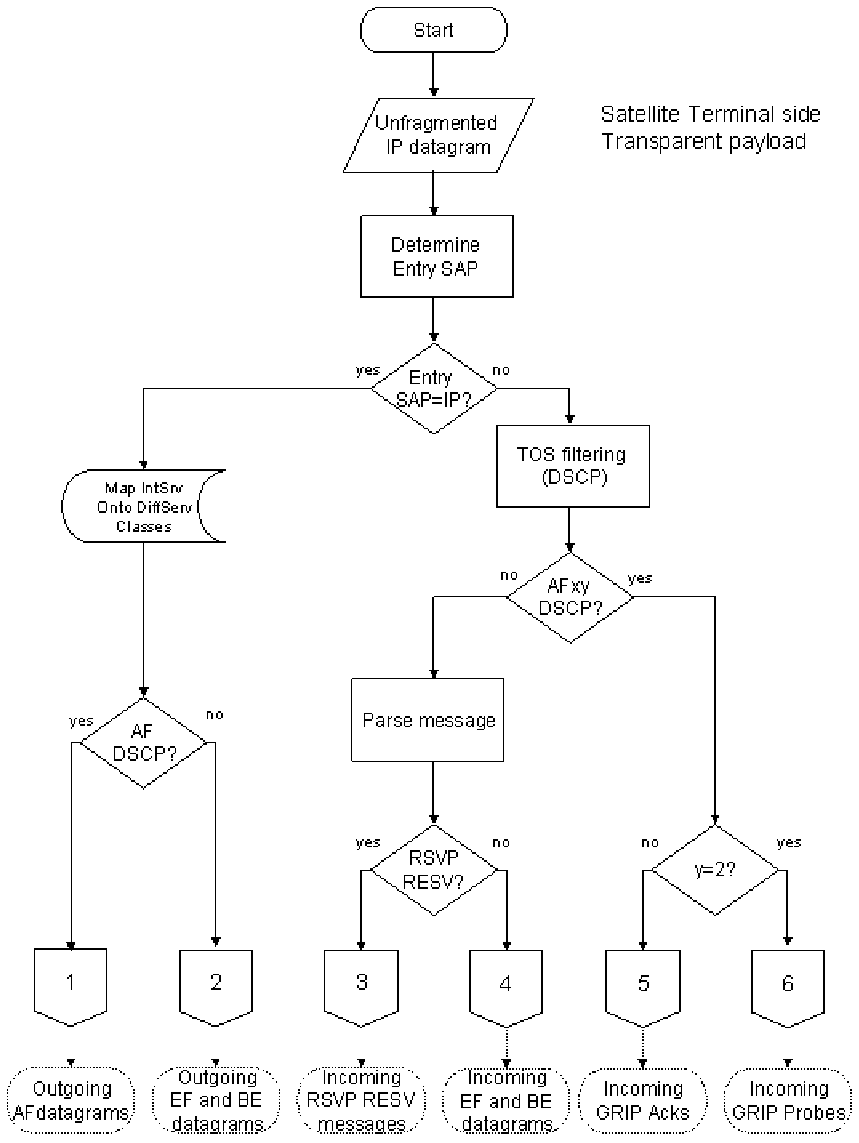


figure 99 - IP packet flow handling at Satellite Terminal

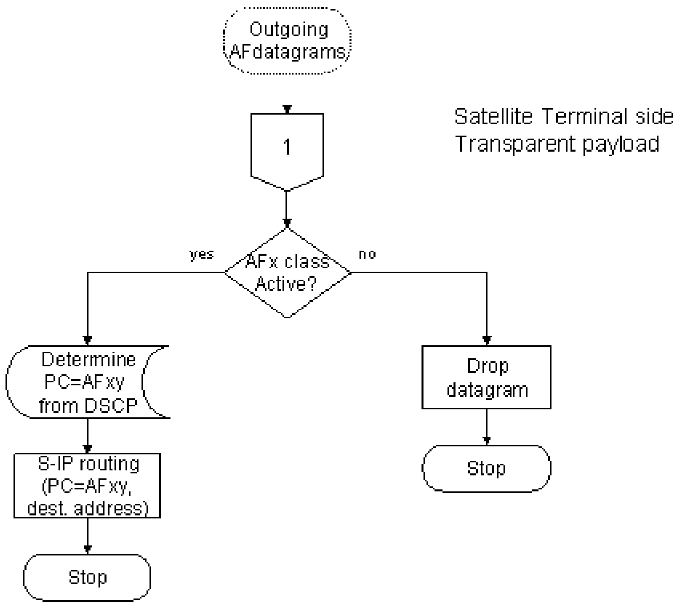

Outgoing AFdatagrams
Satellite Terminal side Transparent payload
1
yes
AFx class Active?
no
Determine PC=AFxy from DSCP
Drop datagram
S-IP routing (PC=AFxy, dest. address)
Stop
Stop

figure 100 - IP packet flow handling at Satellite Terminal – AF traffic entering satellite

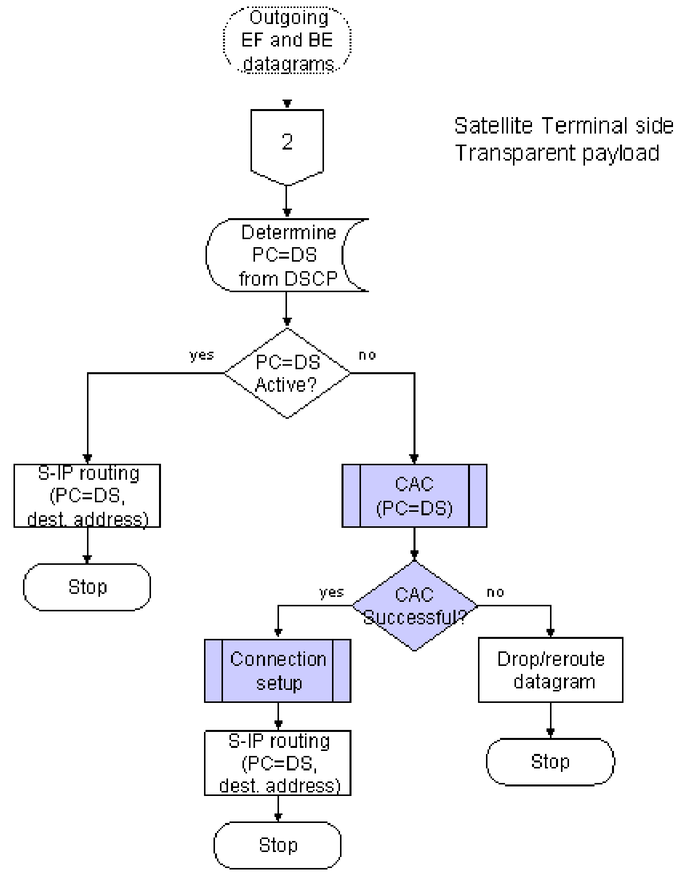

Outgoing EF and BE datagrams
Satellite Terminal side Transparent payload
2
Determine PC=DS from DSCP
yes
PC=DS Active?
no
S-IP routing (PC=DS, dest. address)
CAC (PC=DS)
Stop
yes
CAC Successful?
no
Connection setup
Drop/reroute datagram
S-IP routing (PC=DS, dest. address)
Stop
Stop

figure 101 - IP packet flow handling at Satellite Terminal– EF and BE traffic entering satellite

Note that a generic AFx packet is not able to trigger an AFx class setup connection at Layer 2. On the contrary, EF and BE outgoing datagrams are firstly mapped on their specific DVB PCs, as depicted in figure 101, and then:

- if a Layer 2 connection is ready to transport EF/BE datagrams, those are forwarded;
- if a Layer 2 connection is not available, the satellite terminal is in charge of attempting to open the desired connection, with the appropriate PC value;
    - o if the connection is accepted, datagrams are forwarded;
    - o otherwise datagrams are dropped or re-routed if any default terrestrial link is available.

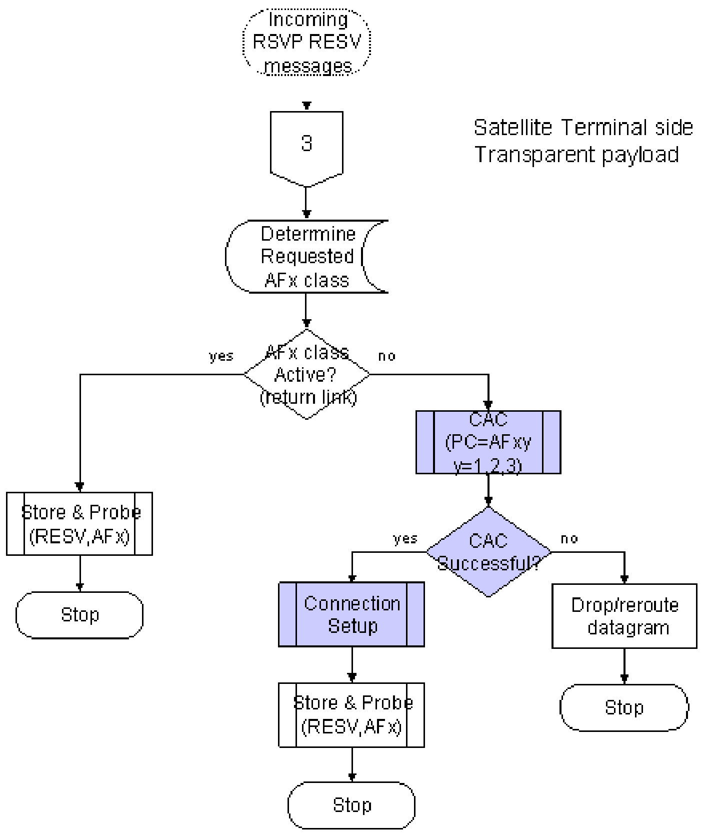


figure 102 - IP packet flow handling at ST – Incoming DiffServ–tunneled RSVP Resv message

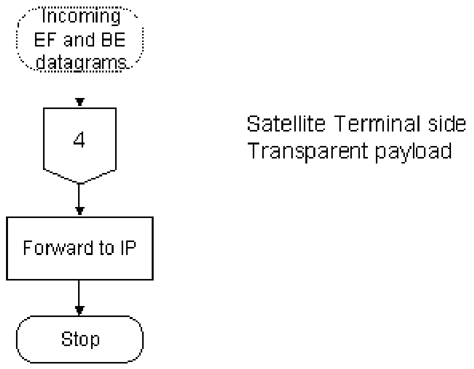


figure 103 - IP packet flow handling at Satellite Terminal – EF and BE traffic leaving satellite

The figure 102 describes logical operations carried out when incoming RSVP Resv messages are intercepted. These are messages generated outside the DiffServ domain they have completely crossed, and they are now leaving. ST can only perform a connection admission control before forwarding the Resv messages to the next RSVP router in the terrestrial network. So, the terminal determines the opportune AF class to be probed (in compliance with the SLA and the resource requirements reported in the payload of the RSVP Resv message), and then:

- if a pre-configured AFx connection is available at Layer 2 in the return link, Resv message is blocked and locally stored, in the meanwhile a GRIP probe is generated and sent to the opposite DiffServ edge node (see figure 106, for a detailed description of "Store & Probe" procedure referenced in figure 102);
- if that is not the case, a CAC procedure is invoked, checking for resource in order to accommodate every precedence level AFxy datagrams;
    - In case of positive CAC outcome, a Layer 2 connection is open for AFx, and the previously mentioned "Store & Probe" procedure is invoked.
    - In negative case, the RSVP Resv has to be dropped or, if possible, re-routed.

The ST actions to be enforced after receiving a BE or a EF datagram from the satellite is drawn in figure 103: it has simply to be forwarded in the IntServ domain. In fact, while crossing a DiffServ domain, packets are not changed but in the TOS field in the IP header. Since TOS is unused in IntServ, datagrams leaving a DiffServ domain, are already in the IntServ format, and do not require any additional elaboration.

Let's now analyze incoming GRIP datagrams. The flow chart in figure 104 concerns with GRIP acks and data (AFx1 for acks and normal data, AFx3 for low priority data). At the beginning, AFx3 datagrams are filtered and forwarded, while AFx1 datagrams are further analyzed, in order to check if they represent acks or normal data. When an incoming AFx1 is received, it needs to look-up at the Probing Table, in order to verify whether the node was waiting for that specific GRIP ack or not:

- if the ack was expected, the previously stored RSVP Resv, correlated with the GRIP ack, is forwarded to its final destination. The correspondent entry in the Probing Table is deleted and the GRIP ack is removed; finally a connection setup is possibly performed in he return link for every kind of AFx traffic[34], if not yet available;

- if non expected, the datagram has to be normally forwarded to the IP layer.

Finally, in figure 105, incoming GRIP probes are taken into account. These are datagrams signaling a connection attempt, and the satellite terminal has to give back an acknowledgement (GRIP ack) if and only if the incoming AFx traffic is not overwhelming the satellite terminal capacity. Thus a measurement based decision criterion is adopted by the GRIP entity. Upon the decision is taken, the following rules apply:

- if the stream (Layer 3) is accepted, and an active AFx connection is available in the return link (Layer 2), a GRIP ack is generated, as reported in the "GRIP ack (probe, class)" procedure in figure 107;

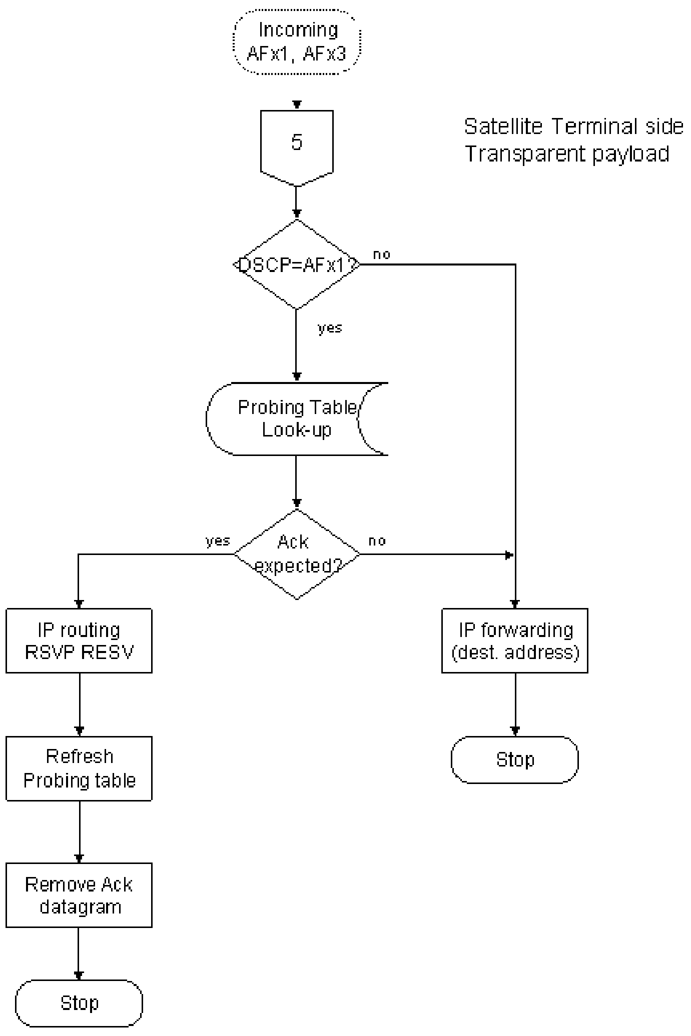


figure 104 - IP packet flow handling at Satellite Terminal – GRIP acks from DiffServ core

[34] Receiving the ack, implicitly signals that satellite network has room for that stream, thus no additional NCC CAC is needed. The same is true for the satellite terminal, since ack is generated only after a probe, and a probe is sent only if local resources are available.

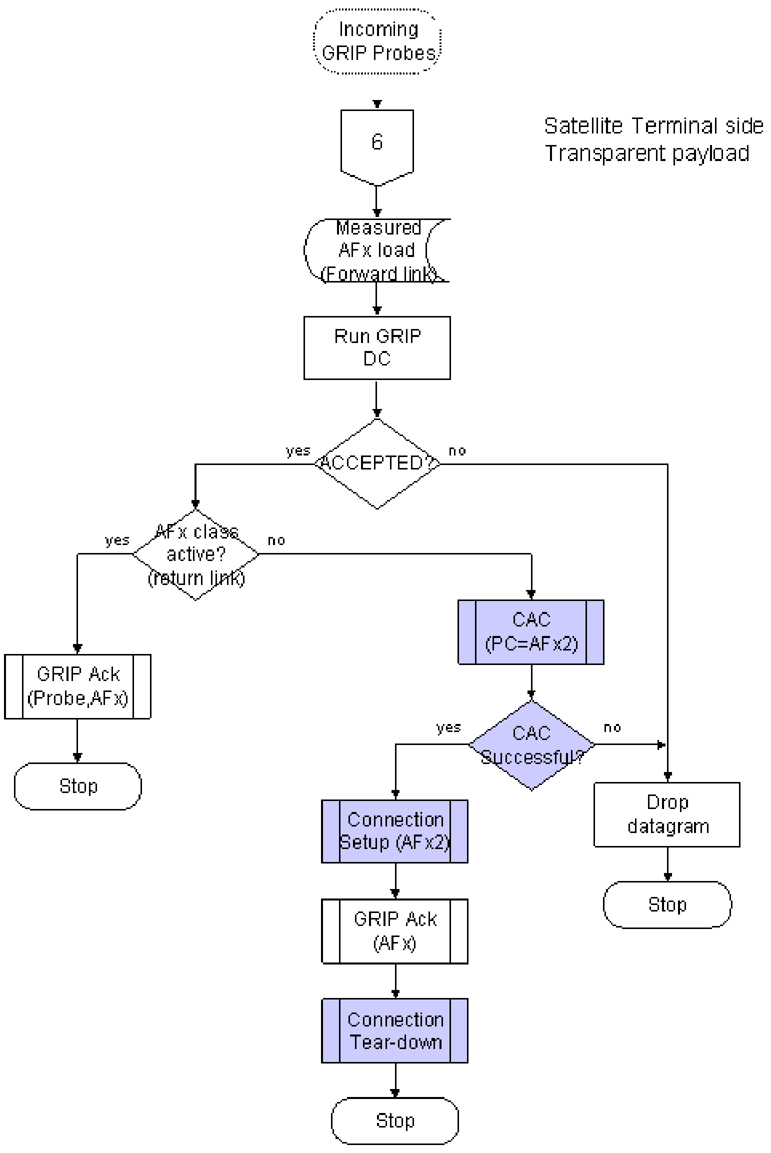


figure 105 - IP packet flow handling at Satellite Terminal – GRIP probes from DiffServ core

- otherwise, if the stream is accepted but an Layer 2 AFx connection is not available, a connection admission control procedure in the return link is invoked in order to setup the connection;
  - if the Layer 2 connection can be established, GRIP ack procedure is invoked;
  - if not, Layer 3 connection attempt is aborted by not replaying, and GRIP probe is removed;
- if the stream is not accepted, GRIP probe is removed and no feedback is sent, so that the new stream is implicitly not admitted (after a timeout expires).

For completeness, we now briefly comment figure 106 and figure 107 in which to relevant GRIP procedures are logically reported. The first figure represents the "Store &

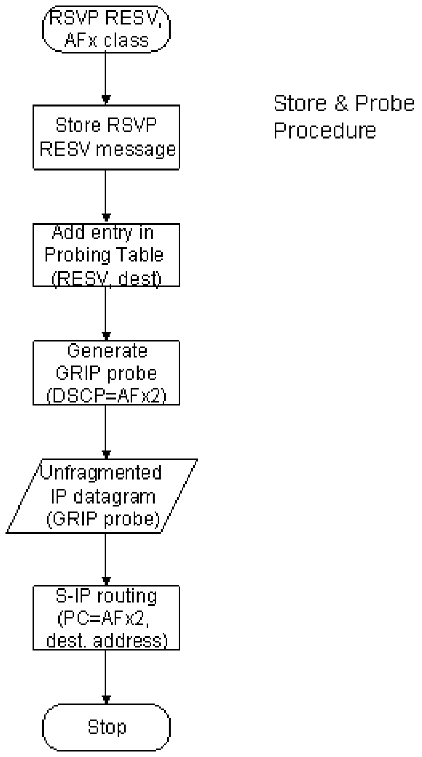

RSVP RESV, AFx class
Store RSVP RESV message
Store & Probe Procedure
Add entry in Probing Table (RESV, dest)
Generate GRIP probe (DSCP=AFx2)
Unfragmented IP datagram (GRIP probe)
S-IP routing (PC=AFx2, dest. address)
Stop

figure 106 - Predefined procedure flow chart diagram: Store & Probe (datagram, class)

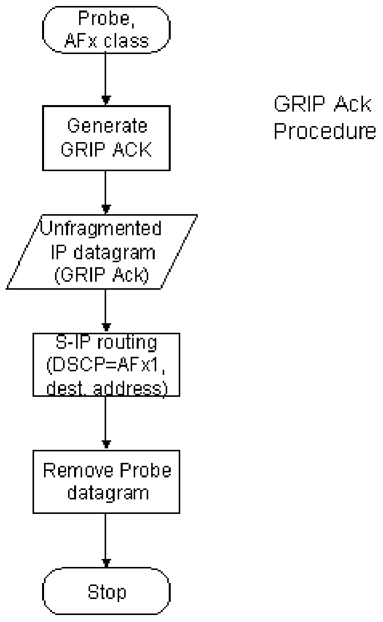


figure 107 - Predefined procedure flow chart diagram: GRIP ack (class)

Probe(message, class)" procedure, in which a RSVP Resv path and a suitable AFx class identifier are used as parameters. The procedure consists in storing in the local storage area the complete Resv message, while recording it in a new entry in the Probing Table. Then a GRIP probe is arranged and marked with an AFx2 DSCP tag. Finally, the probe is sent using as destination the egress DiffServ router the Resv came from. The last procedure, "GRIP Ack(probe_pointer, class_id)" is depicted in figure 107. A pointer to the probe originating the procedure is passed as first parameter (probe_pointer), and also in this case, a suitable AFx class identifier is needed (class_id), since the procedure generates a GRIP ack with an AFx2 tag, and then the datagram is forwarded in the return link. The GRIP probe datagram is finally deleted.

§6.2.3.3 Control-plane interworking

From the description given above, it is clear that GRIP mechanisms rely on pure data-plane operation. Nonetheless, GRIP message exchanges give rise to a set of control-plane actions, both operated at NCC and satellite terminal side. Assuming this perspective, GRIP messages play the role of a QoS signaling protocol which allows the satellite system to perform a dynamic admission control in a transparent payload scenario. The interworking layer, in the control-plane, is in charge of translating GRIP QoS signaling into DVB QoS messages. Furthermore a differentiation between satellite gateway and satellite terminal control operation is due because of the different roles played in the DiffServ architecture: the NCC is inside the DiffServ domain, while the ST is at the edge of the DiffServ domain and it is interfaced with an IntServ domain.

At gateway side, where a NCC is present, both in forward and return links GRIP probes have to be parsed, as already shown in section §6.2.3.1. As to the forward link, the NCC intercepts GRIP probes (labeled with an AFx2 DSCP) originated by the terrestrial DiffServ endpoints and, if the GRIP decision criterion returns a positive response, the AFx class is mapped on three correspondent DVB Profile Classes. Under the assumption that in a AF class, say AFx, AFx2 (probes) and AFx3 (potentially useful for advanced solutions) traffic is negligible, the NCC has to check whether in the satellite forward link there is room enough for a new AFx1-labeled stream. If AFx2 plus AFx3 traffic is relevant, NCC should take into account the aggregate statistics of AFxy traffics, y=1,2,3. Note that, since DiffServ mechanisms are not supported over flying satellite devices, each AFx traffic aggregate is subdivided into three subclasses, AFx1, AFx2 and AFx3, and up to three different satellite connections could be activated in order to strengthen the service differentiation between AF precedence levels.

AF connections from the gateway to a single satellite should be opened only once per satellite terminal, until the satellite terminal logs-off. In fact, when the connection is established, admission control should be performed by GRIP operation. This is still true also for EF and BE connections, since EF, BE and AFx classes follow predefined PHBs, whose resources should be pre-allocated in each DiffServ node. Of course, when a DiffServ class (PHB) does not use its resources, other classes are temporary allowed to use them, and this mechanism is natively adopted by DVB-S and DVB-RCS MAC, using, for instance, the automatic Free Capacity Assignment mechanism. In the proposed solution, DVB-RCS connections (Layer 2) are opened in the forward link only in occurrence of the first datagram requesting to be forwarded towards a destination subnet, i.e., addressed to a satellite terminal not yet supporting the requested DiffServ class in that direction. Moreover, an AFx class setup will be required in the forward link as long as a GRIP probe crosses the gateway while in the accept state, even though the GRIP probe is an incoming datagram, since the correspondent GRIP ack will be accommodated in the forward link. By taking into account that DiffServ classes parameters are predefined, the signaling payload in a GRIP probes datagram consists of its DSCP label.

Satellite connections in the forward link are established by means of NCC-initiated MAC signaling procedures. As regards the return link, the NCC interacts with the satellite terminal, which generally invokes the setup procedure for a new Layer 2 connection. Nonetheless, NCC parses GRIP probes originated by the satellite terminal and implicitly communicates its decision about new streams accommodation via GRIP edge-to-edge mechanisms: this could require an explicit request to the ST in order to open a satellite connection in both forward and return link, to accommodate GRIP messages. Thus, connections in the return link could be setup by means of both NCC- or satellite terminal-initiated MAC signaling procedures.

At terminal side only GRIP datagrams plus RSVP Resv messages (encapsulated in IP datagrams) coming from satellite have to be parsed and only signaling messages addressed to the return link are managed. As regards the return link, a new AF stream connection-attempt triggers the generation of a GRIP probe, upon a Resv message entries the ST from the forward link. This probe signals the stream requirements by means of a suitable DSCP in the AF range by identifying the AFx class to be used, so that a Layer 2 connection for AFx class has to be setup, if not yet available. Local admission control is operated based on the AFx class and the traffic load statistics that the node has meanwhile collected, if any. Connections in the return link should be opened only once per logon. In fact, once the terminal is logged, all the traffic towards the satellite pertains to EF, BE or AFx aggregates (x=1,2,3,4) which has a predefined amount of resources allotted in the return link. Summarizing, the terminal operates a connection setup only in correspondence with:

- the first BE datagram is received from terrestrial network;
- the first EF datagram is received from terrestrial network;
- the first RSVP Resv message requesting AFx class is received from the satellite;
- the first AFx1 GRIP ack has to be sent as positive feedback after an AFx2 GRIP probe is received (this could be optionally triggered by the NCC).

These are events that cause MAC signaling between ST and NCC, and they are initiated by the ST, but for the last point, where signaling could be initiated by ST or NCC. Moreover, we want to stress that dynamic stream admission is supported only for AF PHBs, while BE and EF traffic is generated at domain edge, in accordance with the adopted SLA. As regards forward link, explicit MAC signaling messages are exchanged with NCC, but they are exclusively NCC-initiated when the gateway wants to connect with the terminal.

In conclusion, note that given an AFx class, no other parameters are required in order to perform local or network admission control procedures. A suitable RSVP/GRIP mapping is still needed; in fact, GRIP selects the AFx class in accordance with the traffic parameters specified in the payload of an incoming RSVP Resv message.

§6.3 A concrete example of cross-layer interoperation

A very critical issues is represented by the NSLP translation in a NSIS-like framework (see section §4.1), whose functionalities can also be extended to the interaction with QoS handlers mechanisms in a specific node. This issue (not stressed in the examples presented in subsection §4.1.4) is addressed in the following example. Here we consider a scenario similar to the one presented before in subsections §4.1.2 and §4.1.4, but a specific component of the intermediate domain is highlighted: a satellite DVB-RCS segment with the six Profile Classes and their corresponding behaviors as already defined and commented in §6.1.3 and table 9.

For the sake of simplicity, in this example we consider the scenario depicted in figure 108, where SRC and DST endpoints are RSVP-capable. The operation described in subsection §4.1.4.1 (RSVP over GRIP) is adopted. Hence the RSVP signaling extension is triggered by the Path messages entering the DiffServ domain (at router R2, in the figure). Additionally, we highlight the case of IntServ services to be mapped on AF PHBs, since in this case the proposed GRIP signaling extensions apply.

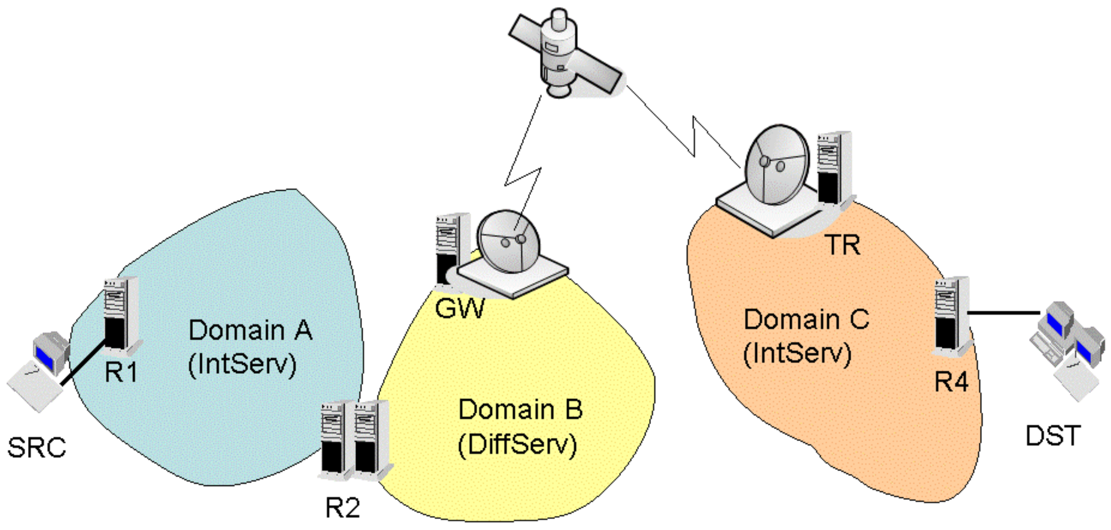


figure 108 - IntServ-DiffServ-Satellite scenario

IntServ Class	Application requirements/priority	DiffServ PHB
GS	High priority, low loss	EF
CLS	High priority and high rate	AF1
	High priority and low rate	AF2
	Low priority and high rate	AF3
	Low priority and low rate	AF4
BE	Best effort	BE

table 14 - IntServ-DiffServ mapping

In order to start a new connection, a Path message is sent from SRC and accordingly dealt with in Domain A. If RSVP routers in Domain A have room enough to accommodate the requested connection, R2 starts an implicit signaling admission control procedure in the scope of Domain B, via GRIP probing operation.

To that aim, RSVP Path messages has to be previously mapped on a specific DiffServ class. In fact, GRIP probing has to be performed solely in the class that is candidate to accommodate the flow, if established. A possible IntServ to DiffServ mapping is reported in table 14, where the CLS service is split in four sub-classes, in correspondence with the four AF PHBs. The mapping takes into account traffic priorities and application requirements, as reported in the second column of table 14. Actually, table 14 can be seen as the combination of the two following tables reported below for IntServ and DiffServ mapping on DVB-RCS PCs.

Hence, router R2 parses the payload conveyed in the Path message and selects the correspondent service, which, in turn, corresponds to a DiffServ PHB. If the selected PHB falls in the range of AF classes, the GRIP procedure begins, otherwise packets are forwarded in EF PHB (for GS service) or BE PHB (for IntServ BE service). Upon an AF DiffServ PHB is selected, the Path message is accordingly labeled with a probing DSCP, and forwarded towards the GW node, that is a satellite gateway located at the edge of a DiffServ domain. In the assumption that DVB-RCS technology is adopted, the signaling message can be forwarded to the satellite, but it is needed to map the DiffServ code point on DVB-RCS PCs. An example of such a mapping is reported in table 15. After the satellite hop, the RSVP Path message (which was labeled as a probe and forwarded within a specific DVB-RCS profile class) is injected by TR (the terrestrial satellite terminal) in the following IntServ domain, and it is parsed by RSVP routers and finally by DST, that possibly generates the Resv message. Of course, the Path message encapsulated in the probe packet could be dropped by the edge router R2, the GW, and also by the DVB-RCS terminal TR, in accordance with the GRIP decision criterion adopted. If the packet is not dropped, TR provides the de-encapsulation function before to be forwarded to DST.

DVB-RCS approach allows the system to use the satellite link in the reverse path, so that the RSVP response possibly sent by DST, i.e., the Resv message, will be forwarded back to Domain B through the return satellite link, and RSVP to DVB-RCS mapping has to be performed by TR. As a reference, one could consider the mapping presented in table 16 where the IntServ CLS has been decomposed in four different services, in accordance with the traffic priorities and data rates of applications. Note that the mapping in table 16 slightly differs from mapping reported in table 10, since the DVB-RCS PC2 is now devoted to host high performance CLS applications instead of a low level GS applications. This alternative approach is twofold: in this way the number of CLS services matches the number of AF PHBs; moreover, application requiring real time compatible performance, are clustered in a single group, which is more easily to deal with. However, we recall that the mapping proposals presented in this thesis can be adopted as a generic guideline for a real implementation, and multiple solutions should be available to meet the manifold operating circumstances that a network provider could encounter.

Hence, RSVP Resv packets are mapped accordingly to the requesting service, as reported in the table 16, while the right DSCP to be used when entering the Domain B is chosen in accordance with the actual DVB-RCS PC (which corresponds to a IntServ service) and based on the combination of mapping functions reported in table 15 and table 16, which explain the nature of table 14. For instances a RSVP Path message relative to a CLS with low delay constrains and high bandwidth requirements corresponds to be behavior granted by PC3 in table 16, and this turns in a DSCP = AF11 in table 15 for data and GRIP feedbacks, while probes will use AF12. Thus, when a Path leaves the node TR with a given AFx2 tag, an entry is added in the TR mapping table, in order to accordingly accommodate the subsequent Resv message with an AFx1 tag. The rest of the network operation follows the behavior already shown in subsection §4.1.4.1.

DiffServ PHB		**DVB-RCS profile class**					
Class	***Precedence***	***1***	***2***	***3***	***4***	***5***	***6***
EF	---	√					
AF1	AF11 – Data/Feedback		√				
	AF12 – Probe			√			
	AF13				√		
AF2	AF21 – Data/Feedback			√			
	AF22 – Probe				√		
	AF23					√	
AF3	AF31 – Data/Feedback				√		
	AF32 – Probe					√	
	AF33						√
BE							√

table 15 - DiffServ-DVB-RCS mapping

IntServ class		**DVB-RCS profile class**					
		1	*2*	*3*	*4*	*5*	*6*
GS		√					
CLS	*High priority, high rate*		√				
	High priority, low rate			√			
	Low priority, high rate				√		
	Low priority, low rate					√	
BE							√

table 16 - IntServ-DVB-RCS mapping

Finally, note that these mapping strategies have to be straight managed by NSLP protocols with the support of NTLP. In fact, NTLP is in charge of preserving signaling messages while traversing the network, while NSLP locally adapts reservation procedures with priorities and

QoS mechanisms resident on the local domain and on the local machine too. Thus, flow priorities and state information have to be dealt with by the NTLP protocol, while NSLP has to be compliant with lower level QoS features, and drive it like in the case of DVB-RCS satellite links.

§6.4 Conclusions

In this chapter we focused on the cross-layer interworking between Layer 3 and Layer 2 (MAC) protocols for an optimized QoS support for applications. The necessity for new network architectures and for the internetworking of reservation and connection control protocols running in heterogeneous domains has been presented in previous chapters of this thesis. Moreover, here we explained how a third level of support comes from the interoperation of network protocols and MAC-layer mechanisms. In particular we focused on the DVB-RCS technology for satellite bidirectional connections, and we proposed multiple examples for the mapping of DiffServ and IntServ classes and services on the underlying satellite MAC facilities. Eventually we have shown that it is possible to harmonize the interaction of DiffServ endowed with GRIP implicit signaling, and IntServ with RSVP protocol for resource reservation, also taking into account a QoS-enabling mapping of network features and services on DVB-RCS Profile Classes.

The mapping of services, that is carried out through the pages of the chapter, mostly concerns three focus topics: i) map GS, CLS and BE IntServ services on DVB-RCS PCs; ii) map EF, AF and BE DiffServ PHB on DVB-RCS PCs; iii) map IntServ services on DiffServ PHBs while mapping DiffServ PHBs and GRIP semantic on DVB-RCS PCs. We propose some guidelines to sharply draw new mapping tables in addition to the mapping exemplification reported through the sections of this chapter.

As to the case of nested mapping issues proposed in point iii), a thorough study of the modified DVB-RSC transport function level behavior was proposed. In particular, the characterization of the operation performed by DVB-RCS nodes (Satellite Terminal or Gateway/Hub node with NCC embedded) is reported in the flow chart diagrams of section §6.2, and a concrete network example is described in section §6.3. This example concludes the chapter and, in turn, the thesis by adopting a scenario in which the following aspects are addressed: DiffServ and IntServ networking for QoS-aware network design; RSVP-like resource reservation; GRIP-like unicast connection blocking function based on run time network load estimate (distributed edge-to-edge measurement based admission control);

interoperation of QoS-enabling mechanisms for stateless and stateful networks (respectively, RSVP and GRIP mechanisms adopted by IntServ and DiffServ networks); cross-layer internetworking (network and MAC layers). Actually, the last example summarizes almost all the issues faced in the entire thesis, with the exception of multicast-related issues and bandwidth sharing issues. It also proposes a concrete example of implementation of the variously proposed approaches in a real network scenario, and it has been proposed as part of an Internet Draft [70] in the frame of the NSIS ongoing discussion. The effectiveness of such an integrated vision is not given in this chapter, but for the evidence that mechanisms and protocols theoretically provided through the pages of this and previous chapters go well beyond the perspective of pure research. Conversely, ideas proposed in this thesis meet concrete networking requirements.

Conclusive remarks

Internet has resource enough to support applications that require quality of service to suitably run at user side. In order to actually permit an effective exploitation of such a capability, the major issue to be addressed is the control of resource usage. Firstly is necessary to introduce some new features to the IP architecture, because the traditionally employed Best Effort model does not provide any means to differentiate data flows and protect some high priority traffic. Secondly, given a model in which multiple services can be distinguished, the network is responsible of the control of traffic admitted to services. In fact, each service could be easily reserved a fixed or variable amount of resources, more difficult is to control the usage of allotted resources, that is useful because performance experienced by packets depends on parameters like the offered load and the traffic burstiness. Networks should be endowed with tools that operate by limiting the load and smooth as much as possible the variability of the aggregate traffic behavior. Due to the complexity of traffic control and due to the unfeasibility of a full fledged reservation approach in the core network, the majority of QoS-enabling schemes should rely on edge-to-edge operations in which the complexity is bound to the network edge and to the endpoints. Thirdly, upon a customer connection is admitted to a service, the network or service provider has to honor the service level specified in a somehow defined service level agreement. Additionally, if a user, or a group of users represented by a proxy, is admitted to a set of services sharing the same resources, it is interest of the customer to smartly share the available resources among active applications. Thus, the active monitoring and control of allotted resources usage represents a value added tool that improve the level of quality experienced by applications. Eventually, some recent approaches to QoS go above the legacy interoperation between network layer entities and propose a cross-layer interoperation to harmonize the effort of Layer 2 and Layer 3 protocols to support services with QoS.

With reference to the four points outlined before, in this thesis we discussed the following arguments. Firstly we introduced the DiffServ and IntServ IP architectures and their pros and cons in real network implementations. Secondly, we discussed the admission control features and in particular the capabilities of measurement based admission control schemes, which can be joined with the flexibility of endpoint admission control schemes in a simple but powerful connection blocking mechanism called GRIP. The approach proposed in the thesis is integrated in a DiffServ network scenario, even though it is a more general approach to light, flexible, scalable and stateless per-flow admission control. It also solves the problem of performance impairments due to traffic bursts by compensating the variability of traffic emitted by flows with the variability of the number of concurrent sources admitted to a service. We proposed admission control variants for unicast, and multicast applications, and we proposed to use the GRIP approach to verify the availability of resources through a stateless domain inserted in the path composed by RSVP-compliant nodes: the stateless domain can be seen as a virtual link between two consecutive stateful routers, and the QoS in the virtual link can be provided by the GRIP admission control rules. As to the third point, we proposed an effective proxy-based approach to bandwidth sharing between concurrent multimedia applications. The effectiveness of the scheme is due to the elastic buffer behavior enforced by the proxy, with the suitable possibility to decouple the usage of network accesses from the delivery of data to the end users. The proxy retrieves data as fast as possible, and it is in charge of allotting the available bandwidth to active applications, trying to equate the amount of pre-fetched data for all concurrent applications. This approach aims at exploiting network resources when they are available, and not simply in correspondence with user requests. The proposed solution is particularly attractive in the case of time variant condition in the network access channel, and the concrete example of satellite link connecting proxies located on mobile vehicles is given. However, this treatment also applies to terrestrial broadband wireless technologies. Eventually, we closed the thesis with an example of cross-layer interoperation between IntServ, DiffServ and DVB-RCS system, in which QoS support to application is the product of an harmonized mapping between IntServ classes, RSVP messages, DiffServ DSCPs and GRIP messages over appropriate DVB-RCS Profile Classes.

Acknowledgements

I would like to thank my tutor, prof. Giuseppe Bianchi, for his constant support through these years and specially for his stimulant ideas. The work presented in this thesis has been partially carried out in collaboration with other people I am very grateful to: Giovanni Neglia for the work about MBAC and LRD, Paolo Di Francesco and the friends from CRES in Monreale for the contribute to test bed implementations of GRIP and the API, Francesco Vetrano for the multicast release of GRIP, Marco Gambardella for the study of services in VANs, and some graduating students for their contribute to my research. Also, I want to express gratitude to my parents and my family since they believed in me and sustained my path, and to my colleagues and friends for their love and patience.

Bibliographic References

[1] P. Abry, D. Veitch, “Wavelet Analysis of Long-Range Dependent Traffic”, IEEE Transactions on Information Theory, 44(1), pp. 2-15, January 1998.

[2] I. F. Akyildiz, S. H. Jeong, “Satellite ATM Networks: A Survey”, IEEE Communications Magazine, pp. 30-43, July 1997.

[3] W. Almesberger, T. Ferrari, J. Y. Le Boudec: “SRP: a Scalable Resource Reservation Protocol for the Internet”, IWQoS’98, Napa (California), May 1998.

[4] F. Baker, R. Guerin, and D. Kandlur, “Specification of Committed Rate Quality of Service,” Internet Draft, June 1996, ftp://ds.internic.net/internetdrafts/draft-ietf-intserv-commit-rate-svc-00.txt.

[5] J. Beran, R. Sherman, W. Willinger, M. S. Taqqu, “Variable-bit-rate video traffic and long-range dependence”, IEEE Trans. on Communications, 1995

[6] G. Bianchi, N. Blefari-Melazzi: “A Migration Path for the Internet: from Best-Effort to a QoS Capable Infrastructure by means of Localized Admission Control”, Lecture Notes on Computer Science, Springer-Verlag, volume 1989, January 2001 (a more detailed technical report can be found at http://drake.diei.unipg.it/netweb/GRIP_tech_rep.pdf).

[7] G. Bianchi, N. Blefari-Melazzi: “Admission Control over AF PHB groups”, internet draft, draft_bianchi_blefari_admcontr_over_af_phb.txt, March 2001, work in progress.

[8] G. Bianchi, A. Capone, C. Petrioli, “Packet Management Techniques for Measurement Based End-to-end Admission Control”, KICS/IEEE Journal on Communications and Networking, June 2000.

[9] G. Bianchi, V. Mancuso, P. Di Francesco, “An API for advanced traffic control in DiffServ routers”, proceeding of Net-Con'2002, Paris, France, October 2002

[10] G.Bianchi, V. Capaccio, N. Blefari-Melazzi: “Per-flow signalling extension across DiffServ domains”, NETCOM 2002, Paris.

[11] G.Bianchi, V.Mancuso and G.Neglia , “On the Self-Similarity of Measurement-Based Admission Controlled Traffic”, proceedings of IEEE Globecom 2002, Taipei, Taiwan, November 2002

[12] G. Bianchi, R. Melen, “The role of the local storage in supporting video retrieval services on ATM networks”, IEEE/ACM Transaction in networking, vol.5, no.6, December 1997

[13] G. Bianchi, “Buffer Sizing for High Speed Video Information Retrieval on ATM Networks”, Proceedings of IEEE GLOBECOM’97, November 1997, Phoenix, AZ, USA.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: “An Architecture for Differentiated Services”, RFC 2475, Dec. 1998.

[15] R. Bless, K. Wehrle: “IP Multicast in Differentiated Services Networks”, Internet Draft, draft-bless-diffserv-multicast-03.txt, March 2002, work in progress.

[16] V. Bolotin, “Modeling Call Holding Time Distributions for CCS Network Design and Performance Analysis”, IEEE Journal on Selected Areas in Communications 12, 3, pp.433-438, April 1994.

[17] F. Borgonovo, A. Capone, L. Fratta, M. Marchese, C. Petrioli, “PCP: A Bandwidth Guaranteed Transport Service for IP networks”, IEEE ICC’99, June 1999.

[18] R. Braden, L Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC2205, September 1997.

[19] P. T. Brady, "A statistical analysis of on-off patterns in 16 conversations", Bell System Technical Journal, vol. 47, pp.73-91, Jan. 1968.

[20] L. Breslau, E. W. Knightly, S. Schenker, I. Stoica, H. Zhang: "Endpoint Admission Control: Architectural Issues and Performance", ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.

[21] L. Breslau, S. Jamin, S. Shenker, "Comments on the Performance of Measurement Based Admission Control Algorithms", Proc. of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.

[22] M. Brunner et al., "Requirements for Signaling Protocols",RFC 3726, April 2004.

[23] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, "A Framework for Integrated Services Operation Over DiffServ Networks", RFC 2998, November 2000.

[24] C. Cetinkaya, E. Knightly, "Egress Admission Control", Proc. of IEEE Infocom 2000, Tel-Aviv, March 2000.

[25] S. Chen, K. Nahrstedt, Y. Shavitt: "A QoS-aware multicast routing protocol", IEEE JSAC, Vol. 18, No. 12, Dec. 2000.

[26] P. Chitre and F Yegenoglu, "Next-Generation Satellite Networks: Architecture and Implementations," IEEE Communications Magazine, pp. 30-36, March 1999.

[27] Cisco IOS Quality of Service Solutions Configuration Guide—Online Manual. http://www.cisco.com/univercd/cc/td/doc/product/software/ ios120/12cgcr/qos_c/

[28] D. Clark, "The Design Philosophy of the DARPA Internet Protocols," Proc. ACM SIGCOMM '88, Aug. 1988.

[29] D. Clark and W. Fang. "Explicit Allocation of Best Effort Packet Delivery Service". IEEE/ACM Transactions on Networking, vol. 6, no. 4, pp. 363-373, August 1998.

[30] M. Crovella, P. Barford, "The network effects of prefetching", Proceedings of IEEE Infocom'98, 1998

[31] M. E. Crovella, A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, 5(6):835--846, December 1997.

[32] Differentiated Services on Linux: http://diffserv.sourceforge.net

[33] E. Ekici, I. F. Akyildiz, and M. D. Bender, "Datagram Routing Algorithm for LEO Satellite Networks," Proceedings of INFOCOM 2000, Tel Aviv, Israel, 2000.

[34] V. Elek, G. Karlsson, "Admission Control Based on End-to-End Measurements", Proc. of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.

[35] Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei: "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, June 1998.

[36] ETSI EN 300 421 v1.1.2 (1997-08): Digital video Broadcasting (DVB); Framing Structure, Channel Coding and modulation for 11/12 GHz satellite services.

[37] ETSI EN 301 192 V1.3.1 (2003-05) Digital Video Broadcasting (DVB); DVB specification for data broadcasting

[38] ETSI ETS 300 802 Digital Video Broadcasting (DVB); Network-independent protocols for DVB interactive services, November 1997

[39] ETSI EN 301 790 v1.3.1 Digital Video Broadcasting; Interaction Channel for Satellite Distribution Systems, 2002/11

[40] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas: "Protocol Independent Multicast - Sparse Mode (PIM-SM) Protocol Specification (Revised)", draft-ietf-pim-sm-v2-new-05.txt, Internet-Draft, work in progress, March 2002.

[41] V. Firoiu, D. Towsley: "Call admission and resource reservation for multicast sessions", IEEE INFOCOM 1996, pp. 94-101.

[42] S. Floyd, "Comments on measurements based admission control for controlled-load services", Technical report, July 1996.

[43] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397-413, August 1993.

[44] N. Ghani, and S Dixit, "TCP/IP Enhancements for Satellite Networks," IEEE Communications Magazine, pp. 64-72, July 1999.

[45] R. Gibbens, F. P. Kelly, "Measurement-Based Connection Admission Control", Proc. of 15th International Teletraffic Congress, June 1997

[46] M. Grossglauser and J. Bolot, "On the Relevance of Long Range Dependence in Network Traffic", IEEE/ACM Transactions on Networking, 1998.

[47] M. Grossglauser, D. Tse, "A Framework for Robust Measurement Based Admission Control", IEEE/ACM Transactions on Networking Vol. 7, No. 3, July 1999.

[48] M. Greiner, M. Jobmann, C. Klüppelberg, "Telecommunication Traffic, Queueing Models, and Subexponential Distributions", Blauer Bericht TUM M9805, Technische Universität München, May 1998

[49] R. Hancock et al., "Next Steps in Signaling: Framework", DRAFT draft-ietf-nsis-fw-05.txt, October 2003.

[50] J. Heinanen, F. Baker, W. Weiss, J. Wroclavski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

[51] http://www.emulab.ee.mu.oz.au/~darryl/secondorder_code.html

[52] G. Huston, "Next Steps for the IP QoS Architecture", RFC2990, November 2000.

[53] IEEE 802.16e Mobility Enhancements IEEE 802.16 Presentation Submission Template (Rev. 8.3) Document Number: IEEE S802.16e-03/05].

[54] IEEE Standard 802.16: A Technical Overview of the WirelessMAN Air Interface for Broadband Wireless Access ; Cal Eklund, Nokai Research Center.

[55] IEEE C802.20-03/104: IEEE 802.20 Working Group on Mobile Broadband Wireless Access.

[56] IEEE 802.16.2-2001, "IEEE Recommended Practice for Local and Metropolitan Area Networks — Coexistence of Fixed Broadband Wireless Access Systems," Sept. 10, 2001.

[57] IETF home page, http://www.ietf.cnri.reston.va.us.

[58] Integrated Services Charter, http://www.ietf.org/html.charters/intservcharter.html.

[59] ISO/IEC 13818-1:2000, Information technology -- Generic coding of moving pictures and

associated audio information: Systems

[60] ISO/IEC 13818-6:1998, Information technology -- Generic coding of moving pictures and associated audio information -- Part 6: Extensions for DSM-CC

[61] N. Imai, K. Kaneco, H. Morikawa, T. Aoyama, "On On-demand Data Prefetching System for Spotted Access Networks", IEICE TRANS. COMMUN., Volume E84-B, Number 10, October 2001

[62] IPoS system architecture, www.hnc.com

[63] V. Jacobson, K. Nichols, K.Poduri, "An Expedited Forwarding PHB", RFC2598, June 1999

[64] R. Jain. "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks". Computer Communication Review, vol. 9, no. 5, pp. 56-71, October 1989.

[65] S. Jamin, P. B. Danzig, S. Shenker, L. A. Zhang, "A measurement-based admission control algorithm for integrated services packet networks", IEEE/ACM Transactions on Networking Vol. 5, No. 1, Feb 1997, pp. 56-70.

[66] S. Jin, A. Bestavros, A. Iyengar, "Accelerating Internet Streaming Media Delivery using Network-Aware Partial Caching" proceeding of the 22nd International Conference on Computing Systems - IEEE ICDCS'02, 2002

[67] F. P. Kelly, P. B. Key, S. Zachary: "Distributed Admission Control", IEEE JSAC, Vol. 18, No. 12, December 2000.

[68] N. Likhanov, B. Tsybakov, N. Georganas, "Analysis of an ATM buffer with Self-Similar ("fractal") Input Traffic", Proc. IEEE INFOCOM '95, pp.985-992, 1995

[69] S. H. Low and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", IEEE/ACM Transactions on Networking, 7(6):861-75, Dec. 1999

[70] V. Mancuso, G.Bianchi, N. Blefari Melazzi, "Implicit Signaling over Stateless Networks", submitted as IETF draft, draft-mancuso-nsis-impl-sign-00.txt, June 2004, category: informational, expires January 2005

[71] V.Mancuso, G.Bianchi, "Streaming for vehicular users via elastic proxy buffer management", IEEE Communication Magazine, Volume: 42 , Issue: 11 , pp. 144-152, November 2004

[72] Z. Miao, A. Ortega, "Scalable Proxy Caching of Video Under Storage Constraints ", IEEE journal on selected areas in communications, vol. 20, no.7, p.1315-1327, September 2002

[73] D. Mitra and J. Seery. "Dynamic Adaptive Windows for High Speed Data Networks: Theory and Simulations". In Proceedings of SIGCOMM'90, pp. 30-40, September 1990.

[74] P. R. Morin, "The Impact of Self-Similarity on Network Performance Analysis", Ph.D. Dissertation, Carleton University, Dec. 1995

[75] K. Nichols, S. Blake, F. Baker, D. Black, "Definitions of the Differentiated Service Field (DS Field) in the Ipv4 and Ipv6 Headers", RFC2474, December 1998.

[76] K.Nichols, B. Carpenter, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification", RFC 3086, April 2001

[77] A. Parekh and R. Gallagher, "A Generalized Processor Sharing Approach to Flow Control — The Single Node Case," IEEE/ACM Trans.Networking, vol. 1, no. 3, 1993, pp. 366–57.

[78] A. Parekh and R. Gallagher, "A Generalized Processor Sharing Approach to Flow Control — The

Multiple Node Case,” IEEE/ACM Trans.Networking, vol. 2, no. 2, 1996, pp. 137–50.

[79] C. Partridge, and T. J. Shepard, “TCP/IP Performance over Satellite Links,” IEEE Networks, pp. 44-49, Sept./Oct. 1997

[80] V. Paxson, S. Floyd, “Wide-Area Traffic: The Failure of Poisson Modeling”, IEEE/ACM Transactions on Networking, Vol. 3 No. 3, pp. 226-244, June 1995.

[81] K. Park, G. Kim, M. Crovella, “On the Effect of Traffic Self-similarity On Network Performance”, Proceedings of SPIE's International Symposium and Education Program on Voice, Video, and Data Communications, November 1997, Dallas, Texas.

[82] K. Park, G. Kim, M. Crovella, “On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic”, Proc. of the International Conference on Network Protocols, pp. 171-180, Oct 1996.

[83] POLLENS web page on ITEA site: http://www.itea-office.org/projects/pollens_facts.html

[84] K. Ramakrishnan and S. Floyd. “A Proposal to add Explicit Congestion Notification (ECN) to IP”, RFC 2481, IETF, January 1999.

[85] M. Reissline, F. Hartanto, K. W. Ross, “Interactive video streaming with proxy servers”, proceeding of IMMCN, February 2000

[86] R. Rejaie, H. Yu, M. Handley, D. Estrin, “Multimedia proxy caching mechanisms for quality adaptive streaming applications in the internet”, proceeding of Infocom 2000, March 2000

[87] Z. Sahinoglu, S. Tekinay, “On Multimedia Networks: Self-Similar Traffic and Network Performance”, IEEE Communications Magazine, January 1999, pp. 48-52.

[88] D. Sanghia and A. Agrawala. “DTP: An Efficient Transport Protocol”, University of Maryland Technical Report, October 1991.

[89] H. Schulzrinne and R. Hancock, “GIMPS: General Internet Messaging Protocol for Signaling”, DRAFT draft-ietf-nsis-ntlp-02.txt, May 30, 2004.

[90] S. Sent, J. Rexford, D. Towsley, “Proxy prefix caching for multimedia streams”, Proceeding of Infocom 99, April 1999

[91] Y. Shuqian Yan, M. Faloutsos, A. Banerjea: “QoS-aware multicast routing for the Internet: the design and evaluation of QoSMIC”, IEEE/ACM Transactions on Networking, Vol. 10, No. 1, pp. 54-66, Feb. 2002.

[92] M. Stemm, R. H. Katz, “Vertical Handoffs in Wireless Overlay Networks,” ACM/Baltzer Mobile Networking and Applications (MONET), Special Issue on “Mobile Networking in the Internet”, Volume 3, Number 4, January 1999, pp. 319-334.

[93] I. Stoica, H. Zhang, “Providing guaranteed services without per flow management”, Proc. of ACM SIGCOMM 1999, Cambridge, MA, September 2000.

[94] A. Striegel, G. Manimaran: "A Scalable Protocol for Member Join/Leave in DiffServ Multicast", in Proc. of Local Computer Networks (LCN) 2001, Tampa, Florida, Nov. 2001.

[95] A. Striegel, G. Manimaran: “A survey of QoS multicasting issues”, IEEE Communications, pp. 82-87, June 2002.

[96] M. S. Taqqu, V. Teverosky, W. Willinger, “Estimators for long-range dependence: an empirical study”, Fractals 1996

[97] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, M. Dahlin, “The potential costs and benefits of long-term pre-etching for content distribution”, Computer communications journal, 25(4): 367-375, 2002

[98] B. Wang, S. Sen, M. Adler, D. Towsley, “Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution”, proceedings of Infocom 2002, 2002

[99] Y. Wang, Z. L. Wang, D. H. Du, D. Su, “A network-conscious approach of end-to-end video delivery over wide area networks using proxy servers”, Proceeding of Infocom 98, 1998

[100] Z. Wang and J. Crowcroft. “A New Congestion Control Scheme: Slow Start and Search (Tri-S)”. Computer Communication Review, vol. 21, no. 1, pp. 32-43, January 1991.

[101] M. Werner, A Jahn, E Lutz, and A Böttcher, “Analysis of System Parameters for LEO/ICO-Satellite Communication Networks,” IEEE Journal on Selected Areas in Communications, vol. 13, no. 2, pp. 371-381, Feb. 1995.

[102] WiMax forum official site: http://www.wimaxforum.org.

[103] W. Willinger, M. Taqqu, R. Sherman, D. Wilson, “Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level”, IEEE/ACM Transactions on Networking, Vol. 5, No. 1, pp. 71-86, February 1997.

[104] L. Wood, A. Clerget, I. Andrikopoulos, G. Pavlou, and W. Dabbous, “IP Routing Issues in Satellite Constellation Networks,” IEEE Journal on Selected Areas in Communications, vol. 18, no. 6, Dec. 2000.

[105] J. Wroclawsky, “The use of RSVP with IETF Integrated Services”, RFC2210, September 1997.

[106] Z. Xiang, Q. Zhang, W. Zhu, Y. Zhong, “Cost-Based Replacement Policy for Multimedia Proxy across Wireless Internet”, IEEE Globecom’01, 2001

[107] B. Yang, P. Mohapatra: "Multicasting in Differentiated Service Domains", IEEE Globecom 2002.

[108] G. K. Zipf, “Human behavior and the principle of least effort”, Reading, MA: Addison-Wesley, 1949

List of Acronyms

A2M	*All to Minimum*
ABR	*Available Bit Rate*
AC	*Admission Control*
AdSpec	*Advertising Specification*
AF	*Assured Forwarding*
API	*Application Program Interface*
ATM	*Asynchronous Transfer Mode*
BA	*Behavior Aggregate*
BE	*Best Effort*
BR	*Border Router*
CAC	*Connection Admission Control*
CBWFQ	*Class Based Weighted Fair Queueing*
CBR	*Constant Bit Rate*
CIR	*Committed Information Rate*
CLS	*Controlled Load Service*
CR	*Core Router*
DAMA	*Demand Assignment Multiple Access*
DiffServ	*Differentiated Services*
DRT	*Delayed Real Time*
DSCP	*DiffServ Code Point*
DSL	*Digital Subscriber Line*
DVB	*Digital Video Broadcast*
DVB-RCS	*Digital Video Broadcast – Return Channel Satellite*
DVB-S	*Digital Video Broadcast – Satellite*
EAC	*Endpoint Admission Control*
ECN	*Early Congestion Notification*
ED	*Equally Distributed*
EDGE	*Enhanced Data rates for GSM Evolution*
EEAC	*End-to-end Endpoint Admission Control*
EF	*Expedited Forwarding*
EMBAC	*Endpoint Measurement Based Admission Control*
FCA	*Free Capacity Assignment*
FEC	*Forward Error Correction*
FF	*Fixed Filter*
FSM	*Finite State Machine*
GEO	*Geostationary Earth Orbit*
GPRS	*General Packet Radio Service*
GRIP	*Gauge & Gate Reservation with Independent Probing*
GS	*Guaranteed Service*
ICMP	*Internet Control Message Protocol*
IE	*Information Element*
IETF	*Internet Engineering Task Force*
IGMP	*Internet Group Management Protocol*
IntServ	*Integrated Services*
IP	*Internet Protocol*
ISP	*Internet Service Provider*
LAN	*Local Area Network*
LEO	*Low Earth Orbit*
LOS	*Line-of-sight*
LRD	*Long Range Dependence*
MAC	*Medium Access Control*
MAN	*Metropolitan Area Network*
MBAC	*Measurement Based Admission Control*
MBR	*Maximum Burst Size*
MCR	*Minimum Cell Rate*

MEO	*Medium Earth Orbit*
MF-TDMA	*Multi Frequency Time Division Multiple Access*
MoD	*Multimedia-on-Demand*
MPEG	*Moving Picture Experts Group*
MPLS	*Multi-Protocol Label Switching*
MPTD	*Maximum Packet Transfer Delay*
MRK	*Markovian*
MRT	*Multicast Routing Table*
NCC	*Network Control Centre*
NE	*NSIS Entity*
NF	*NSIS Forwarder*
NI	*NSIS Initiator*
NLOS	*Non-Line-of-Sight*
NSIS	*Next Step In Signaling*
NSLP	*NSIS Signaling Layer Protocol*
NTLP	*NSIS Transport Layer Protocol*
OPWA	*One Pass With Advertising*
PBAC	*Parameter Based Admission Control*
PC	*Profile Class*
PCR	*Peak Constant Rate*
PDB	*Per Domain Behavior*
PDP	*Policy Decision Point*
PDR	*Peak Data Rate*
PDU	*Packet Data Unit*
PHB	*Per Hop Behavior*
PIM	*Protocol Independent Multicast*
PIM-SM	*PIM - Sparse Mode*
PLR	*Packet Loss Ratio*
PtPDV	*Peak-to-Peak Delay Variation*
RSpec	*Request Specification (RSVP)*
QoS	*Quality of Service*
RCS	*Return Channel via Satellite*
RED	*Random Early Detection*
RFC	*Request For Comments*
RIO	*RED with In and Out*
RSVP	*ReSerVation Protocol*
RT	*Real Time*
SAP	*Service Access Point*
SDR	*Sustainable Data Rate*
SE	*Shared Explicit*
SLA	*Service Level Agreement*
SLS	*Service Level Specification*
SSP	*Satellite Service Provider*
ST	*Satellite Terminal*
TCA	*Traffic Conditioning Agreement*
TCP	*Transmission Control Protocol*
TOS	*Type Of Service*
TSpec	*Traffic Specification (RSVP)*
UDP	*User Datagram Protocol*
UMTS	*Universal Mobile Telecommunication System*
VAN	*Vehicular Area Network*
VoD	*Video-on-Demand*
VPN	*Virtual Private Network*
WF	*Wildcard Filter*