UNIVERSITY OF HELSINKI

DEPARTMENT OF COMPUTER SCIENCE

INTSERV OVER DIFFSERV FOR IP QOS IN RADIO ACCESS NETWORKS

Giovanni Costa

Supervisor:

Professor Kimmo Raatikainen

INTSERV OVER DIFFSERV FOR IP QOS IN RADIO ACCESS NETWORKS

Abstract

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

Ongoing research on QoS has proven that its deployment on the Internet introduces complexity in its overall functionality. Therefore, finding an efficient solution for end-to-end QoS over Internet is a tough undertaking. It becomes even tougher when one is introducing QoS in an environment of mobile hosts and wireless networks.

The current Internet architecture offers a simple service widely known as best-effort service. The best-effort service is adequate for traditional Internet applications like e-mail, web browsing or file transfer but not for the new emerging applications like IP-telephony, multimedia conferencing or audio and video streaming, which require high bandwidth and low delay and delay variation.

The efforts to enable end-to-end QoS over IP networks have led to the development of two different architectures, the Integrated Services architecture and more recently, the Differentiated Services, which although different, support services that overcame the best-effort service. The Integrated Services model provides QoS guarantees, but due to a per flow management of traffic it introduces several scalability problems in the core network elements. Learning from the IntServ experiences, the DiffServ model is intended to avoid scalability problems, providing quality differentiation on aggregates but without strict guarantees on individual micro-flows.

The main focus of this document is to propose a combined service model offering end-to-end QoS guarantees both in a wired and in a wireless Internet environment for several kinds of real-time applications.

INTSERV OVER DIFFSERV FOR IP QOS IN RADIO ACCESS NETWORKS

Abstract

Il continuo sviluppo di applicazioni real-time sulla rete Internet e la rapida crescita dei sistemi radio mobili indica che l'architettura della futura Internet dovra' essere in grado di supportare varie applicazioni con differenti requisiti di Qualita' del Servizio (QoS), indifferentemente per utenti fissi e mobili.

La ricerca continua in questo campo (QoS) ha portato alla conclusione che il suo impiego nella rete Internet introduce complesse problematiche. Quindi trovare una soluzione efficiente per garantire end-to-end QoS su Internet e' una questione ancora aperta e si presenta ancora piu' complessa quando inserita in contesto di mobilita': reti wireless ed utenti mobili.

L'attuale architettura della Rete (Internet) offre un servizio basilare a tutti noto come best-effort. Tale servizio e' adeguato per tradizionali applicazioni Internet come: e-mail, web-browsing o file-transfer ma non per nuove emergenti applicazioni come: telefonia su IP (VoIP), conferenze multimediali o flussi audio-video (video on demand) che richiedono elevata banda bassi ritardi e piccole variazioni del ritardo (jitter).

Gli sforzi prodotti per abilitare end-to-end QoS su Internet hanno portato allo sviluppo di due diverse architetture di rete: modello a servizi integrati (IntServ) e piu' recentemente modello a sevizi differenziati (DiffServ). Sebbene diversi, questi due approcci migliorano entrambi il modello best-effort. Il modello IntServ garantisce la desiderata QoS grazie ad un approccio orientato al singolo micro-flusso, che tuttavia introduce problemi di scalabilita' nel cuore della rete, dove il numero di flussi da trattare puo' crescere notevolmente. Dall'esperienza dell' IntServ, il modello DiffServ ha come obbiettivo evitare i problemi legati alla scalabilita' fornendo una differenzizione dei servizi non basata sul singolo flusso ma su classi di traffico.

L'obbietivo principale di questa tesi e' quello di proporre una soluzione combinata dei due modelli per superare i problemi che essi introducono garantendo una soddisfacente end-to-end QoS nell'ambiente di una rete ad accesso radio per diversi tipi di applicazioni real-time.

Contents

1 Introduction

For some years now, the telecommunication world has needed a common platform for diverse communications requirements: voice, video, Web browsing, electronic mail and so on. Several network architectures have been proposed to obtain the necessary convergence for these different requirements: for example the Asynchronous Transfer Mode (ATM) network architecture, the Integrated Services Digital Network (ISDN), and the Broadband cable network architecture. Despite all these convergence efforts, the industrial community is in agreement on one common idea: *If there is to be convergence, the platform for it will be the Internet.*

Nevertheless, enabling *end-to-end Quality of Service (QoS) on the Internet* introduces complexity in several areas including applications side, network architecture, and network management and business models. These problems will become even more complex in the environment of mobile hosts. It is possible to provide adequate service for mobile hosts in a private access network, but when the corresponding node is behind some wider public networks, keeping the promised QoS becomes harder.

There are two main problems in this case: the first is the guarantee and maintenance of the QoS in an IP network, designed originally by the IETF to provide a best-effort service. The second is tied with mobility management in a radio access network, which introduces several other problems, such as handover management and radio resource administration.

In the literature we can find several proposals to solve these problems individually, and these solutions give fundamental guidelines on how to find the optimal solutions if they exist. Separating the network access domain from the core network domain, for example is a common strategy both for mobility management and for QoS administration.

Because of scalability troubles, at the edges of the network the QoS is guaranteed according to the IntServ model [1], whereas in the core network the DiffServ model [2] is adopted to guarantee the promised QoS. Over the last years QoS management has been much influenced by the interoperability of Integrated Services and Differentiated Services, therefore it has been necessary to develop mapping between these models [3].

Even mobility management show a clear-cut division between global or macro mobility protocols [18,19] when two or more administrative domains are interested in user movement, and micro or regional mobility protocols [20-29], when the mobile user is moving inside only one administrative domain.

Only recently the cooperation between different protocols has became a practicable road to new solutions. The QoS and micro-mobility mechanisms can be coupled to ensure that reservations are installed as soon as possible after a mobility event such as handover. Several proposals already exist and provide improvements in performance and efficiency, but at the expense of additional complexity and loss of independence from the underlying mobility mechanism. With regard to that written above, two roads seem practicable: independent development and improvement of the protocols already existing or combination and cooperation between these two modes.

In this document we focus our attention especially on the enhancements of mapping between IntServ and DiffServ as a solution to provide a more scalable and efficient end-to-end QoS architecture also in a radio access network. Since DiffServ is basically based on static network configurations, the use of RSVP/IntServ can bring major flexibility into the DiffServ architecture, guaranteeing bandwidth and delay requirements in a congested network, as well. On the other hand DiffServ management in the core network guarantees a more efficient utilization of the link bandwidth. We propose a possible implementation of IntServ/DiffServ interoperation utilizing the combination of an IntServ edge router coupled with one or more DiffServ core routers. Further, we will give particular attention to different traffic scheduler mechanisms in order to compare different implementation proposals for DiffServ-IntServ cooperation.

Structure of the Thesis.

This document is organised in two sections. In the first section, a description of the already existing QoS architectures is given, continuing with a mobility management overview and concluding with a brief outline of Mobile QoS Architectures. The second section will focus on the main aim of this thesis describing experimental design, results and conclusions of the work. The road map of the document is as follows:

Chapter 2 is an overview of existing QoS architectures, the Integrated Services architecture and the Differentiated services architecture, with a description of the services they provide. It also includes a discussion about the combining of Integrated Services and Differentiated Services, and even a brief overview of ATM, MPLS and RTP is presented.

Chapter 3 outlines the major mobility protocols, distinguishing between micro and macro mobility schemes, Chapter 4 outlines the principal QoS architectures, which already exist, and the scenario for the future QoS architectures. In chapter 5 we describe different traffic scheduler and congestion control mechanisms in order to explain how the QoS model can be implemented in a Linux environment. In Chapter 6 we present network configurations and test cases studied on our experimental design. Chapter 7 shows the results obtained in our experiments and finally conclusions about our study are reported in Chapter 8.

I. Quality of Service and Mobility environments

The first part of my thesis describes the most important QoS architectures in chapter 2, mobility management protocols in chapter 3, and QoS aware architectures chapter in 4, existing in the literature.

2 Quality of Service Architectures

The most significant existing IETF architectures for providing different levels of services to IP flows can be classified into three types according to their fundamental operation:

- the *Integrated Services framework* provides explicit reservations end-to-end;
- the *Differentiated Services architecture* offers hop-by-hop differentiated treatment of packets;
- *Integrated Services over Differentiated Services* guarantees simultaneously scalability and appropriate QoS for each flow but needs adequate mapping functions.

2.1 QoS Definition

Before starting with the description about the main Internet QoS Architectures it would be useful to give some definitions. QoS is a combination of two quite ambiguous words: Quality and Service. Both these terms leave room for free interpretations.

Quality is often used to describe a superior performance in terms of probability of data loss, minimal network delay, minimal jitter, efficient use of network resources; but in another sense quality can be considered synonymous with reliability and predictability of the service. Even the term service carries implicit ambiguity, often people use service to describe the communication capabilities offered to the end user of a network, but in another context this term could have other meanings, for example in a layered architecture it assumes a more specific definition.

For many applications the most important categories of QoS parameters are [5]:

- timeliness
- volume
- reliability
- criticality
- perceived quality of results
- cost

Timeliness is described in terms of the experienced delay between stimulus and response including deadlines, and earliest acceptable result times, latency, and the variation in that delay, or jitter. These characteristics are usually measured in milliseconds for multimedia systems, but the unit of time most applicable to the scale of the requirements should be used.

Volume or *bandwidth* characteristics describe the amount of data passed per time unit. This may be measured in computer-centric terms such as Mbits/sec, however it is often more meaningful to describe volume in terms of the content, e.g. frames of video per second, or discrete operations of a type per second.

Reliability describes a range of issues, including the Mean Time Between Failure (MTBF), Mean Time To Repair (MTTR), percentage time a service is available, and loss or corruption rates, measured in frames per second, bits per frame or other appropriate units.

Criticality can describe the value or importance associated with an interaction, e.g."this is for our most important customer". It is also possible to use this to provide a hierarchy of available choices when resolving conflicting requests.

Perceived quality of results is described in terms of the media, or media combination, and while they may be represented quantitatively, they stem from subjective judgments. Typical categories include picture resolution, colour depth, and audio sample rate. A change in the perceived quality of results will generally also impact the other QoS characteristics.

Cost is a slightly different category to the others described, as it is not an intrinsic part of the visible results of most transactions. Cost is generally described in terms of a monetary value per interaction, or in terms of the time spent interacting. It is often the case that cost will be used to place upper and lower limits on other characteristics.

A further important classification of QoS requirements, or more particularly the systems implementing the requirements, is the class of service provided. [6] subdivides classes of service into five levels:

• Deterministic guarantee

• Statistical guarantee

• Target objectives

• Best effort

• No guarantee

Deterministic guarantees will always be met or bettered, under all circumstances, while a *statistical guarantee* allows a percentage of time where the guarantee is not met. The last three levels provide no guarantee. A system, which takes account of target objectives, will try to satisfy requirements, with some knowledge of their implications, which could then be used to determine scheduling priority. A *best effort* system, like the Internet, would provide the same

QoS for all services i.e. with no real consideration of QoS factors, although protocols such as RSVP are moving the Internet towards several guarantees of QoS [6]. Some historic information about performance is the only guide to dependability on the service in this case. *No guarantee* is a similar class to best effort although it is unlikely any information about system performance is available with this class of service.

Finally we can find several different definitions about QoS and related QoS implementations, but we should not forget that a determining factor in the success or failure of a specific implementation is human perception, which is often variable and not objective. It is the end user who pays for the service and whose perspectives matter the most.

2.2 Useful definitions of QoS parameters

In the BRAIN project [7] the basic QoS parameters are:

- Bandwidth
- Service duration
- Maximum Delay
- Maximum Jitter
- Maximum Loss rate

1. Bandwidth parameters

The bandwidth is classically defined by three parameters: the *peak* rate, the m*edium/sustainable* bit rate, and the *variance* rate. Recently, proposals have been made to replace the medium and variance bit rate parameters by a single parameter: the *equivalent-bandwidth*, which is more appropriate with a leaky bucket strategy. The equivalent bandwidth is defined with respect to the overflow probability in a leaky bucket. If the traffic were Poisson with constant rate ?, then the probability of overflow *P(B,W)* in a leaky bucket of size *B* and service bandwidth *W* is asymptotically equal to exp($-W \times B$ / ?).

The equivalent bandwidth ?*e*, when it exists, is the quantity:

$$l\,e = W \times \lim_{B \to \infty} \frac{B}{\log P(B,W)}$$

The equivalent bandwidth is very useful in order to fix the leaky bucket parameters, namely the total bucket size *B* and the total service rate *Wc* for this class. In general ?*e* is larger than the medium rate. One possible management is to make *Wc* = (1+ e)?   ?e.

The peak rate is needed in order to fix the maximum value of the service rate of this class:

$Wc = \max\{\text{Peak rates}\}$.

The peak rate is not necessarily equal to the maximum instantaneous bit rate of the service. For example in Video-on-Demand (VoD) services, the client stores on incoming video packet in a buffer whose aim is to absorb packet delivery jitter. Packets are de-stored after a delay *T*, which corresponds to the maximum absorbable jitter.

In this case the peak rate is computed according to the following formula:

$\max\{\frac{1}{T}\int_{x}^{x+T} l(t)dt\}$ where ?(t) is the expected instantaneous bit rate of the server at time *t*.

2. Service duration

An expected service duration could be specified by the user. This can be done in two different ways: a 'fixed' one as in switched networks where the time defined by explicitly sending a reservation request and later a reservation finish message. Another way is a 'soft state' one as in RSVP IntServ basis, where the user specifies a refresh period (10 sec, for example). In this case, a service, which would actually last more than the expected duration or allowed duration, will need a refresh declaration, to be initiated by the client/application or by the local QoS broker. In the other case, reservation is finished implicitly.

3. Maximum delay

For interactivity purpose, a short delivery delay may be requested. A telephone needs a delay delivery in the order of some 10 or 100 msec (depending whether echo cancellation is used or not). This indication will be provided in the maximum delay parameter. This will help the QoS broker to fix the class of service (expedited services or other) and to fix the bucket buffer size.

4. Maximum jitter

Jitter is the variation of delay over a period of time. Video on demand and audio on demand require that any jitter in packet delivery be controlled. The client and the application according to the size of their delivery buffers will determine the maximum jitter. In a QoS architecture following a bandwidth broker-like mechanism, the QoS broker will use the maximum jitter parameter in order to fix the service class and the bucket size and bucket server rate.

5. Maximum loss rate

The loss rates (in packets) will indicate to the QoS broker the maximum tolerable packet loss. A telephone service for example can accept an unrecoverable loss rate of the order of 10 %. However, the service could support *recoverable* packet loss. In this case the service could accept packet loss provided that a higher-level data recovery protocol will retransmit packets (like in Real-Server streaming process). If the client indicates a recoverable loss rate, the QoS broker will need to expand the bandwidth requirement a little more. For example a service that needs a 1 Mbps constant bit rate with 10 % recoverable packet loss would apply for a 1.1 Mbps bandwidth reservation, if packet loss occurs. In any case (unrecoverable and recoverable) the packet loss should not exceed this requirement.

2.3 Low layer QoS Architectures

In this section we describe the two most relevant architectures that provide QoS provisioning based on link layer technology. In fact, although the IETF protocols suite represents the most spread solution in this field, other protocol architectures can be considered in order to satisfy specific customers' QoS requests.

2.3.1 Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) [8] is one of the family of high-speed, fast packet services that includes frame relay and the Switched Multimegabit Data Service (SMDS). ATM networks are connection-oriented, and inside these networks multiple service types, such as voice, video, or data, are conveyed in small, fixed-size cells.

Fast packet switching technologies take advantage of the fact that today's networks employ digital signaling and optical fiber and are, therefore, much cleaner than "traditional" packet networks. Fast packet protocols, then, perform only minimal error checking and no error correction based upon the assumption that bit errors are rarely present. If an error is, in fact, found, the offending data is discarded; it is the responsibility of higher layer end-to-end protocols to fix errors. There are many technology drivers for these new services, including the increased capability of microprocessors, the increasing speed of LANs, and the increasing bandwidth demand of graphics-based applications. But ATM goes beyond the capabilities of frame relay and SMDS by extending the basic concepts of the Integrated Services Digital Network (ISDN); namely, the goal of a single network infrastructure that can be used to offer many different types of services. Increasingly, voice, video, image, and data are all transmitted digitally and, therefore, appear the same to network switches and transmission lines; while the service characteristics are all different, the switching and transport needs are the same. ATM is the technology that has been chosen as the transport mechanism for Broadband ISDN (B-ISDN) services. B-ISDN has a relatively simple definition: it refers to those services that require channel speeds greater than 2 Mbps, which is the maximum speed of an ISDN Primary Rate Interface (PRI). B-ISDN services include videotelephony, videoconferencing, image mail, high-speed data transfer, information distribution services, user-controlled entertainment, and video on demand. But B-ISDN services are not just characterized by high speed; they may also be described by their utilization of the available bandwidth, tolerable cell loss, and the tolerable amount of end-to-end delay. Voice, for example, requires a channel speed of 64 kbps, utilizes the entire bandwidth, cannot tolerate an end-to-end delay of much more than about 20-40 ms, and requires that the end-to-end delay remain relatively constant. LAN interconnection, on the other hand, might need a channel capacity of several tens of millions of bits per second, but is very bursty in nature, resulting in low bandwidth utilization; in addition, end-to-end delays of several hundred milliseconds is acceptable and this delay may vary widely from packet to packet.

ATM is a cell-switching and multiplexing technology that combines the benefits of circuit switching (guaranteed capacity and constant transmission delay) with those of packet switching (flexibility and efficiency for intermittent traffic). It provides scalable bandwidth from a few megabits per second (Mbps) to many gigabits per second (Gbps). Because of its asynchronous nature, ATM is more efficient than synchronous technologies, such as *time-division multiplexing* (TDM).

With TDM, each user is assigned to a time slot, and no other station can send in that time slot. If a station has a lot of data to send, it can send only when its time slot comes up, even if all other time slots are empty. If, however, a station has nothing to transmit when its time slot comes up, the time slot is sent empty and is wasted. Because ATM is asynchronous, time slots are available on demand with information identifying the source of the transmission contained in the header of each ATM cell.

ATM Cell Basic Format and Cell-header format

ATM transfers information in fixed-size units called *cells*. Each cell consists of 53 octets, or bytes. The first 5 bytes contain cell-header information, and the remaining 48 contain the "payload" (user information). Small fixed-length cells are well suited to transferring voice and video traffic because such traffic is intolerant of delays that result from having to wait for a large data packet to download, among other things.

An ATM network consists of a set of ATM switches interconnected by point-to-point ATM links or interfaces. ATM switches support two primary types of interfaces: *UNI* and *NNI*. The UNI connects ATM end systems (such as hosts and routers) to an ATM switch. The NNI connects two ATM switches. An ATM cell header can be of one of two formats: *UNI* or the *NNI*, as shown in the figure below.

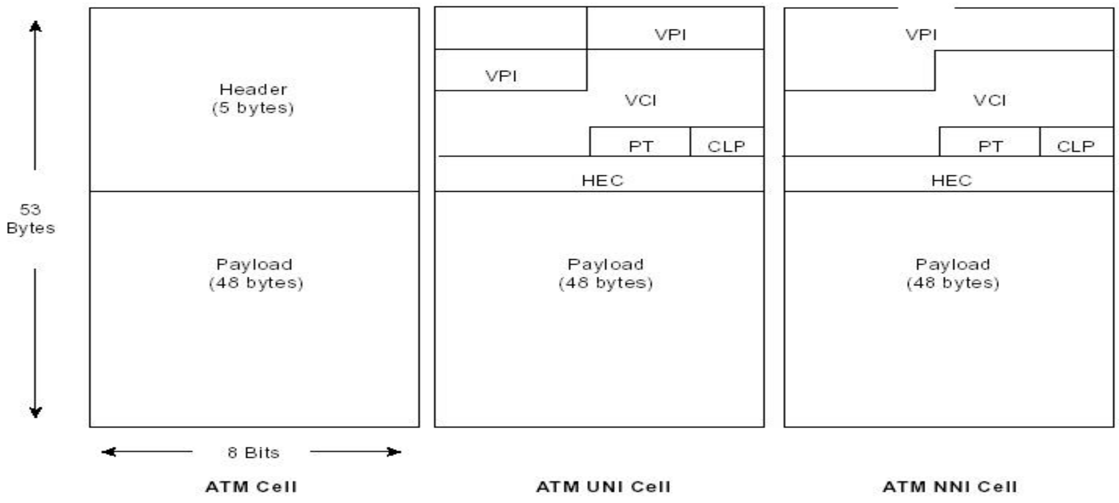


Figure 2.1 ATM Cell-header formats

Unlike the UNI, the NNI header does not include the Generic Flow Control (GFC) field.

Additionally, the NNI header has a Virtual Path Identifier (VPI) field that occupies the first 12 bits, allowing for larger trunks between public ATM switches.

In addition to GFC and VPI header fields, several others are used in ATM cell-header fields. The following descriptions summarize the ATM cell-header fields illustrated in Figure 2.x

- *Generic Flow Control (GFC)*—Provides local functions, such as identifying multiple stations that share a single ATM interface. This field is typically not used and is set to its default value.
- *Virtual Path Identifier (VPI)*—In conjunction with the VCI, identifies the next destination of a cell as it passes through a series of ATM switches on the way to its destination.
- *Virtual Channel Identifier (VCI)*—In conjunction with the VPI, identifies the next destination of a cell as it passes through a series of ATM switches on the way to its destination.
- *Payload Type (PT)*—Indicates in the first bit whether the cell contains user data or control data. If the cell contains user data, the second bit indicates congestion, and the third bit indicates whether the cell is the last in a series of cells that represent a single AAL5 frame.
- *Congestion Loss Priority (CLP)*—Indicates whether the cell should be discarded if it encounters extreme congestion as it moves through the network. If the CLP bit equals 1, the cell should be discarded in preference to cells with the CLP bit equal to zero.
- *Header Error Control (HEC)*—Calculates the checksum only on the header itself.

ATM Service Categories

Two types of ATM services exist: *permanent virtual* circuits (PVC) and *switched virtual* circuits (SVC). A PVC allows direct connectivity between sites. In this way, a PVC is similar to a leased line. Among its advantages, a PVC guarantees availability of a connection and does not require call setup procedures between switches. Disadvantages of PVCs include static connectivity and manual setup.

An SVC is created and released dynamically and remains in use only as long as data is being transferred. In this sense, it is similar to a telephone call. Dynamic call control requires a signaling protocol between the ATM endpoint and the ATM switch. The advantages of SVCs include connection flexibility and call setup that can be handled automatically by a networking device. Disadvantages include the extra time and overhead required to set up the connection.

ATM networks are fundamentally connection-oriented, which means that a *virtual channel* (VC) must be set up across the ATM network prior to any data transfer. (A virtual channel is roughly equivalent to a virtual circuit.)

Two types of ATM connections exist: *virtual paths*, which are identified by virtual path identifiers, and *virtual channels*, which are identified by the combination of a VPI and a *virtual channel identifier* (VCI). A virtual path is a bundle of virtual channels, all of which are switched transparently across the ATM network on the basis of the common VPI. All VCIs and VPIs, however, have only local significance across a particular link and are remapped, as appropriate, at each switch. A transmission path is a bundle of VPs. Figure 2.2 illustrates how VCs concatenate to create VPs, which, in turn, concatenate to create a transmission path.

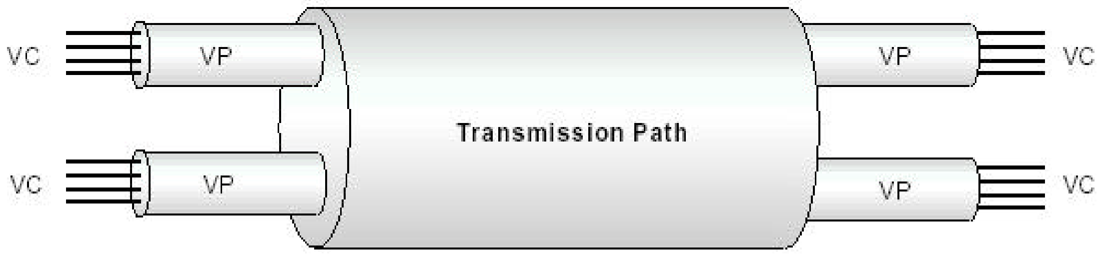


Figure 2.2 VC-VPs relation

ATM offers the user the capability to request dynamically different service levels during SVC connection setup. Nevertheless it is not uncommon to discover that SVCs also are manually provisioned in many instances, like PVC connections.

QoS classes of service

Currently five ATM Forum-defined service categories exist:

- Constant bit rate (CBR)

- Real-time variable bit rate (rt-VBR)
- Non real-time variable bit rate (nrt-VBR)
- Available bit rate (ABR)
- Unspecified bit rate (UBR)

The *CBR service classes* is intended for real-time applications, i.e. those requring tightly constrained delay and delay variation, as would be appropriate for voice and video applications. The consistent availability of a fixed quantity of bandwidth is considered appropriate for CBR service. Cells which are delayed beyond the value specified by CTD (cell transfer delay) are assumed to be of significantly less value to the application.

The *real time VBR service class* is intended for real-time applications, i.e., those requiring tightly constrained delay and delay variation, as would be appropriate for voice and video applications. Sources are expected to transmit at a rate, which varies with time. Equivalently the source can be described as bursty. Cells, which are delayed beyond the value specified by CTD are assumed to be of significantly less value to the application. Real-time VBR service may support statistical multiplexing of real-time sources, or may provide a consistently guaranteed QoS.

The *non-real time VBR service class* is intended for non-real time applications which have 'bursty' traffic characteristics and which can be characterized in terms of a GCRA. For those cells, that are transferred, it expects a bound on the cell transfer delay. Non-real time VBR service supports statistical multiplexing of connections.

The *UBR service class* is intended for delay-tolerant or non-real-time applications, i.e., those, which do not require tightly constrained delay and delay variation, such as traditional computer communications applications. Sources are expected to transmit non-continuous bursts of cells. UBR service supports a high degree of statistical multiplexing among sources. UBR service includes no notion of a per-VC allocated bandwidth resource. Transport of cells in UBR service is not necessarily guaranteed by mechanisms operating at the cell level. However it is expected that resources will be provisioned for UBR service in such a way as to make it usable for some set of applications. UBR service may be considered as interpretation of the common term "best effort service".

ABR definition. Many applications have the ability to reduce their information transfer rate if the network requires them to do so. Likewise, they may wish to increase their information transfer rate if there is extra bandwidth available within the network. There may not be deterministic parameters because the users are willing to live with unreserved bandwidth. To support traffic from such sources in an ATM network will require facilities different from those for Peak Cell Rate of Sustainable Cell Rate traffic. The ABR service is designed to fill this need.

From the definitions reported above and from the definition of the ATM traffic and topology parameters we can distinguish two types of ATM QoS classes: specified QoS class and unspecified QoS class. Only the former explicitly specifies performance parameters. In an unspecified QoS class, no objective is specified for the performance parameters. A correlation for QoS classes to ATM service categories results in a general set of service classes:

Service class A. Circuit emulation, constant bit rate video.

Service class B. Variable bit rate audio and video.

Service class C. Connection oriented data transfer.

Service class D. Connectionless data transfer.

Currently the following QoS classes are defined:

QoS Class 1. Supports a QoS that meets service class A performance requirements. This should provide performance comparable to digital private lines.

QoS Class 2. Supports a QoS that meets service class B performance requirements. Should provide performance Acceptable for packetized video and audio in teleconferencing and multimedia applications.

QoS Class 3. Supports a QoS that meets service class C performance requirements. Should provide performance for interoperability connection-oriented protocols, like Frame Relay.

QoS Class 4. Supports a QoS that meets service class C performance requirements. Should provide performance for interoperability connectionless protocols, like IP.

The following mapping has been proposed between ATM and IIS for IP [52].

ATM Service	Internet Service
CBR or rtVBR	Guaranteed Service
nrtVBR or AVB (minimum cell rate)	Controlled Service
UBR or ABR	Best effort

Table 2.1 ATM IIS Mapping

2.3.2 Multi-Protocol Label Switching

The Multi-Protocol Label Switching (MPLS) [7] is specified by the IETF mainly to be used in combination with the DiffServ concept and is an advanced forwarding scheme that extends routing with respect to packet forwarding and path controlling.

In a connectionless network layer protocol, a packet traveling from one router to the next, causes each router to make an independent forwarding decision for that packet. That is, each router analyzes the packet's header, and each router runs a network layer routing algorithm. Each router independently chooses a next hop for the packet, based on its analysis of the packet's header and the results of running the routing algorithm.

Packet headers contain considerably more information than is needed simply to choose the next hop. Choosing the next hop can therefore be thought of as the composition of two functions. The first function partitions the entire set of possible packets into a set of "Forwarding Equivalence Classes (FECs)". The second maps each FEC to a next hop. Insofar as the forwarding decision is concerned, different packets, which get mapped into the same FEC are indistinguishable. All packets, which belong to a particular FEC, and which travel from a particular node, will follow the same path.

In conventional IP forwarding, as the packet traverses the network, each hop in turn reexamines the packet and assigns it to a FEC. In MPLS, the assignment of a particular packet to a particular FEC is done just once, as the packet enters the network. The FEC to which the packet is assigned is encoded as a short fixed length value known as a "label". When a packet is forwarded to its next hop, the label is sent along with it; that is, the packets are "labeled" before they are forwarded. The packet forwarding is used to create topology driven paths through the network.

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

The path controlling can be achieved by using traffic engineering. In this case the LSP establishment, maintenance and release is strictly controlled and managed by using signaling. This signaling carries information related to the required characteristics for each LSP. Each node must ensure that these requirements are satisfied. An LSP can include the following requirement characteristics: *Path Selection* (based on QoS constraints), *Delay and delay variation*, *Bandwidth* including sustained peak data rates and maximum burst sizes.

Multi-protocol label switching plays an important role in many networks to support QoS services. The MPLS functionality fit the requirements of IntServ architecture well. IntServ requires the establishment of a path between the ingress and the egress point, within the MPLS environment this path can correspond to an LSP. The advantage for the IntServ routers is that the reservation state and forwarding directive can be inferred from the incoming MPLS label, thereby reducing the overload required to check the whole IP header. Nevertheless the use of MPLS LPSs as a mechanism of supporting IntServ reserved paths does not introduce important advantages in terms of scaling properties.

There are two main approaches in MPLS over the DiffServ implementation field. The first consist on the direct mapping between each DiffServ codepoint and a new MPLS label-switched path. This results in one label-switched path per Forwarding Equivalence Class per DiffServ per- hop- behavior.

An alternative approach is to group a number of PHBs that share some scheduling behavior into a single LSP that supports a scheduling aggregate (SA) per LSP. The resulting capability to manage traffic flows in a DiffServ behavior aggregate basis is a very powerful tool, enabling the network to manage traffic that shares a common service requirement in a single managed unit.

2.4 Integrated Services

The Integrated Services (IS) model [1] has been defined by an IETF working group to offer a set of extensions to support best-effort traffic delivery, currently in place in the Internet, as well as the real time applications. The Internet in fact, as originally conceived, offers only a very simple quality of service (QoS), point-to-point best-effort data delivery.

Because of routing delays and congestion losses, real-time applications do not work very well on the current best-effort Internet. Video conferencing, video broadcasting, and audio conferencing software need guaranteed bandwidth to provide video and audio of acceptable quality. To support these service requirements it has been necessary to modify the Internet infrastructure to provide control over end-to-end packet delay and bandwidth administration. Further, from the beginning this extension was designed for multicast purposes.

2.4.1 Integrated Services model

To support the Integrated Services model, an Internet router must be able to provide an appropriate QoS for each flow, in accordance with the service model. The router function that provides different qualities of service is called *traffic control*. It consists of the following components: the admission control, the packet classifier, the packet scheduler and the reservation setup protocol, see figure 2.3. The router can be divided into two broad functional categories: to the forwarding path below the thick horizontal line and to the background code above the line. The forwarding path of the router is executed for each packet and it is divided into three sections: the input driver, the Internet forwarder and the output driver. The background routines control the forwarding path and handle reservation requests.

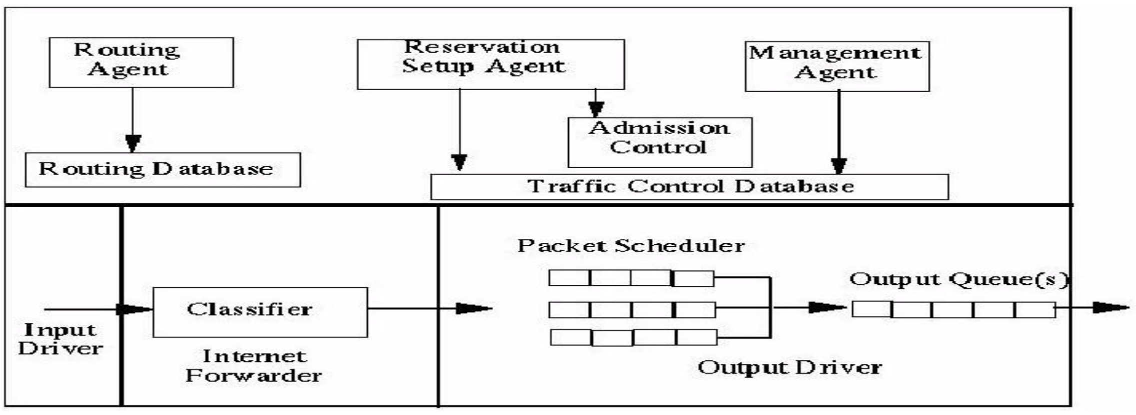


Figure 2.3 Implementation model [1]

Admission and policy control The admission control contains the decision algorithm that a router uses to determine if there are enough routing resources to accept the requested QoS for a new flow. If there are not enough free routing resources, accepting a new flow would impact on earlier guarantees and the new flow must be rejected. If the new flow is accepted, the reservation instance in the router assigns the packet classifier and the packet scheduler to reserve the requested QoS for this flow.

Admission control is invoked at each router along a reservation path, to make a local accept/reject decision at the time a host requests a real-time service. The admission control algorithm must be consistent with the service model.

Admission control is sometimes confused with *policy control*, which is a packet-by-packet function, processed by the packet scheduler. It ensures that a host does not violate its promised traffic characteristics. Nevertheless, to ensure that QoS guarantees are honored, the admission control will be concerned with enforcing administrative policies on resource reservations. Several policies will be used to check the user authentication for a requested reservation. Unauthorized reservation requests can be rejected. As a result, admission control can play an important role in accounting costs for Internet resources in the future.

Packet classifier The packet classifier identifies packets of an IP flow in hosts and routers that will receive a certain level of service. To realize effective traffic control, each incoming packet is mapped by the classifier into a specific class. All packets that are classified in the same class get the same treatment from the packet scheduler. The choice of a class is based upon the source and destination IP address and port number in the existing packet header or an additional classification number, which must be added to each packet. A class can correspond to a broad category of flows. For example, all video flows from a video-conference with several participants can belong to one service class. But it is also possible that only one flow belongs to a specific service class.

Packet scheduler The packet scheduler manages the forwarding of different packet streams in hosts and routers, based on their service class, using queue management and various scheduling algorithms. The packet scheduler must ensure that the packet delivery corresponds to the QoS parameter for each flow. A scheduler can also police or shape the traffic to conform to a certain level of service. The packet scheduler must be implemented at the point where packets are queued. This is typically the output driver level of an operating system and corresponds to the link layer protocol.

2.4.2 Resource Reservation Protocol

The fourth component of the original implementation framework is a reservation set-up protocol. The IntServ model uses the resource reservation protocol (RSVP) [10] to manage the provision of QoS in an IP network. IntServ and RSVP provide unidirectional resource reservations on a per-flow basis. Is interesting to note that RSVP is a signaling protocol, which may carry IntServ information, but is not the only one: RSVP and IntServ are separable, but the current prevailing model is based on a combined RSVP/IntServ.

In this model the sender of a flow first sends a PATH message to the receiver. The message is updated at every router on the path. The receiver responds with a RESV message and indicates the resources needed at every hop to support the forthcoming flow. Any router on the end-to-end path may deny the flow if resources are scarce. If the sender receives the RESV message the resources for supporting the flow requirements have been granted. The figure below shows the reference model about RSVP.

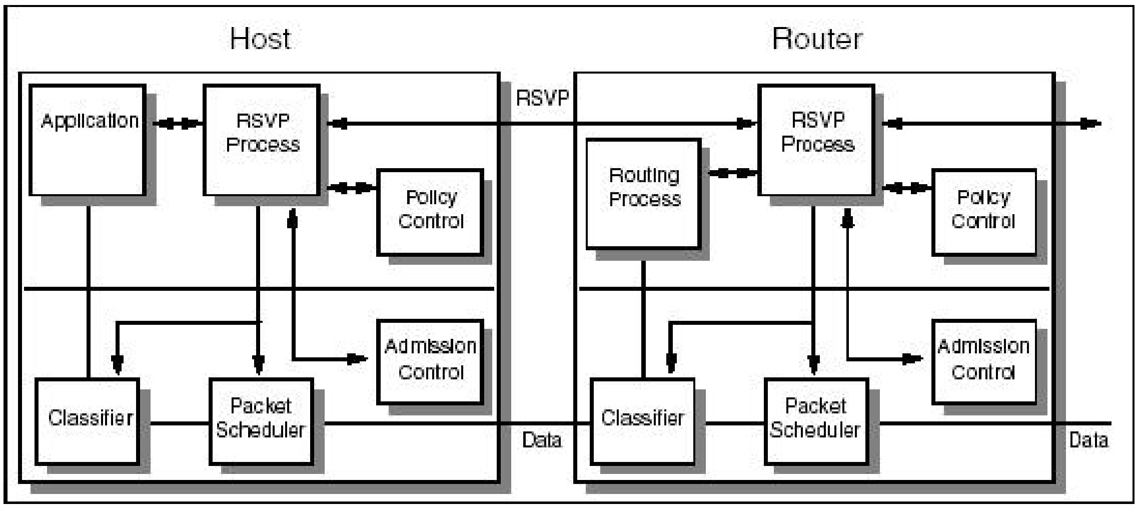


Figure 2.4 Host-Router model in RSVP [10]

As shown in Figure 2.4, the application that wants to send data packets in a reserved flow communicates with the reservation instance RSVP. The RSVP protocol tries to set up a flow reservation with the requested QoS, which will be accepted if the application fulfills the policy restrictions and the routers can handle the requested QoS. RSVP advises the packet classifier and packet scheduler in each node to process the packets for this flow adequately. If the application now delivers the data packets to the classifier in the first node, which has mapped this flow into a specific service class complying with the requested QoS, the flow is recognized with the sender IP address and is transmitted to the packet scheduler. The packet scheduler forwards the packets, dependent on their service class, to the next router or, finally, to the receiving host. Because RSVP is a simplex protocol, QoS reservations are only made in one direction, from the sending node to the receiving node. If the application wants to cancel the reservation for the data flow, it sends a message to the reservation instance, which frees the reserved QoS resources in all routers along the path. The resources can then be used for other flows.

Because the IS model provides per-flow reservations, each flow is assigned a flow descriptor. The flow descriptor defines the traffic and QoS characteristics for a specific flow of data packets. In the IS specifications [11], the flow descriptor consists of a filter specification (filterspec) and a flow specification (flowspec), as illustrated in Figure 1.5

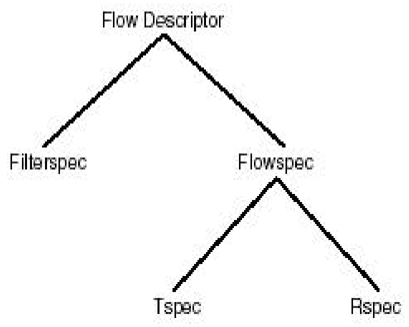


Figure 1.5 Flow Descriptor model [11]

The filterspec is used to identify the packets that belong to a specific flow with the sender IP address and source port. The information from the filterspec is used in the packet classifier. The flowspec contains a set of parameters that are called the *invocation information*. It is possible to sort the invocation information into two groups:

• Traffic Specification (Tspec)

• Service Request Specification (Rspec)

The Tspec describes the traffic characteristics of the flow identified by the filterspec, further Tspec carries traffic information usable by either the Guaranteed or Controlled-Load QoS control services.

The Service Request Specification (Rspec) specifies the Quality of Service the application wants to request for a specific flow.

In particular a path message carries the following objects:

SENDER_TEMPLATE: It identifies the sender and consists of the sender IP address and the source port number.

SENDER_TSPEC: It describes the traffic characteristic of the flow generated by the sender.

ADSPEC: It describes the aggregate QoS characteristics of the path.

PHOP: It identifies the previous hop, which sent this path message.

Instead a Resv message contains a FLOWSPEC object, which consists of two sets of numeric parameters, as described before in the general model:

RSPEC that defines the desired QoS of the data flow

TSPEC that describes the traffic characteristics of the data flow

Two different possible styles for reservation setup protocols are defined [1], the "hard state" (HS) approach (also called "connection-oriented"), and the "soft state" (SS) approach (also called "connectionless"). Under the HS approach, this state is created and deleted in a fully deterministic manner by cooperation among the routers. Once a host requests a session, the "network" takes responsibility for creating and later destroying the necessary state. RSVP takes the SS approach, which regards the reservation state as cached information that is installed and periodically refreshed by the end hosts. The SS approach was chosen to obtain the simplicity and robustness that have been demonstrated by connectionless protocols such as IP [1].

Another important function implemented by RSVP is the merging flowspecs, in fact when there are multiple reservation requests from different next hops for the same session with the same filter spec, the RSVP installs only one reservation on that interface, and we can say that the flowspecs have been merged. In RSVP, protocol overhead is reduced, by merging control messages.

2.4.3 Classes of Service

The Integrated Services model uses different classes of service that are defined by the Integrated Services IETF working group. The current IS model includes: the *Guaranteed Service* [12], and the *Controlled Load Service* [13].

The Integrated Services definition for *controlled-load service* attempts to provide end-to-end traffic behavior that closely approximates traditional best-effort service within the environmental parameters of an unloaded or lightly utilized network conditions. In other words, the service is better than what best-effort can offer in a congested network. The applications using this service may assume that a high percentage of the transmitted packets will be successfully delivered by the network, i.e. the amount of dropped packets should be equivalent to the error rate of the transmission media. The latency introduced into the network will not greatly exceed the minimum delay experienced by any successfully transmitted packet. The controlled-load service does not make use of specific target values for such control parameters as delay or loss. The controlled-load service is to be used with tolerant real-time applications, e.g. unidirectional audio and video casts, which do not need absolute delay requirements, but are able to buffer the received data.

The *guaranteed service class* provides a framework for delivering traffic for applications that require a guarantee on both available bandwidth and an upper delay bound from the network. The term 'guaranteed bandwidth' implies that no queuing losses due to buffer overflow will occur if the flow stays within the bounds of its specified traffic parameters. 'Guaranteed delay' refers to the upper bound of end-to-end delay that the datagram will experience on the path from the sender to receiver. Guaranteed service does not attempt to minimize jitter, it merely controls the maximum queuing delay. Guaranteed service represents one extreme end-to-end delay control for networks. In order to provide this high level of assurance, guaranteed service is typically useful only if every network element along the path supports it. Moreover, the dynamic of this service requires that the set-up protocol is used to request services and to update them to intermediate routers.

The major problem in the Integrated Services model is tied with the per-flow-state and per-flow-processing approach. If the per-flow-processing rises dramatically, it could become a scalability concern for large networks; in fact the resource requirements (computational processing and memory consumption) for running per-flow resource reservations on routers increase in direct proportion to the number of separate reservations that need to be accommodated. Therefore this approach represents a good solution only at the edges of the network.

2.5 Differentiated Services

Differentiated Services (DiffServ) [2] is a direct extension to the work done by Integrated Services and RSVP working groups. The goal is basically the same for both architectures: to provide quality of service and service differentiation in IP networks. While IntServ provides per-flow guarantees, Differentiated Services follows the philosophy of mapping multiple flows into a few service levels, an approach sometimes referred to as Class of Service (CoS). The aggregation results in *more scalable but also more approximate service* to user flows and the lack of control signaling can be seen as a weakness in view of the total operation. The basic principle of DiffServ is to keep the architecture as simple as possible and therefore it provides service differentiation in one direction only. Unlike RSVP, DiffServ is sender orientated, which means that the traffic sender is responsible for the QoS of the transmission. The other main difference between DiffServ and IntServ is that IntServ offers service classes, while DiffServ specifies the building blocks from which the services can be built.

A central component of DiffServ architecture is the Service Level Agreement (SLA). The SLA is a service contract between a customer and a service provider, which specifies the details of the traffic classifying and the corresponding forwarding service a customer should receive. A customer may be a user organization or another DiffServ domain. The service provider must assure that the traffic of a customer, with whom it has an SLA, gets the contracted QoS. Therefore, the service provider's network administration must set up the appropriate service policies and measure the network performance to guarantee the agreed traffic performance. To distinguish the data packets from different customers in DS-capable network devices, the IP packets are modified in a specific field. A small bit-pattern, called the *DS field*, see next section, in each IP packet is used to mark the packets that receive a particular forwarding treatment at each network node.

2.5.1 DS field

Differentiated Services architecture consists of a small, well-defined set of building blocks, which are deployed in network nodes. One of the most important blocks is the DS field, which is used to select a specific packet-forwarding treatment. The DS field uses the space of the former TOS octet in the IPv4 IP header and the traffic class octet in the IPv6 header. Nevertheless the DiffServ model is quite different from the earlier interpretation of the IP TOS field, because in this architecture the DS field is marked by the network, upon ingress to the network. All network traffic inside of a domain receives a service that depends on the traffic class that is specified in the DS field.

This field is six bits long and the value of the field is called DS codepoint (DSCP), see figure 2.6. DSCP is used to select the PHB at each node. A two-bit currently unused (CU) field must be ignored in DS treatment. This field is used for explicit congestion notification, and it is beyond the scope of DiffServ.

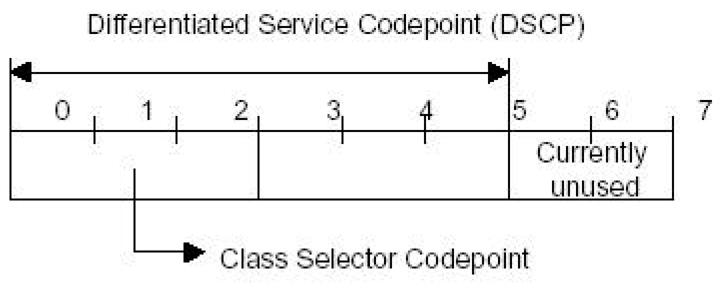


Figure 2.6 DS codepoint

The six-bit DSCP field is divided into three pools for the purpose of codepoint assignment and management. Pool 1 with 32 codepoints is assigned to Standard Action, pool 2 of 16 codepoints is reserved for experimental or local use and pool 3 also of 16 codepoints is initially available for experimental or local use, but can be utilized for standardized assignments if pool 1 is at some point exhausted.

Pool 1	Codepoint space	Assignment Policy
1	xxxxx0	Standard Action
2	xxxx11	EXP/LU
3	xxxx01	EXP/LU or Standard Action

Table 2.2 DS codepoints [14]

There are 64 possible DSCP values, but there is no such limit on the number of PHBs. In a given network domain, there is a locally defined mapping between DSCP values and PHBs. Recommended codepoints should map to specific, standardized PHBs, but network operators may also choose alternative mappings.

2.5.2 Per-hop behavior

The other main building block of DiffServ is Per-Hop Behavior (PHB). A per-hop behavior is defined as “description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate” [2]. It is important to note that PHB is not a service, but reasonable services can be built by implementing similar PHBs in every node along the traffic path. Another significant feature of the PHB is that the forwarding treatment is applied to traffic aggregates, not to individual micro flows. The term ‘micro flow’ refers to a distinguishable stream of related datagrams from a single user. As a technical term PHB is a combination of forwarding, classification, scheduling and drop behavior at each hop. Moreover, PHB is a term that service providers, administrators and vendors can use for practical discussion. PHBs may be specified in terms of their resource (e.g. bandwidth) priority relative to other PHBs.

As said before, the DS field uses six bits to determine the Differentiated Services Code Point (DSCP). Each node into the network to select the PHB will use this code point. Each DS-capable network device must have information on how packets with different DS fields should be handled. In the DS specifications, this information is called the *per-hop behavior* (PHB). Further, the PHB can be described as a set of parameters inside of a router that can be used to control how packets are scheduled onto an output interface. This can be a number of separate queues with priorities that can be set, parameters for queue lengths, or drop algorithms and drop preference weights for packets. At present, there are three sets of PHBs specified by the IETF DiffServ working group: the *Class Selector* (CS) PHB [14], the *expedited Forwarding* (EF) PHB [15] and the *Assured Forwarding* (AF) [16].

The *Class Selector* PHB attempts to preserve the semantics to the TOS IP Precedence field (the first 3 bits in the old-style IP TOS octet). The DSCP values ‘xxx000’ are reserved for Class Selector PHB. These eight CS Codepoints must yield at least two independently forwarded classes of traffic. A CS codepoint with a large numerical value should get a higher relative order than a packet marked with a lower numerical value. CS PHB tries to avoid contradictions between the old and new definitions of the DS field. Moreover, it can be used to build a simple prototype DiffServ architecture since the existing routers support the usage of an IP precedence field.

The class Selector PHB can be realized by a variety of queuing disciplines, e.g. WFQ, CBQ or priority queuing, see Chapter 5. The definition of CS PHB does not specify any traffic-condition functions. It is therefore not reasonable to build services using only a CS PHB group.

Instead, a service provider can avoid upgrading all nodes and still offer some level of differentiation by using existing routers with the CS as an interior node and by upgrading only boundary nodes. The nature of this service type is relative and is therefore suitable for applications like Web browsing.

The objective of the *Expedited Forwarding* PHB is to support a behavior aggregate of low loss, low latency, low jitter, assured bandwidth end-to-end service through a DiffServ network. This service attempts to resemble a virtual leased–line service across an IP network. Since packet loss, latency and jitter are all due to the queues that the traffic experiences while traveling in a network, the way to provide low latency and jitter service is to ensure that the traffic aggregate sees no queues at all or only very small ones. The queues are formed when the traffic arrival rate exceeds the departure rate. To ensure that no queues will occur for some traffic aggregate, the aggregate's maximum arrival rate has to be smaller than the aggregate's configured departure rate. Thus EF PHB is defined as a forwarding treatment of the traffic aggregate where the configured packet data rate must equal or exceed a real departure data rate. The EF traffic should receive the configured rate independently of the intensity of any other traffic attempting to transit the node. This implies strict bit-rate control in the boundary nodes (i.e. traffic that exceeds the negotiated rate must be discarded) and as quick forwarding in the interior nodes as possible. [15]

EF can be implemented by several different queue-scheduling mechanisms, see chapter 5. The simple priority queuing will work appropriately as long as the higher priority queue, i.e. EF, does not starve the lower priority queues. This could be accomplished by using a some kind of rate police (e.g. token bucket) in each priority queue that defines how much the queue can starve the other traffic. The EF PHB can provide a leased line service over the public Internet, so that the users feel that they have a fixed bit rate pipe between their intranets. Since EF provides low latency service it is suitable for several real-time applications, like the IP phone and videoconference.

The *Assured Forwarding* PHB Group is a collection of PHBs that assures a deliver service conformant to a given service profile, or (TCA) Traffic Conditioning Agreement. The assured Forwarding (AF) group offers different levels of forwarding assurances. The current AF specification provides delivery of IP packets in four classes, each with three drop precedence levels, see table 2.3. Packets within one class must be forwarded independently of the packets in other AF classes and a DiffServ node must allocate resources to each implemented AF class, but all classes are not required to be implemented. The level of the forwarding assurance of an IP packet depends on the amount of bandwidth and buffer space that has been configured to the AF class, the current load of the AF class and the drop precedence of the packet. Another important feature of the AF PHB group is that the IP packets of the same microflow should not be reordered if they belong to the same AF class. Like the other PHBs, AF is not an end-to-end service model, but a building block for service building. Since QoS depends on the other traffic entering the node, the service is likely to be relative. As a result, the AF PHB group does not guarantee any quantitative characteristics. On the other hand, by over-provisioning an AF class even the low loss and low latency services can be implemented. [16]

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

Table 2.3 AF Classes [16]

The definition of the AF PHB group does not specify any queuing discipline for the implementation. However, the CBQ queuing mechanism allocates bandwidth for different queues and is therefore exactly what AF requires. The drop precedence inside the class needs active queue management. The Random Early Drop (RED) algorithm with multiple threshold levels is able to handle that, see chapter 5 The implementation of the AF PHB group would thus consist of a CBQ scheduler that divides the bandwidth for PHB classes and a RED algorithm which drops packets inside the queues according to the drop precedences. In addition to queuing management, traffic marking and metering is needed to implement the AF PHB group. The boundary nodes measure the incoming packet stream and traffic that exceed the definition in SLA is marked for lower importance level. Unlike EF PHB, the excess or bursty traffic is not immediately dropped but is instead delivered with smaller forwarding probability [16].

2.5.3 DiffServ Domains

The setup of QoS guarantees is not made for specific end-to-end connections but for well-defined Differentiated Services domains. The IETF working group defines a Differentiated Services domain as a contiguous portion of the Internet over which a consistent set of Differentiated Services policies are administered in a coordinated fashion. It can represent different administrative domains or autonomous systems and different network technologies.

A DS domain consists of boundary components that are used to connect different DS domains to each other and interior components that are only used inside of the domains. The traffic conditioning is done inside a boundary node by a *traffic conditioner*. It classifies, marks, and possibly conditions packets that enter or leave the DS domain. The most recent conceptual model of a DiffServ router proposed in literature [40] is composed of the following key elements:

Traffic Classification elements Classification is performed by a classifier element. A classifier selects packets based on their packet header and forwards the packets that match the classifier rules for further processing. The DS model specifies substantially two types of packet classifiers:

- Multi-field (MF) classifiers, which can classify on the DS field as well as on any other IP header field, for example, the IP address and the port number, like an RSVP classifier. A common type of MF classifier is a 6-tuple classifier that classifies based on six fields from IP and TCP or UDP headers (destination address, source address, IP protocol, source port, destination port and DSCP).
- Behavior Aggregate (BA) classifiers, which uses only the DiffServ codepoint (DSCP) in a packet's IP header to determine the logical output stream to which the packet should be directed.

Only recently other types of classifiers are defined: the free-form classifier for example, which uses a set of user definable arbitrary filters combinable into filter groups to form very powerful filters.

Metering functions Traffic meters measure whether the forwarding of the packets that are selected by the classifier corresponds to the traffic profile that describes the QoS for the SLA between customer and service provider. A meter passes state information to other conditioning functions to trigger a particular action for each packet, which either does or does not comply with the requested QoS requirements.

In the most recent DiffServ example, three levels of conformance are discussed in terms of colors, with green representing conforming, yellow representing partially conforming and red representing non-conforming. These different conformance levels may be used to trigger different queuing, marking or dropping treatments.

Some examples of possible meters are the following: Average Rate Meter, Exponential Weighted Moving Average (EWMA) Meter, Two-Parameter Token Bucket Meter, Multi-Stage Token Bucket Meter.

Action elements The classifiers and meters described before are generally used to determine the appropriate action to apply to a packet. The set of possible action that can be applied inclcude: Marking, Absolute Dropping, Multiplexing, Counting, Null action.

DSCP Marker: DSCP markers set the DSCP field of the incoming IP packets to a particular bit pattern. The PHB is set in the first 6 bits of the DSCP field so that the marked packets are forwarded inside of the DS domain according to the SLA between service provider and customer.

Absolute dropper: Absolute Droppers simply discard packets. It is worth mentioning that the absolute droppers are not the only elements that discard packets**,** another element is an Algorithmic Dropper in the queuing disciplines for example.

Multiplexor; Sometimes it is necessary to multiplex traffic streams into an element with a single input. A multiplexor is a simple logical device for merging traffic streams.

Counter The counter simply monitors data packets processed by a router. The resulting statistics might be used later for customer billing, service verification or network engineering purposes.

Null action A null action element performs no action on the packet. Such an element is useful to define in the event that the configuration or management does not have the flexibility to omit an action element in a datapath segment.

Queuing Elements Queuing disciplines implemented in the queuing elements modulate the transmission of packets belonging to different traffic streams and determine their ordering, possibly storing or discarding them. Usually packets are stored either because available resources are insufficient to immediately forwarding them, or because these elements are used to alter the temporal properties of a traffic stream.

In this model packets are discarded because of buffering limitations, because a buffer threshold is exceeded, in response to a reactive control protocol signaling, or because a meter exceeds a configured profile. Various mechanisms of queuing disciplines are present in the literature, and will be discussed later in this document.

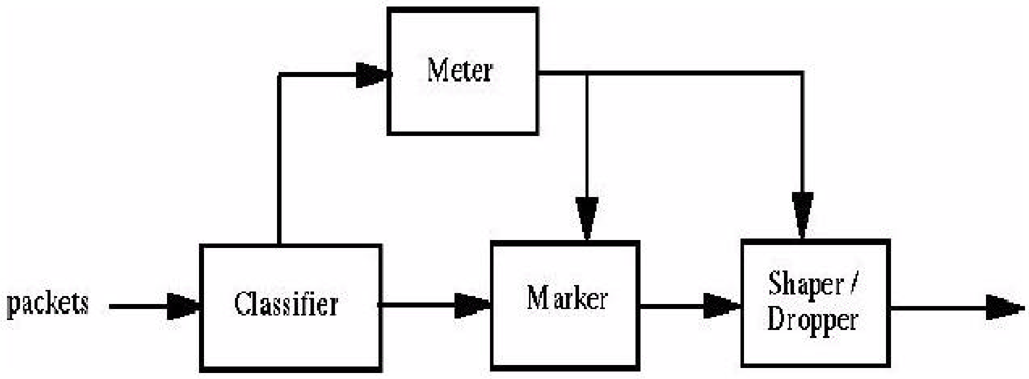


Figure2.7 DS Traffic Conditioner

Finally we can conclude that the DiffServ architecture is certainly promising, but there are a lot of open issues related to the dynamic resource provisioning that need to be defined and researched. For example, since the DiffServ does not (yet) specify services, ISPs have a very important role in the realization of DiffServ. The risk is that ISPs implement PHBs differently, so that predictable end-to-end services can not be achieved. In addition, the network provisioning might be a very difficult task, since no signaling is used and nodes have to be pre-configured. Especially EF might be very problematic if dynamic end-points are allowed. On the other hand, AF provides more elastic possibilities, but the services built from AF can be too ambiguous. When DiffServ services are provided for end users it is quite clear that technical terms such as AF or EF are not mentioned. It might be difficult to sell these services (assured service) if their clear advantages, e.g., guaranteed bandwidth, can not be demonstrated.

One interesting issue of the DiffServ analysis will be TCP behavior, since DiffServ is specified to be only unidirectional. If both end points do not have a similar QoS service, the presumptive service will not be realized when some acknowledgements are dropped. Some simulations [52] have already been made and they imply that acknowledgements also need some QoS service.

2.6 Integrated Services over Differentiated Services

The Telecommunication community concords on the idea that neither the IntServ model nor the DiffServ model are adapted to support a multi-service network platform. This observation leads to a proposal to combine the two models, using the IntServ control mechanism at the edge of the network and DiffServ within the core network.

The Integrated Services architecture provides a means for the delivery of end-to-end Quality of Service applications over heterogeneous networks. To support this end-to-end model, the IntServ architecture must be supported over a wide variety of different types of network elements. In the framework by the IETF network WG, a network that supports Differentiated Services may be viewed as a network element in the total end-to-end path. In the literature [4], the most employed reference model for supporting the IntServ/DiffServ cooperation includes a DiffServ region in the middle of two regions supporting the IntServ model. The interface network elements between those regions are generally called: Edge routers (ER) and Border router (BR). The former located adjacent to the DiffServ network regions, and the latter located in the DiffServ network region. Figure 2.8 shows this reference model.

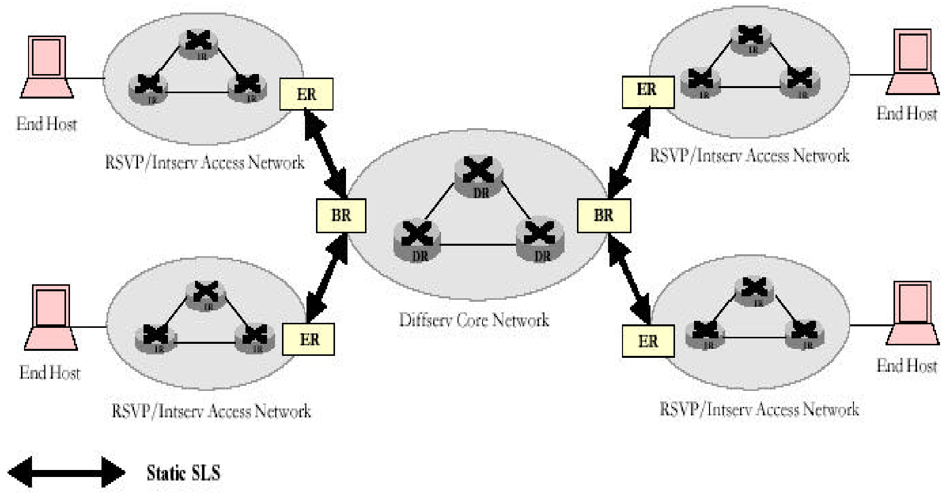


Figure 2.8 The IntServ/DiffServ Architecture [3]

Several options exist for management of resources in a DiffServ region to meet the end-to-end QoS requirements from users. The most important are:

- Resources statically provisioned
- Resource dynamically provisioned by RSVP
- Resource dynamically provisioned by other means (for example Bandwidth broker)

The first option consists of a static contract (service level specification SLS) between the customers of the DiffServ region and the owners of the same DiffServ region. The mechanism proposed is that the application uses RSVP to request admission into the network. If the request was accepted by the IntServ capable components of the network, this request has been mapped into a compatible aggregate service class within the DiffServ network.

The DiffServ network uses MF admission classifiers, where the MF classification is based on the IntServ network flow specification. The service specification of the IntServ resource reservation is mapped to an aggregate DiffServ behavior aggregate, and the MF admission filter will mark all such packets into the associated aggregate service class.

The whole mechanism of the end-to-end QoS can be summarized:

- The sending host generates an RSVP path message
- The path message is carried across the IntServ network normally
- The path message is carried transparently through the DiffServ transit network.
- The path message is carried across the IntServ terminating network.
- The receiving host generates an RSVP Resv response back to the sender through the terminating IntServ.
- At the interface with the DiffServ network, a DiffServ admission control system (DACS) is invoked. If the request fits the resource availability, the DACS record this reservation and the associated DSCP, and pass the Resv message back through the DiffServ network .
- The Resv message is passed to the host through the IntServ network.
- The host is then permitted to begin the traffic flow.

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

By contrast, when the DiffServ region is RSVP aware the negotiation of an SLS is dynamic. The use of RSVP within the DiffServ region improves the general resource management of the network and offers support to a dynamic provisioning and a topology-aware admission control mechanism. Various proposals have been proposed in the literature for supporting those objectives: aggregated RSVP, per flow-RSVP and bandwidth brokers.

In the aggregated RSVP approach, the boarder routers interact with the core network and the other border routers using aggregated RSVP to reserve resources between the edge and the DiffServ region. This approach offers a dynamic and topology-aware admission control without introducing excessive additional scalability concerns.

The per-flow RSVP approach provides better benefits than the previous approach, in term of efficient use of the network, but the demands on RSVP signaling resources may be significantly higher. Finally a different approach is based on the bandwidth broker (BB) mechanism. A brief description of this mechanism is reported below.

A Bandwidth Broker BB [17] manages the QoS resources within a given domain based on the Service Level Specifications (SLS), which have been agreed in that domain. The SLS is a translation of a Service Level Agreement (SLA) into the appropriate information necessary for provisioning and allocating QoS resources within the network devices, and in particular at the edges of the domain on links connecting the domain to adjacent domains. The BB is also responsible for managing inter-domain communication, with BBs in neighboring networks, for the purpose of coordinating SLSs across the domain boundaries.

The BB collects and monitors the state of QoS resources within its domain and on the edges of the domain going to and from adjacent domains. That information, together with the policy information (from the policy rules data base) is used for admission control decisions on QoS service requests to the network. The policy manager, if it exists, verifies such requests against installed policy rules, checking for conflicts with existing requests, taking appropriate preemption actions, etc. The network state information from the BB is also used to verify that resources are currently available in the network to support the request. This information may be obtained directly from the BB via an interface to the policy manager or, more likely, the BB places it in a common database shared by the policy manager.

Across boundaries, the SLS may be on an aggregate basis, where aggregation is on all flows within the domain of a particular QoS service type (i.e. DiffServ codepoint). Within a domain, individual flows may allocate resources from the SLS by issuing Resource Allocation Requests (RARs). It is the responsibility of the BB to coordinate allocation and provisioning of the aggregate resources of the SLSs, into and out of its domain, with those resources requested from RARs.

Figure 2.9 shows a sample network configuration. It consists of three domains AS1, AS2, and AS3 with bandwidth brokers BB1, BB2 and BB3, respectively. Also shown are the SLSs in place between AS1 and AS2, and between AS2 and AS3 as well as individual service users attached to AS1 sending RARs to BB1.

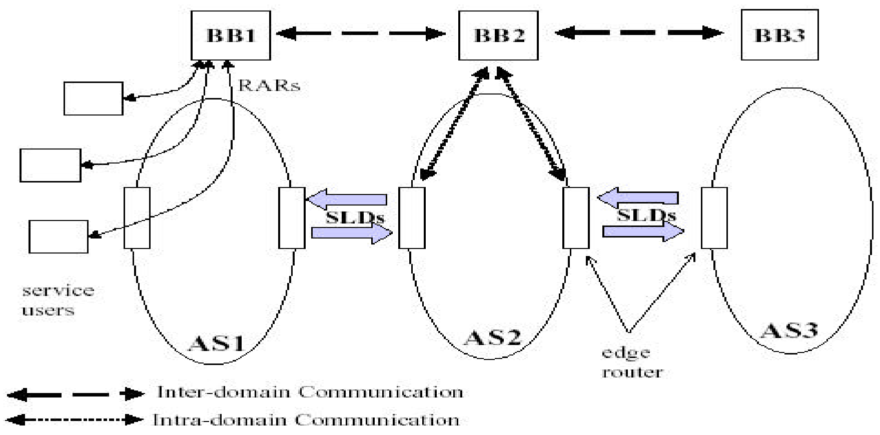


Figure 2.9 Bandwidth broker model [18]

2.6.1 Service mapping

Independently of the DiffServ resource management, the service mapping of IntServ defined services to DiffServ defined services is fundamental for IntServ over DiffServ operations. The proposed implementation can be summarized as in table 2.4.

IntServ	Qualifier	DiffServ PHB
Guaranteed Services	-	EF
Controlled Load	L	AF (higher priority)
	H	AF (lower priority)

Table 2.4 Default mapping [41]

Service mapping depends on appropriate selection of PHB, admission control and policy control on the IntServ request based on the available resources and policies in the DiffServ. There is a known default mapping defined as shown in table 2.4. So, the Guaranteed Service is to be mapped to EF PHB, while the Controlled load depending on whether it is latency tolerant or not can be mapped to AF PHB at different priority levels.

The routers that are able to handle both RSVP and DiffServ packets will perform the service mapping above and, those routers can always override it. These routers will usually be situated at the edges of the DiffServ region. If however, there is a marking at the host or routers located outside the DiffServ region, then the new marking values should be communicated to the marking device by using a mechanism such as a RSVP DCLASS object [54].

2.7 Real-time Transport protocol

The Real-time Transport Protocol (RTP) [19] provides end-to-end delivery services, such as payload type identification, timestamping and sequence numbering, for data with real-time characteristics, e.g. interactive audio and video. It can be used over unicast or multicast networks. RTP is run “on top" of a transport protocol such as UDP. RTP does not reserve resources on the end-to-end path, but rather tries to adapt to prevailing network conditions.

The RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence. The RTP consists of two parts:

- The real-time transport protocol (RTP) itself, to carry data that has real-time properties, briefly described above.
- The RTP control protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session.

An RTCP message consists of a number of “stackable" packets, each with its own type code and length indication. Their format is fairly similar to data packets; in particular, the type indication is at the same location. RTCP packets are multicast periodically to the same multicast group as data packets. Thus, they also serve as a liveness indicator of session members, even in the absence of transmitting media data. The functionality of RTCP is described briefly below:

QoS monitoring and congestion control: RTCP packets contain the necessary information for quality-of-service monitoring. Since they are multicast, all session members can survey how the other participants are faring. Applications that have recently sent audio or video data generate a sender report. It contains information useful for inter-media synchronization, as well as cumulative counters for packets and bytes sent. These allow receivers to estimate the actual data rate.

Session members issue receiver reports for all video or audio sources. They contain information on the highest sequence number received, the number of packets lost, a measure of the interarrival jitter and timestamps needed to compute an estimate of the round-trip delay between the sender and the receiver issuing the report.

Intermedia synchronization: The RTCP sender reports contain an indication of real time and a corresponding RTP timestamp. These two values allow the synchronization of different media, for example, lip-syncing of audio and video.

Identification: RTP data packets identify their origin only through a randomly generated 32-bit identifier. For conferencing applications, a bit more context is often desirable. RTCP messages contain an SDES (source description) packet, in turn containing a number of pieces of information, usually textual. One such piece of information is the so-called canonical name, a globally unique identifier of the session participant. Other possible SDES items include the user's name, email address, telephone number, application information and alert messages.

Session size estimation and scaling: RTCP packets are sent periodically by each session member. The desire for up-to-date control information has to be balanced against the desire to limit control traffic to a small percentage of data traffic even with sessions consisting of several hundred members. The control traffic load is scaled with the data traffic load so that it makes up a certain percentage of the nominal data rate (5%).

This associated protocol provides a feedback to a source on the quality of the data flow, therefore RTP can be thought as a control mechanism to provide the desired QoS, but at a price of a higher protocol overhead.

3 Mobility Management

Often, there is confusion between wireless and mobile networks. The term wireless literally means without wires and was born to avoid the cabling issue. Different from wireless networks, where terminals are fixed, with mobile networks we mean networks able to offer your users mobility capabilities conserving connectivity.

3.1 Mobility terminology

The mobile networks can be classified according to which type of mobility service they offer your users. We can distinguish between three classes of mobility [20]:

User mobility: refers to the ability of a user to access services from different physical hosts. This usually means the user has an account on these different hosts or that a host does not restrict users from using the host to access services.

Personal mobility: complements user mobility with the ability to track the user's location and provide the user's current location to allow sessions to be initiated by and towards the user by anyone on any other network. Personal mobility is also concerned with enabling associated security, billing and service subscription authorization made between administrative domains.

Host mobility: refers to the function of allowing a mobile host to change its point of attachment to the network, without interrupting IP packet delivery to/from that host. There may be different sub- functions depending on what current level of service is being provided; in particular, support for host mobility usually implies active and idle modes of operation, depending on whether the host has any current sessions or not. Access Network procedures are required to keep track of the current point of attachment of all the MNs or establish it at will. Accurate location and routing procedures are required in order to maintain the integrity of the communication. Host mobility is often also called 'terminal mobility'.

It should be useful further, for the following sections, to define two other important terms: Mobile Node (MN) and Correspondent Node (CN).

Mobile Node (MN): An IP node capable of changing its point of attachment to the network. A Mobile Node may have routing functionality.

Correspondent Node (MN): A peer with which a mobile node is communicating. A correspondent node may be either mobile or stationary.

The current mobility protocols can be classified into two main categories:

- Global or Macro Mobility protocols, when a MN is moving from one administrative network domain to another.

- Local or Micro Mobility protocols, when a MN is moving inside one of these administrative network domain.

3.2 Global Mobility

With Global Mobility we mean mobility over a large area [23]. This includes mobility support and associated address registration procedures that are needed when a mobile host moves between IP domains. The Mobile IP protocol [21,22] is the current standard for supporting macroscopic mobility in IP networks i.e. host mobility across IP domains while maintaining transport level connections. It is transparent for applications and transport protocols, which work the same on fixed and mobile hosts. It can be scaled to provide mobility across the Internet and it allows nodes using Mobile IP to interoperate with nodes using the standard IP protocol.

3.2.1 Mobile IPv4

Briefly Mobile IPv4 allows a device to maintain the same IP address (its *home address*) wherever it attaches to the network. However, the mobile device also has a *care-of* address, which connects to the subnet where it is currently located. The care-of address is managed by a *home agent*, which is a device on the home subnet of the mobile device. Any packet addressed to the IP address of the mobile device is intercepted by the home agent and then forwarded on to the care-of address through a tunnel. Once it arrives at the end of the tunnel, the datagram is delivered to the mobile device through the *foreign agent*.

Mobile IP operations

Mobility agents (home agents and foreign agents) advertise their presence in the network by means of *agent advertisement* messages, which are ICMP router advertisement messages with extensions. A mobile node may also explicitly request one of these messages with an agent solicitation message. When a mobile node connects to the network and receives one of these messages, it is able to determine whether it is on its home network or a foreign network. If the mobile node detects that it is on its home network, it will operate normally, without the use of mobility services. In addition, if it has just returned to the home network, having previously been working elsewhere, it will deregister itself with the home agent. This is done through the exchange of a registration request and registration reply.

If, however, the mobile node detects from an agent advertisement that it has moved to a foreign network then it obtains a care-of address for the foreign network. This address may be obtained from the foreign agent (a foreign agent care-of address, which is the address of the foreign agent itself), or it may be obtained by other mechanisms, such as DHCP (in which case, it is known as a co-located care-of address). The use of co-located care-of addresses has the advantage that the mobile node does not need a foreign agent to be present at every network that it visits, but it does require that the DHCP server make a pool of IP addresses available for visiting mobile nodes.

Note that communication between a mobile node and a foreign agent takes place at the link layer level. It cannot use the normal IP routing mechanism, because the mobile node's IP address does not belong to the subnet in which it is currently located.

Once the mobile node has received its care-of address, it needs to register itself with its home agent. This may be done through the foreign agent, which forwards the request to the home agent, or directly with the home agent. Once the home agent has registered the care-of address for the mobile node in its new position, any datagrams intended for the home address of the mobile node are intercepted by the home agent and tunneled to the care-of address. The tunnel endpoint may be at a foreign agent (if the mobile node has a foreign agent care-of address), or at the mobile node itself (if it has a co-located care-of address). Here the original datagram is removed from the tunnel and delivered to the mobile node. The mobile node will generally respond to the received datagram using standard IP routing mechanism. Mobile IP operations are shown in figure 3.1

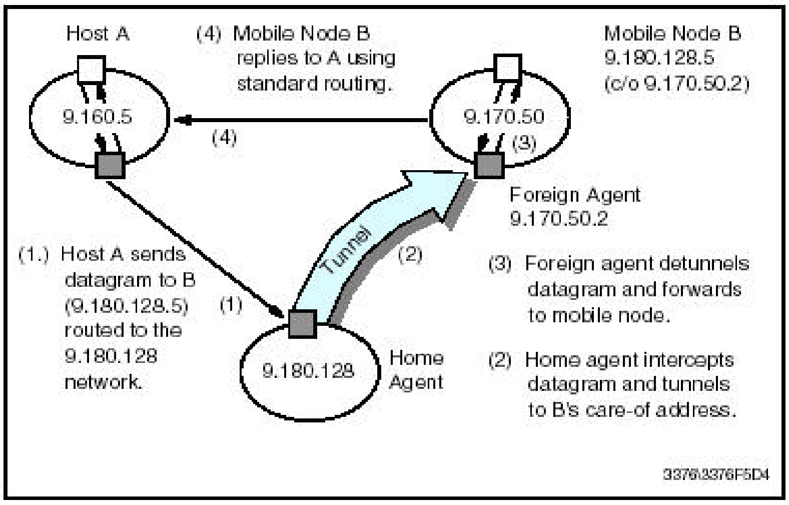

Host A
(4) Mobile Node B replies to A using standard routing.
Mobile Node B 9.180.128.5 (c/o 9.170.50.2)
9.160.5
(4)
9.170.50
(3)
Foreign Agent 9.170.50.2
Tunnel
(2)
(1.) Host A sends datagram to B (9.180.128.5) routed to the 9.180.128 network.
(1)
(3) Foreign agent detunnels datagram and forwards to mobile node.
9.180.128
Home Agent
(2) Home agent intercepts datagram and tunnels to B's care-of address.
3376\3376F5D4

Figure 3.1Mobile IP operations [39].

3.2.2 Mobile IPv6

The first version of Mobile IP operates in Ipv4 networks and has several issues, such as triangular routing. Mobile IP support in IPv6 [22] is based on the experiences gained from the development of Mobile IP support in IPv4, and the opportunities provided by the new features of the IP such as an increased number of available IP addresses, additional automatic IP-configuration features and route optimisation. The protocol also enables IPv6 nodes to cache the binding of a mobile node's home address with its care-of address, and then to send any packets destined for the mobile node directly to it at this care-of address.

Many differences exist between Mobile Ipv4 and Mobile Ipv6 and the most interesting is the integration of the Ipv4 Route optimisation option that allows direct routing from any correspondent node to any mobile node, without needing to pass through the mobile node's home network and be forwarded by its home agent, and thus eliminates the problem of "triangle routing" present in the base Mobile IPv4 protocol.

To provide those optimisations Mobile Ipv6 requires the exchange of additional messages, defined as Ipv6 Destination Options:

Binding Update

The Binding update option is used by a mobile node to inform its home agent or any other correspondent node about its current care-of address.

Binding Acknowledgement

The binding Acknowledgement option is used to acknowledge the receipt of a Binding message if an acknowledgement was requested.

Binding Request

Each node to request a mobile node to send a Binding Update with the current care-of address uses the binding Request option.

Home Address

The Home Address option is used in packets sent by the mobile node to inform the receiver of this packet about the mobile node’s home address.

Finally the most important optimizations introduced by Mobile IPv6 are:

Route optimisation

To avoid triangle routing a mobile node can send Binding Updates to any (mobile or stationary) correspondent node. This allowsIPv6 CN to cache the current care-of address and send the packet directly to a mobile node.

Any IPv6 node sending a packet, first checks its Binding Cache for this destination address. If there is an entry, it will send the packet to the mobile node using a routing header. The route specified by this routing header has two hops. The first hop is the care-of address and the second is the home address.

Binding Cache

Every IPv6 node has a binding cache, which is used to hold the binding for the other nodes. A mobile node, which has configured a new care-of address as primary care-of address, has to register this new address at this home agent and the CN. For this purpose the mobile node sends a Binding Update containing its new binding. The mobile node can enforce the receiver to acknowledge the receipt of the Binding Update by responding with a Binding Acknowledgement. Now, if the sending correspondent node has a Binding entry for that mobile node, it addresses the packets directly to the mobile node's care-of address.

Movement Detection

When a mobile node is away from your home link you can use any combination of mechanism to detect the new link: to wait for a periodically Router Advertisement for example like in Ipv4, but in this case as soon as the mobile node detects that it has moved to another link, it sends a Binding Update to its home agent and to the correspondent node.

Other additional advantages are listed in [22]: Use of Ipv6 anycast addresses, use of stateless address auto configuration and an IPSec option for all security requirements.

3.3 Local Mobility

In recent times, a number of micro-mobility protocols have been discussed in the IETF Mobile IP Working Group, that address some of these performance and scalability issues. Micro-mobility protocols are designed for environments where mobile hosts change their point of attachment to the network so frequently that the basic Mobile IP protocol tunneling mechanism introduces network overhead in terms of increased delay, packet loss and signaling. For example, many real-time wireless applications (e.g., voice-over-IP) would experience noticeable degradation of service with frequent handoff. The establishment of new tunnels can introduce additional delays in the handoff process causing packet loss and delayed delivery of data to applications. This delay is inherent in the round-trip incurred by Mobile IP as the registration request is sent to the home agent and the response sent back to the foreign agent. Micro-mobility protocols aim to handle local movement (e.g., within a domain) of mobile hosts without interaction with the Mobile IP-enabled Internet. This has the benefit of reducing delay and packet loss during handoff and eliminating registration between mobile hosts and possibly distant home agents when mobile hosts remain inside their local coverage areas. Eliminating registration in this manner reduces the signaling load experienced by the core network in support of mobility.

Micro-mobility protocols aim to support fast handoff control with minimum or zero packet loss, and to minimize signaling through the introduction of paging techniques thereby reducing registration to a minimum. These enhancements are necessary for the Internet to scale to support very large volumes of wireless subscribers.

3.3.1 Protocols overview

The following section describes the basic functionalities of the most important protocols for managing mobility within an administrative network domain.

HMIPv6

The Hierarchical Mobile IPv6 is an extension to the plain IPv6, which aims to reduce control signaling between the mobile node and the correspondent node.

The Hierarchical MIPv6 scheme introduces a new element, the Mobility Anchor Point (MAP), and slight extensions to the MN and the Home Agent operations. The CN operation will not be affected. The introduction of the MAP concept minimizes the latency due to handoffs between access routers since it will take less time to bind-update a local MAP than a distant HA. Just like MIPv6, this solution is independent of the underlying access technology, allowing Fast Handoffs within, or between, different types of access networks. Furthermore, a big architectural migration from Hierarchical MIPv4 networks is not necessary, since a dual operation of IPv4 and IPv6 Hierarchies will be possible making use of the similarity in architecture. The introduction of the MAP concept will further diminish signaling generated by MIPv6 over a radio interface. This is due to the fact that a MN only needs to perform one local BU (Binding Update) to a MAP when changing its layer 3 access point within the MAP domain.

Describing in brief, when a MN change its point of attachment inside a local MAP domain, it needs to register this change only on the local MAP, hence its global CoA (RCoA) does not change and it does not need to communicate with the HA or the CN.

The first phase after a MN arrives in a foreign network is called discovery phase and is performed via a router advertisement sent by the access router. By this message a MN will discover the address of the local MAP and the distance between the MN and the MAP. Clearly if the MN receives a different MAP’s address, it must communicate this change by a BU to the HA and the CN.

In order to initiate the route optimization the MNs need to know the original source of the message. In case the MAP is used in extent mode the address will be tunneled by the MAP, therefore a check needs to be performed on the internal packet's routing header to find out whether the packet was tunnelled by the HA or originated from a CN using route optimization instead.

In order to use in the most efficient manner the bandwidth resources, the MN can register with more than one MAP simultaneously. This is a good solution when the CN resides in the same link, because it avoids sending all packets via the "highest" MAP in the hierarchy. Two valid alternatives of this approach are Regional Registration and Fast Handover protocols

Regional Registration

Like the Mobile IP, when a mobile node first arrives at a visited domain, it performs a registration with its home network. At this registration, the home agent registers the care-of address of the mobile node, nevertheless if the home agent resides far from the actual position of the MN, this registration may be quite long. Instead, when the visited domain supports regional registrations, the care-of address that is registered at the home agent is the address of a Gateway Foreign Agent (GFA), a Foreign Agent which has a publicly routable IP address. The GFA keeps a visitor list of all the mobile nodes currently registered with it.

Since the care-of address registered at the home agent is the GFA address, it will not change when the mobile node changes foreign agents under the same GFA. Thus, the home agent does not need to be informed of any further mobile node movements within the visited domain.

Figure 3.x illustrates the signaling message flow for registration with the home network. After the registration at the home agent, the home agent records the GFA address as the care-of address of the mobile node. If the GFA address was dynamically assigned to the mobile node, the Registration Reply has an extension indicating the IP address of the GFA to the mobile node.

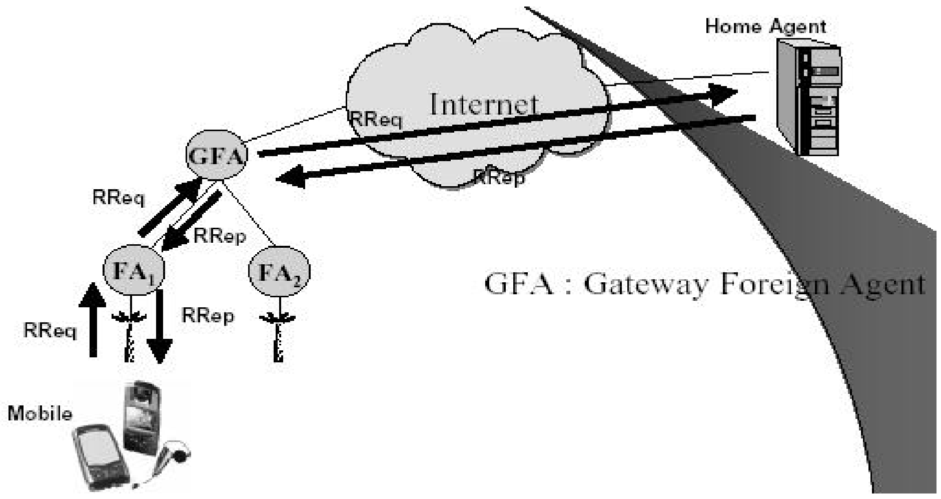


Figure 3.2 Regional Registration messages

Figure 3.2 illustrates the signaling message flow for regional registration. Even though the mobile node's local care-of address changes, the home agent continues to record the GFA address as the care-of address of the mobile node. Two new message types for regional registrations were introduced: Regional Registration Request and Regional Registration Reply. The Regional Registration Request is used by a mobile node to register with its current GFA.

The Regional Registration Reply message, delivers the indication of regional registration acceptance or denial to a mobile node. The definition of this message format is analogous with the Registration Request and Registration Replay [25], except for the fields: Type, GFA IP Address, and CoA.

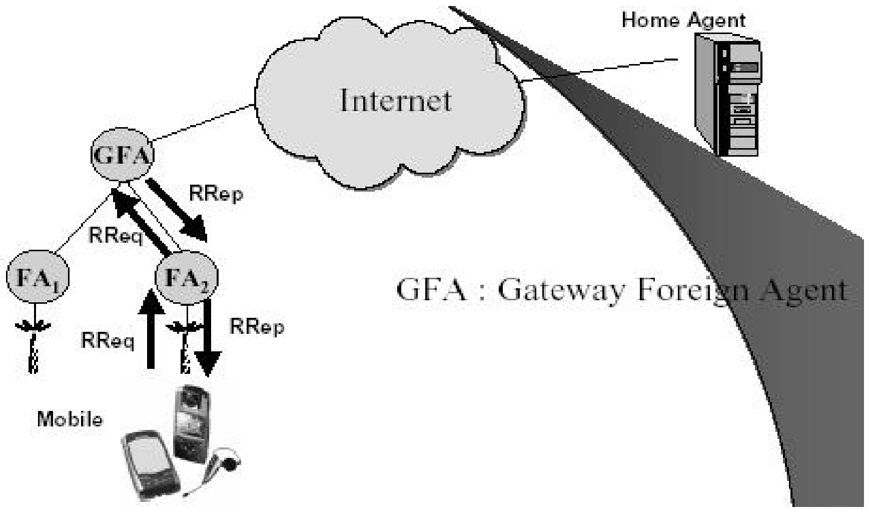


Figure3.3 Local Regional Registration messages

Fast Handovers in Mobile IPv6

The mechanism was designed to obtain fast L3 handoffs for Mobile IPv6 [24]. This mechanism involves the use of L2 triggers, which allow the L3 handoff to be anticipated rather than being performed after the L2 handoff completion as normal. Fast Handoffs are required to ensure that the layer 3 (Mobile IP) handoff delay is minimized, thus also minimizing and possibly eliminating the period of service disruption, which normally occurs when a MN moves between two ARs. This period of service disruption usually occurs due to the time required by the MN to update its HA after it moves between ARs.

During this time period the MN cannot resume or continue communications. While the MN is connected to its old Access Router (oAR) and is about to move to a new Access Router (nAR), the Fast Handoffs in Mobile IPv6 requires:

- the MN to obtain a new care-of address at the nAR while connected to the oAR, the MN to send a BU to its old anchor point (e.g. oAR) to update its binding cache with the MN's new care-of address.
- the old anchor point (e.g. oAR) to start forwarding packets destined for the MN to nAR.

This mechanism allows the anticipation of the layer 3 handoff, such that data traffic can be redirected to the MN's new location before it moves there. However it is not easy to determine the correct time to start forwarding between oAR and nAR, which has an impact on how smooth the handoff will be. Packet loss will occur if this is performed too late or too early with respect to the time in which the MN detaches from oAR and attaches to nAR. Also, some measure is needed to support the case in which the MN moves quickly back-and-forth between ARs (ping-pong). Simultaneous bindings provide a solution to these issues.

Fast handover is implemented by adding a number of new messages, which we do not mention in this context, implemented between access routers and between an access router and a mobile node. This mechanism facilitates the mobile node to configure a new care-of-address before it moves to a new access router in a way that can use this new care-of-address immediately on connection with a new access router. The key elements of this protocol are new care-of-address configuration, Duplicate Addressing avoidance and Neighbor Discovery.

HAWAII

The HAWAII [28] is a domain-based approach for supporting mobility. HAWAII uses specialized path setup schemes, which install host-based forwarding entries in specific routers to support intra-domain micro-mobility and defaults to using Mobile-IP for inter-domain macro-mobility. These path setup schemes deliver excellent performance by reducing mobility-related disruption to user applications, and by operating locally, reduce the number of mobility-related updates. Also, in HAWAII, mobile hosts retain their network address while moving within the domain, simplifying Quality of Service support. Furthermore, reliability is achieved through maintaining soft-state forwarding entries for the mobile hosts and leveraging fault detection mechanisms built in existing intra-domain routing protocols.

As well as other micro mobility mechanisms, HAWAII uses a Hierarchical approach to provide transparent mobility to correspondent hosts. Each host has an IP address and a home domain, within which the mobile host retains its IP address. When the mobile host moves into a foreign domain, the traditional Mobile IP mechanisms provide mobility support.

The protocol contains three types of message to accomplish the path set-up: power-up, update and refresh. The path setup power-up message has the effect of establishing a host specific route for the mobile host inside its home domain. HAWAII uses path setup update messages to establish and update host-based routing entries for the mobile host in selective routers in the domain. Finally the mobile host rarely sends periodic path refresh messages to the base station to which it is attached to maintain the host-based entries.

The design of maintaining the mobile host address unchanged within a domain simplifies per flow QoS support. In the case of HAWAII, support for QoS is simplified since a mobile host's address is unchanged as long as the user remains within a domain; further when a routing change was necessary, since the HAWAII routing changes are localized, they result in fast reservation restorations for the mobile user.

Cellular IPv6

Cellular IP (CIP) [29] is a proposal, which aims to combine smooth mobility support of telephony systems with the flexibility and robustness of IP-based networking. Macro mobility is taken care of by Mobile IP. The getaway router on the border of the wireless access network is one end-point for the Mobile IP tunnel, while the home agent in the home network is the other one. In the wireless access networks mobile node are identified by their home address. For global mobility, they utilize the address of the CIP gateway as their Mobile IP care of address. For the use with IPv6 Cellular IP is slight modified [30].

Figure 3.4 below presents a schematic view of the architecture of Cellular IPv6.

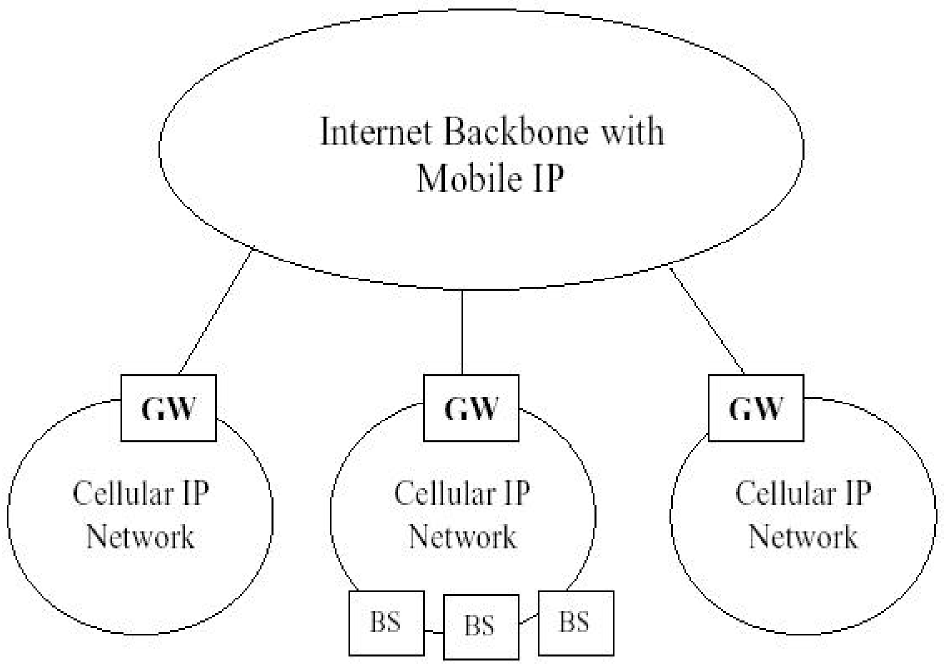


Figure 3.4 Cellular IP Architecture

In this architecture, the Base Stations periodically transmit beacon signals. Mobile hosts use these beacon signals to locate the nearest Base Station. A mobile host can transmit a packet to the nearest Base Station.

All IP packets transmitted by a mobile host are routed from the Base Station to the Gateway by hop-by-hop shortest path routing, regardless of the destination address.

Cellular IP nodes maintain Route Cache. Packets transmitted by the mobile host create and update entries in each node's Cache. An entry maps the mobile host's IP address to the neighbor from which the packet arrived to the node.

The chain of cached mappings referring to a single mobile host constitutes a reverse path for downlink packets addressed to the same mobile host. As the mobile host migrates, the chain of mappings always points to its current location because its uplink packets create new and change old mappings.

IP packets, addressed to a mobile host, are routed by the chain of cached mappings associated with the mobile host. In order to prevent its mappings to fall in timing out, a mobile host can periodically transmit control packets. Control packets are ICMP packets with specific authentication payloads. Mobile hosts that are not actively transmitting or receiving data but want to be reachable for incoming packets, let their Route Cache mappings time out but maintain Paging Cache mappings. Paging Caches will rout IP packets addressed to these mobile hosts. Paging Caches have a longer timeout value than Route Caches and are not necessarily maintained in every node.

Cellular IP is optimized to support local mobility but efficiently interworks with Mobile IP to provide wide area mobility support. Cellular IP shows great benefits in comparison with existing host mobility proposals for environments where mobile hosts migrate frequently.

3.3.2 Mobility protocols classification

A good classification about the micro mobility protocols already existing can be found in the literature [23]. The two major categories of Regional Mobility protocols are:

- Proxi-Agent Architectures (PAA)
- Localized Enhanced-Routing Schemes (LERS)

Proxy Agents Architecture Schemes (PAA):

These schemes extend the idea of Mobile IP into a hierarchy of Mobility Agents. This way, when the MN changes its CoA, the registration request does not have to travel up to the HA but remains ‘regionalized’. Examples include the initial Hierarchical Mobile IP [24] and its alternatives: Mobile IP Regional Registration [25], Transparent Hierarchical Mobility Agents (THEMA) [26], Fast Handoff in Mobile IPv4 [27]. Mobile IP version 6 [22] has also had some optional extensions supporting a hierarchical model.

Localized Enhanced-Routing Schemes (LERS):

These schemes introduce a new, dynamic Layer 3 routing protocol in a ‘localized’ area. There are several distinctive approaches:

- *Per host Forwarding Schemes:* Inside a domain, a specialized path set-up protocol is used to install soft-state host-specific forwarding entries for each MN. The domain is connected to the Internet via a special default gateway. Examples include Handoff-Aware Wireless Access Internet Infrastructure (HAWAII) [28] and recently, Cellular IPv6 [30], which includes IPv6 enhancements to the original Cellular IP.

- *Multicast-based Schemes:* A multicast-CoA is assigned to a single MN, which can then be used to instruct neighboring multicast-enabled routers to join the MN's virtual multicast group, either prior to or during handovers. This can be visualized as a multicast cloud centered on the MN's current location but also covering future locations. Examples include Dense mode multicast-based [31] and the recent Sparse-mode multicast-based [32].

- *MANET-based Schemes:* MANET protocols were originally designed for mobile ad-hoc networks, where both hosts and routers are mobile. The routing is multi-hop and adapts as the MNs move and connectivity in the network changes. Currently there is only one proposal in this category: MER-TORA [33].

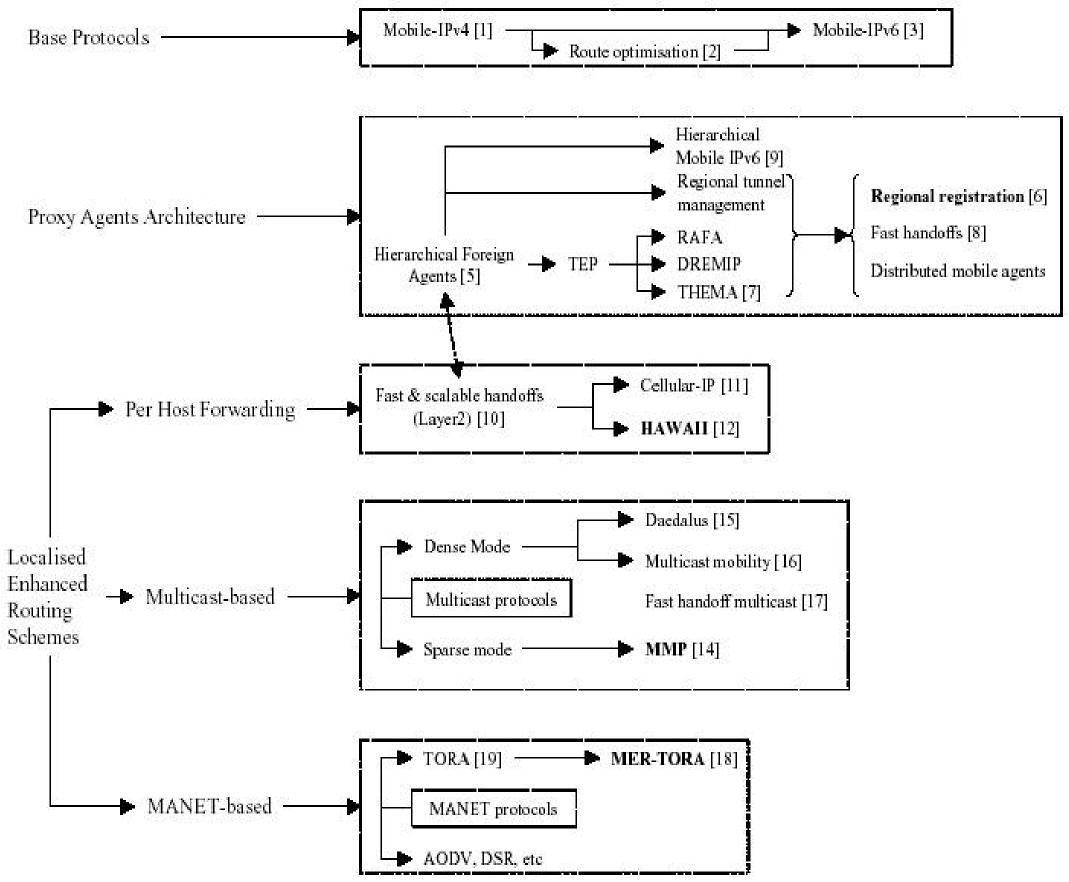


Figure 3.5 General picture on Mobility mechanisms [23]

4 QoS Management in Mobile environments

In this chapter we introduce the most significant protocol architectures already existing and the new proposals that manage QoS in a mobile environment.

4.1 General Packet Radio Service

Until recently, the main application of most mobile systems, e.g., Global System for Mobile communications (GSM), has been mobile telephony. Recently, the use of mobile data applications such as the GSM Short Message Service (SMS) has gained popularity. However, the GSM system can only support data services up to 9.6 kbit/s, circuit switched. The General Packet Radio Service (GPRS) [38] developed by the European Telecommunications Standards Institute (ETSI), provides packet switched services in a GSM network that can allow bitrates, theoretically up to 170 kbit/s per user. However, commercial GPRS systems will support a maximum bit rate of 115 kbit/s. Furthermore, GPRS is able to provide additional features such as: more efficient use of scarce radio resources, faster set-up / access times , connectivity to other external packet data networks by using IPv4 or X.25, and user differentiation based on Quality of service (QoS) agreements.

In addition, GPRS packet transmission offers a more user-friendly billing than that offered by circuit switched services. In circuit switched services, billing is based on the duration of the connection. This is unsuitable for applications with bursty traffic. The user must pay for the entire airtime, even for idle periods when no packets are sent (e.g., when the user reads a Web page). In contrast to this, with packet switched services, billing can be based on the amount of transmitted data. The advantage for the user is that he or she can be "online" over a long period of time but will be billed based on the transmitted data volume.

4.3.1 GPRS System Architecture

To integrate GPRS into the existing GSM architecture, a new class of network nodes, called GPRS support nodes (GSN), has been introduced [37].

GSNs are responsible for the delivery and routing of data packets between the mobile stations and the external packet data networks (PDN). Fig. 4.1 illustrates the system architecture.

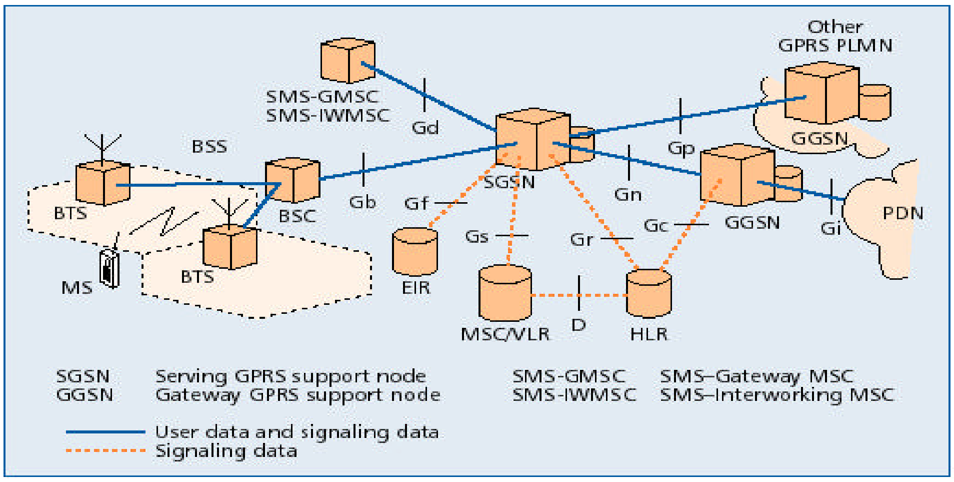


Figure 4.1 GPRS system architecture

A serving GPRS support node (SGSN) is responsible for the delivery of data packets from and to the mobile stations within its service area. Its tasks include packet routing and transfer, mobility management (attach/detach and location management), logical link management, and authentication and charging functions. The location register of the SGSN stores location information (e.g., current cell, current VLR) and user profiles (e.g., IMSI, address(es) used in the packet data network) of all GPRS users registered with this SGSN.

A gateway GPRS support node (GGSN) acts as an interface between the GPRS backbone network and the external packet data networks. It converts the GPRS packets coming from the SGSN into the appropriate packet data protocol (PDP) format (e.g., IP or X.25) and sends them out on the corresponding packet data network. In the other direction, PDP addresses of incoming data packets are converted to the GSM address of the destination user. The readdressed packets are sent to the responsible SGSN. For this purpose, the GGSN stores the current SGSN address of the user and his or her profile in its location register. The GGSN also performs authentication and charging functions.

Fig. 4.1 further shows the interfaces between the new network nodes and the GSM network as defined by ETSI. The Gb interface connects the BSC with the SGSN. Via the Gn and the Gp interfaces, user data and signaling data are transmitted between the GSNs. The Gn interface will be used if SGSN and GGSN are located in the same PLMN, whereas the Gp interface will be used if they are in different PLMNs.

Across the Gf interface, the SGSN may query the IMEI of a mobile station trying to register with the network. The Gi interface connects the PLMN with external public or private PDNs, such as the Internet or corporate intranets. Interfaces to IP (IPv4 and IPv6) and X.25 networks are supported.

The HLR stores the user profile, the current SGSN address, and the PDP address(es) for each GPRS user in the PLMN. The Gr interface is used to exchange this information between HLR and SGSN. For example, the SGSN informs the HLR about the current location of the MS. When the MS registers with a new SGSN, the HLR will send the user profile to the new SGSN. The signaling path between GGSN and HLR (Gc interface) may be used by the GGSN to query a user's location and profile in order to update its location register.

In addition, the MSC/VLR may be extended with functions and register entries that allow efficient coordination between packet switched (GPRS) and circuit switched (conventional GSM) services. Examples of this are combined GPRS and non-GPRS location updates and combined attachment procedures. Moreover, paging requests of circuit switched GSM calls can be performed via the SGSN. For this purpose, the Gs interface connects the databases of SGSN and MSC/VLR. To exchange messages of the short message service (SMS) via GPRS, the Gd interface is defined. It interconnects the SMS gateway MSC (SMS-GMSC) with the SGSN.

In general, there is a many-to-many relationship between the SGSNs and the GGSNs: a GGSN is the interface to external packet data networks for several SGSNs; an SGSN may route its packets over different GGSNs to reach different packet data networks.

All GSNs are connected via an IP-based GPRS backbone network. Within this backbone, the GSNs encapsulate the PDN packets and transmit (tunnel) them using the GPRS Tunneling Protocol GTP. There are two kinds of GPRS backbones:

- Intra-PLMN backbone networks connect GSNs of the same PLMN and are therefore private IP-based networks of the GPRS network provider.
- Inter-PLMN backbone networks connect GSNs of different PLMNs. A roaming agreement between two GPRS network providers is necessary to install such a backbone.

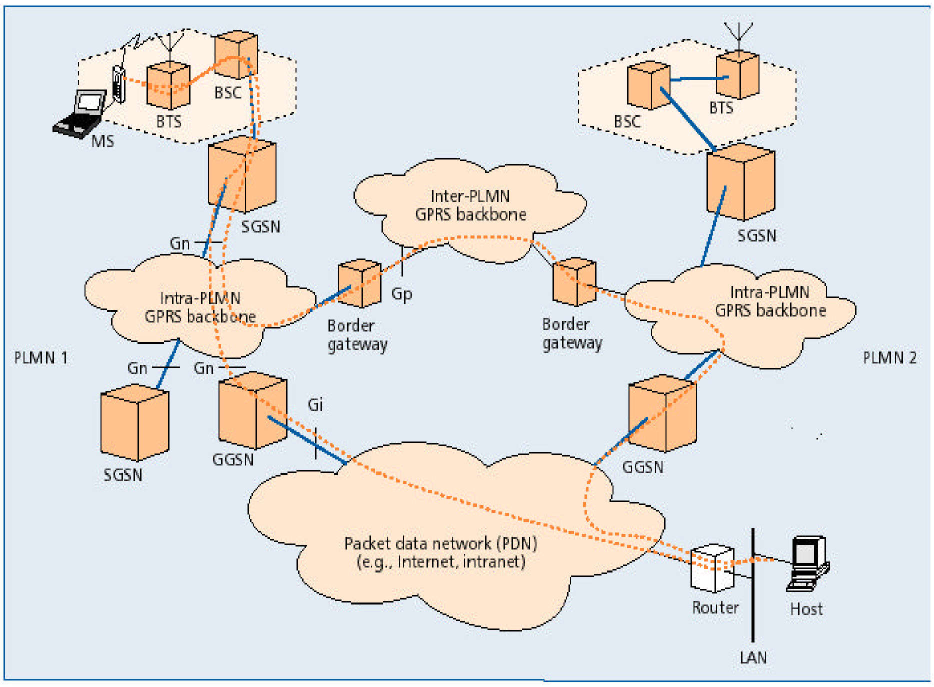


Figure 4.2 GPRS intra-inter connections

Fig. 4.2 shows two intra-PLMN backbone networks of different PLMNs connected with an inter-PLMN backbone. The gateways between the PLMNs and the external inter-PLMN backbone are called border gateways. Among other things, they perform security functions to protect the private intra-PLMN backbones against unauthorized users and attacks. The Gn and Gp interfaces are also defined between two SGSNs. This allows the SGSNs to exchange user profiles when a mobile station moves from one SGSN area to another.

4.3.2 GPRS QoS

The Quality of Service requirements of typical mobile packet data applications are very diverse, because they range from real-time multimedia, Web browsing to e-mail transfer. The possibility to support and specify different QoS classes, for each individual session, is therefore an important feature. GPRS allows defining QoS profiles using the parameters service precedence, reliability, delay, and throughput [38].

The *service precedence* is the priority of a service in relation to another service. There exist three levels of priority: high, normal, and low.

The *reliability* indicates the transmission characteristics required by an application. Three reliability classes are defined, which guarantee certain maximum values for the probability of loss, duplication, mis-sequencing, and corruption (an undetected error) of packets.

The *delay* parameters define maximum values for the mean delay and the 95-percentile delay. The latter is the maximum delay guaranteed in 95 percent of all transfers. The delay is defined as the end-to-end transfer time between two communicating mobile stations or between a mobile station and the Gi interface to an external packet data network. This includes all delays within the GPRS network, e.g., the delay for request and assignment of radio resources and the transit delay in the GPRS backbone network. Transfer delays outside the GPRS network, e.g., in external transit networks, are not taken into account.

The *throughput* specifies the maximum/peak bit rate and the mean bit rate.

The GPRS QoS profile can be requested either by the mobile user, during a phase known, or, if no profile is requested, a default one is assigned to the user upon subscription to the network. The SGSN is the entity responsible for fulfilling the QoS profile requested of the MN. When a MN wishes to modify its QoS profile, it must detach from the network and then attach again specifying a new QoS profile. On the other hand, the SGSN can modify the QoS profiles according to network load and resources available.

Inside the GPRS core network, all flows are treated uniformly (best effort IP), thus the QoS parameters, described above, are used to allocate resources outside the core network: further more the MN is responsible for shaping its traffic to the negotiated QoS. This implies that in a realistic topology, where the ingress and the egress nodes are not directly linked, degradations in the QoS satisfaction are possible.

Communication between the SGSN and GGSN is based on the IP tunnel. When a standard IP packet reaches a GSN it is encapsulated in a new IP packet and routed accordingly. The GPRS Tunnel Protocol (GTP) is responsible for performing this encapsulation. This restriction implies a major difficulty in applying IP QoS frameworks, such as IntServ/RSVP and DiffServ within a GPRS network. Another problem arising from this tunnel-based approach is that different data flows addressed to the same terminal are treated in the same manner. There is one-to-one mapping between the MN IP address and tunnel identifiers (TIDs) and this represent a problem particularly with the IntServ framework, where micro-flows are a basic concept.

In the literature we can find several possible implementations for utilising QoS-enabling schemes to enhance the GPRS architecture, based on IntServ or DiffServ models [38]. Nevertheless both solutions seem to suffer from a scalability issue. The IntServ model capitalizes on a soft-state, but introduces signal overhead, the DiffServ model involves the use of more than one IP address per MN.

4.2 Universal Mobile Telecommunication System

After the tremendous growth in the past few years, the wireless market is heading into a very challenging time. As market penetration is slowing down, operators are shifting their focus from acquisition of new subscribers to retaining existing ones. 3G wireless technology promises extensive revenue opportunities onto the service delivery, but these opportunities come with challenges such as high license costs or expensive investments in infrastructure.

UMTS is the first wireless technology in which serious emphasis on end-to-end QoS has been given. For 3G wireless operators or service providers, the ability to provide consistent policy-based QoS across the network offers business advantage over the current wireless Internet service providers, improving both revenue and profit.

4.2.1 UMTS System Architecture

The Universal Mobile Telecommunications System (UMTS) [39] as specified by the third generation partnership project (3GPP) was formally adopted by the ITU as a member of its family of IMT-2000 third generation mobile communication standard in November 1999.

The UMTS System Architecture consists of two parts:

- UTRAN System Architecture: RNC, Node B
- UMTS Core Network: U-MSC

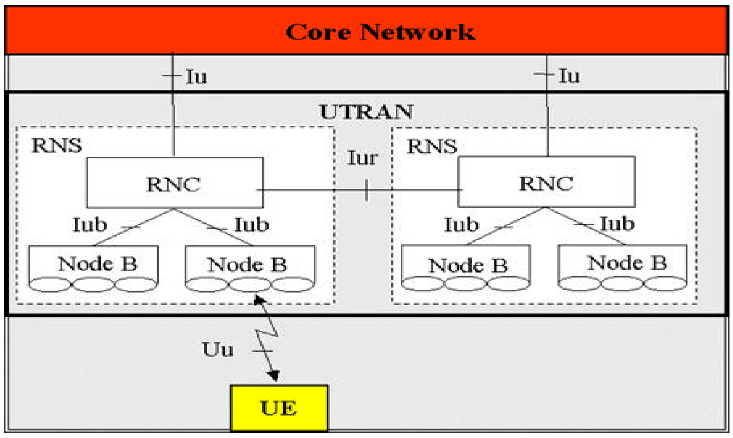


Figure 4.3 UMTS Architecture

UTRA has been defined as an access network (the UMTS Terrestrial Radio Access Network, or UTRAN) as shown above. This means that the radio interface independent functions, essentially call control and mobility management, are outside the scope of the UTRAN specifications and handled by the core network. A UTRAN consists of one or more Radio Network Subsystems (RNSs), which in turn consist of base stations (termed Node Bs) and Radio Network Controllers (RNCs). A Node B may serve one or multiple cells. Mobile Stations are termed User Equipments (UEs) and in practice are likely to be multi-mode to enable handover between the FDD and TDD modes and, prior to complete UMTS geographical coverage, GSM as well.

The UTRAN permits, under certain circumstances, the use of multiple radio links across multiple cells in support of a single UTRAN-UE connection (termed soft handover). These links may exist across different Node Bs in neighbouring RNCs, in which case the necessary signaling and data transfers occur across the Iur interface. The Iur also participates in mobility switching (hard handover), i.e. where switching between Iu instances occurs.

The Core Network is divided into circuit switched and packet switched domains. Circuit switched elements are Mobile services Switching Centre (MSC), Visitor location register (VLR), and Gateway MSC and packet switched elements are Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN). Both domains share some network elements, like EIR and AUC.

Asynchronous Transfer Mode (ATM) is defined for UMTS core transmission. ATM Adaptation Layer type 2 (AAL2) handles circuit switched connection and packet connection protocol AAL5 is designed for data delivery. The architecture of the Core Network may change when new services and features are introduced. The Number Portability DataBase (NPDB) will be used to enable the user to change the network while keeping their old phone number. The gateway Location Register (GLR) may be used to optimise the subscriber handling between network boundaries. MSC, VLR and SGSN can merge to become a UMTS MSC.

In UMTS, as with all other technologies, a radio network must be created to provide service areas for the mobile user equipment (UE). Any UE within a service area should then be able to connect to the core network, providing it with the various UMTS services.

To create the radio network, and thus the service area, a number of radio sites (Node Bs) must be designed. Each node B transmits to and receives from UEs in its service area, thus providing them with UMTS service. The service area for any particular sector on a node B is referred to as a cell. The total network service area can therefore be thought of as a matrix of contiguous cells. UMTS cells must be designed with an appropriate planning tool and a specialized understanding of the interdependent relationship of UMTS coverage, capacity and quality.

Unlike non-CDMA technologies, the cell sizes in UMTS are dynamic. Each UE in the network is designed to operate at a particular ‘Energy per Bit / Noise Spectral Density’ (E_b/N_o) value. With very few UEs in a particular cell, there would be little contribution to the noise (N_o), meaning low interference. However, with too many UEs in the cell, the N_o value would increase to an unacceptable level, degrading network performance. In such a case, the Node B sector would shrink its cell size so it contains less UEs, resulting in less interference in that cell, thus maintaining the required QoS (Quality of Service).

The node B sectors in UMTS (unlike the sectors in non-CDMA technologies) each use the same frequencies. The signals from each sector are spread across the frequency band using a PN (pseudo noise) offset. Each sector’s PN offset is designed to be different to the PN offsets of other sectors within interference distance, enabling each sector to be differentiated from other sectors within the vicinity. PN offset planning in UMTS can be compared to frequency planning in other technologies, such as GSM.

The delivery of multimedia services to the user in the mobile domain requires that 3G mobile communications systems support data rates that are substantially higher than those provided by the second generation (2G) networks (9.6 kbit/s currently supported on GSM). In UMTS for example, users will be provided with datarates of up to 144 kbit/s in macrocellular environments, up to 384 kbit/s in microcellular environments, and up to 2 Mbit/s in indoor or picocellular environments.

Many of the data rates providing these services can be achieved with FDD (Frequency Division Duplex) equipment. However, in low mobility environments (e.g., offices) where the higher data rates are required, TDD (Time Division Duplex) equipment can provide the same level service. TDD, as well as supporting the highest data rates, provides greater spectrum efficiency and (as there is no cell breathing effect) reduces the complexity of the radio network planning.

A further key requirement for UMTS design is the need for an evolution of the core network architecture used in GSM to allow current GSM operators to protect their infrastructure investments during the upgrade of their networks to support UMTS.

4.2.1 QoS in UMTS

The combination of UMTS and IP-based enabled QoS networks seems the right step towards the provision of a quality assured mobile Internet. In the UMTS QoS provisioning, we have to distinguish between a mobile terminal (TE) communicating with other terminals in the UMTS domain and a TE communicating with terminals in the external IP domain. In the former case only the QoS provisioning mechanisms of UMTS are responsible for the quality perceived by the terminals; in the latter case the quality perceived by TE depends on the QoS mechanisms of the UMTS and IP domains, as well as the inter-working agreements between the UMTS and IP domains.

In the end-to-end QoS UMTS architecture we can recognize three different types of bearer services: TE/TM local bearer service, UMTS bearer service and the external bearer service.

The TE/TM local bearer service provides a mapping between QoS application requirements and UMTS services. The mobile termination (TM), in this context, represents the access point of a terminal in the UMTS domain.

The UMTS bearer service mainly provides the UMTS QoS, it comprises the Radio Access bearer service and the core network bearer service. The external bearer service may use network services from other domains. The UMTS bearer service functions are handled by both the user and the control plane. The user plane is responsible for maintenance of the data traffic specified by the QoS attributes; the control plane functions manage the establishment, modification and maintenance of the services.

In order to permit the Internet applications to operates efficiently over the UMTS services, a mapping mechanism must be provided between the UMTS domain and the IP one, independently of IP-specific QoS architecture. To operate this mapping, we must assume that there is service level specification (SLS) between the two domains, and according to the existing SLSs, the Translator function will perform the mapping between the UMTS QoS classes and the IP QoS attributes.

In case the external IP network supports the differentiated services, the Translator function will map the UMTS QoS classes to the appropriate DiffServ codepoints; if the external IP network supports the Integrated services, the Translator will map the UMTS QoS classes to the appropriate IntServ traffic specification (Tspec). As we know, the IntServ relies on the RSVP to setup and maintain the resources reservation. The signaling-based RSVP nature matches the functions in the UMTS control plane. The main functionality of the RSVP daemon is the interception of RSVP messages and their transformation into the format of the reservation request used in the UMTS control plane and vice versa.

Although the general idea behind the UMTS and DiffServ interworking is similar to the UMTS and IntServ case, there is a significant difference. The DiffServ capable network does not use any resources reservation protocol, but the control information is contained inside the IP packet header. Thus the main responsibility of the DiffServ Manager is to mark the IP packets according to the directions of the external bearer service manager.

Four generic QoS classes are proposed for UMTS [39]. These are the conversional, streaming, interactive and background classes. The main distinction among them is based on the delay parameters. The conversional is the most delay sensitive, while the background one is the most delay insensitive.

The conversional class targets applications with strict QoS demand, like voice over IP and video conferencing tools. The streaming class is intended for one-way streaming applications, like video-on-demand. The interactive class will be used by applications that involve human interaction, like Web browsing, and database retrieval. Lastly, the background class targets applications without timing requirements, for example download of email.

The most important parameters that discriminate the classes of service in the UMTS bearer service aforementioned are: the maximum bit rate, the delivery order, the transfer delay and the guaranteed bit rate. The exact mapping of these parameters between the UMTS domain and the IP domain is hard work for the traffic-engineering administrators.

4.3 Existing propriety architectures

This section identifies the most significant schemes for providing parts of an all-inclusive support of QoS-aware mobility. A full support of mobile terminals with QoS requirements can be accomplished by a combination of these schemes [23].

4.3.1 INSIGNIA

INSIGNIA [34] is an in-band signaling system for supporting quality of service (QoS) in mobile ad hoc networks. The term `in-band signaling´ refers to the fact that control information is carried along with data in IP packets. Opposite the out-of-band approaches (e.g., RSVP) this mechanism is more appropriate when supporting end-to-end quality of service in highly dynamic environments such as mobile ad hoc networks where network topology, node connectivity and end-to-end quality of service are strongly time-varying.

INSIGNIA is designed to support the delivery of adaptive real-time services and includes fast session/flow/microflow reservation, restoration and adaptation algorithms between source/destination pairs. To establish an adaptive real-time flow, INSIGNIA uses a new IP option to setup, restore and adapt resources between source- destination pairs. When intermediate nodes receive packets with these IP options set they attempt to reserve, restore or adapt resources forwarding date packets toward the destination. The protocol commands, encoded using the IP option field, include *service mode, payload type, bandwidth indicator and bandwidth request* fields.

Briefly, when a source wants to establish a fast reservation to a destination node, it sets the reservation (RES) mode bit in the IP option service model field of the data packet to be sent. The intermediate routers execute admission control to accept or deny the request, and consequently the packets are scheduled. When the destination receives a RES packet it sends a QoS report to the source to confirm the reservation establishment.

The service mode indicates the level of the service assurance requested in support of adaptive services, we can distinguish substantially only two types of service that mark the packets as normal best effort packets (BE) or RES packets.

The IP option also includes an indication of the payload type, which identifies whether the packet is a base QoS (BQ) or enhanced QoS (EQ) packet. The bandwidth indicator IP option is used to setup a max-reserved or min-reserved service respectively by the parameters MAX or MIN.

In order to manage this resource reservation mechanism in a dynamic environment, the soft-state approach is adopted in the intermediate routers. The source node is informed of the on-going status of flows by the QoS reporting. Destination nodes monitor continually the QoS status and periodically transmit to the source information about this status and end-to-end adaptation level. INSIGNIA in fact provides a set of adaptation levels that can be selected. The time scale, over which the adaptation action occurs, is dependent on the application considered, and these scaling actions could be instantaneous or low-pass filter operations.

4.3.2 Mobile RSVP

The MRSVP [35] is a reservation protocol to provide real-time services to mobile users in an Integrated Services Packet Network. The currently proposed reservation protocol on the Internet, RSVP, is not adequate to satisfy mobility independent service guarantees, because a mobile cannot make advance reservation to multiple locations using RSVP. Other important limitations in the RSVP are any provision for the passive reservation, and the impossibility to initiated a reservation from a location when the sender or the receiver is not present at that location.

MRSVP requires *proxy agents* to make a reservation along the paths from the location in the MSPEC of the sender to the locations in the MSPEC of the receiver; where the MSPEC is the set of locations the mobile host will visit during the lifetime of the connection. We can distinguish between *local proxy agent* and *remote proxy agent* depending on whether it is situated at the current location of the mobile host or not. Normally the remote proxy agent makes passive reservations on behalf of the mobile host, instead the local proxy agent acts as a normal router for the mobile host making active reservations from the sender to the mobile. An important issue in this context is how the mobile host discovers who will be your proxy agent, but we do not deal with that in more detail here.

After the mobile host knows the IP address of its proxy agent, it has to set up the path of the reservation. We have to distinguish now between whether the mobile host is the sender or the receiver of the flow. In the first case, the path of the active reservation from the current location of the mobile host and the path of the passive reservations from its proxy agents are determined by the routing mechanism of the network. When the mobile host is the receiver of the flow there are two separates situations: if the mobile host joins a multicast flow, it directs the proxy agents to join the multicast group and the data paths are set up along the multicast routes; if the mobile host initiated a unicast flow, the paths may be set up by unicast routing or by multicast routing.

As well as the basic RSVP, the two most important messages in MRSVP are *path* and *Resv* messages; nevertheless in this case we have even a passive version of these messages. Describing briefly the mechanism of the reservation protocol, the sender host sends active path messages to the flow destination periodically, and its proxy agent will send passive path messages at the same time. When the mobile host and the proxy agents receive the path messages, that will send respectively a Resv message for the active reservation and a Resv message to make a passive reservation. Resv messages for active reservations are converted to Resv messages for passive reservation when they are forwarded towards nodes, which contain only proxy agents of mobile senders and non-active senders. In addition to these messages, present even in the RSVP, additional messages are required in MRSVP, to handle multicast groups or to deal with the MSPEC list.

Finally we can say that the main features of this protocol are the ability to make advance reservations at the multiple locations in the MSPEC and the use of passive and active reservation in order to improve the network resource utilization.

4.3.3 ITSUMO

ITSUMO [36] is a research project that focuses on the design of the next generation IP networks. Its goal is to use IP and next generation wireless technologies to design a unique platform to support end-to-end services for real-time and non real-time applications.

In our context, ITSUMO is a QoS architecture following a bandwidth broker-like mechanism. The architecture proposed is based on DiffServ, in that traffic is aggregated and forwarded in a backbone based on PHBs. In this architecture there is at least one QoS global server (or QGS) and several QoS local nodes (or QLN). The QGS has the global information about the resources available in the whole domain. The MN interacts only with the QGS to request certain degrees of QoS in this domain. The QGS decides what services are available for the MNs and send the decisions to particular QLNs. Finally we can say that the QGS is an intelligent entity in the control plane for QoS negotiation and signaling.

The QLNs are the ingress nodes of the DS domain. QLNs have the local information about the resource in the local domain and could be part of the edge router and controller (ERC) or could reside inside the radio access network (RAN), for example inside one base station (BS). These devices manage transport functions and are less intelligent than QGS. Figure 4.4 depicts the scenario of the ITSUMO approach.

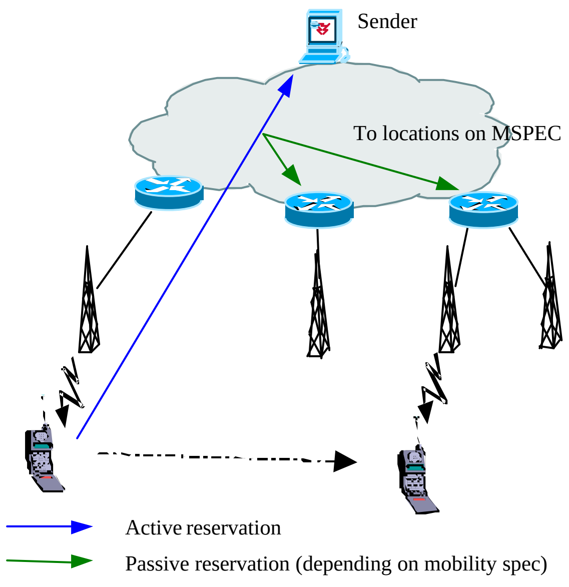


Figure 4.4 ITSUMO approach [23]

The QoS guarantee is obtained thanks to the service level specification (SLS), usually agreed on by both the user and the service provider when a user signs up with a service provider. A user has to contact the service provider to change the SLS in a wired network, after this negotiation the user can utilise the new SLS. Since the QGS has a quite complete picture of the state of the network, dynamic SLS can be achieved easily and efficiently in the ITSUMO approach. This represents a great advantage for mobility requirements of the next generation of wireless IP networks.

The ITSUMO approach offers a spectrum of 12 possible different types of service classes. Only two parameters are considered: *latency* and *loss*.

For each parameter, different possible values are defined: High, moderate, and low for latency; and High, moderate, low and none for loss. The possible combinations are illustrated in the following figure 4.5.

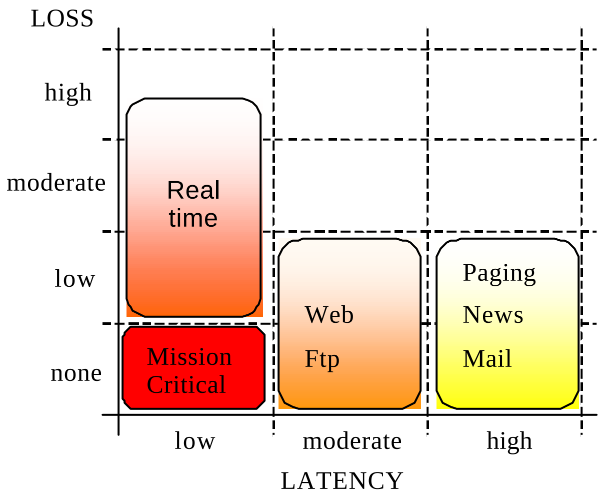


Figure 4.5 ITSUMO classes of service [23].

The main disadvantage of this approach is tied with the centralized administration and maintenance of the resources. One of the bets advantages in this architecture is the separation between the control plane and the transport plane; by separating these planes, the architecture is more flexible, it is easy to add new services, and more efficient for mobile environment.

Even though ITSUMO and MRSVP can be classified in the same category, advance reservation mechanism, the ITSUMO adopts a different philosophy. In fact although the MN has to explicitly request a reservation and specify a mobility profile, the principal role in the advance reservation is played by the QGS. In fact based on the local information and mobility pattern negotiated in the SLS, the QGS decides how much bandwidth should be reserved for each QLN, transmits this information to the QLNs and then periodically updates the QLNs likely to be visited by a MN. Finally we can say that, opposite from the MRSVP approach, where the advance reservation has to be signaled explicitly by the MN to every station according its mobility pattern, in the ITSUMO approach the QGS process as the mobility pattern and makes the advance reservation.

4.4 Additional mechanisms

Micro-mobility protocols will have to support the delivery of a variety of traffic including best effort and real-time traffic. There has not been much work on a suitable QoS model for micro-mobility. Extending the differentiated services model to micro-mobility seems like a logical starting point. However, the differentiated services concepts such as aggregation, per-hop behavior, service level agreement and slow time scale resource management may be impractical in wireless IP networks. For example, it may be impractical to allocate resources at every base station in a wireless access network in support of a service level agreement that offers assured service, or to use traffic-engineering techniques that promote under-utilization of wireless links in support of some per-hop behavior characteristic. In Mobile IP a host acquires a new address each time it hands off to a new base station. A new reservation between the mobile host and its home agent would be triggered in this case. This would be rather disruptive in support of micro-mobility because most of the path between the home agent and mobile host would remain unchanged. Work on QoS support for micro-mobility is predicated on differentiated services first being resolved in the wired network.

At the moment several approaches have been proposed, but none of those provide a complete support to ensure successful end-to-end QoS delivery.

Basically we can distinguish four different types of approach, classified as below [23]:

- Strict shaping and policing at the network edge
- Advance reservation
- Coupling of mobility and resource control
- Pre-handover negotiation and context transfer solution

Strict shaping and policing at the network edge

At the network edge, the access routers and the gateways accomplish security management for each packet arriving, based on IP address and port number. Those devices are often called firewalls, but it is possible to use the firewall even to perform admission control mechanisms. The access routers perform admission control for packets from the mobile node, and the gateways for flows emerging from external networks respectively.

According to the SLA established between the customer and the owner of the network domain, the edge node will decide if the arriving packet can be accepted or not inside the domain.

The best solution to accomplish this function is a bandwidth broker-like mechanism, which assures an efficient use of the network resources by means of a general vision of the network state and a dynamic control mechanism. This method operates with aggregate service classes and not with per-flow mechanisms like RSVP.

Coupling of mobility and resource control

In the Integrated Service Architecture, in order to improve the behaviour of the reservation mechanism in a mobile environment, a coupling of the micro-mobility protocol and QoS management can be adopted. This coupling assures a faster reservation set up after a mobility event such as a handoff. We can distinguish three levels of coupling mechanism:

De-coupled.

In this option the mobility protocol operates independently from the QoS mechanism. This approach guarantees independence between different protocols, but does nit provide a rapid restoration of the resources and an efficient use of the network capabilities.

Losely coupled.

This approach uses the mobility events to trigger the generation of RSVP messages. It guarantees a shorter time between an handoff and the reservation setup, minimizes the disruption to the application's traffic streams, it also avoids the problem of trying to install a reservation across the network before the routing update information has been propagated. The latency for installing the reservation can also be reduced. The two major disadvantages of this approach are obviously a more complicated implementation within the inter-mediate nodes across the network, and the generation of bursts after a mobility event.

Closely coupled

The closely coupled uses the same signaling mechanism to propagate the mobility and QoS information. This approach minimizes the disruption to traffic streams after a handover by ensuring that the reservation is in place as soon as possible after the handover by installing routing and QoS information simultaneously in a localized area. It also provides a means to install multiple reservations using one signaling message; further more the reservation along the old path can also be explicitly removed. The major disadvantage of this approach is that it requires support from particular micro-mobility mechanisms, so that the opaque QoS information can be conveyed across the network. A clear example of this approach is represented with the INSIGNIA protocol.

Advance reservation

The currently adopted reservation protocol in Internet (RSVP) is not suitable to guarantee a reservation in a mobile environment because after a handover a mobile node should require resources, which the new AR could not guarantee. To avoid this problem a mobile host needs to make *advance resource reservations* at the multiple locations it may visit during the lifetime of the connection.

We can find several proposals for advanced reservations in the Internet Community that can be classified into two groups, depending on the techniques they use:

- *Admission control priority*
- *Explicit advanced reservation signaling*

The basic idea in the *Admission control priority* mechanism is that a wireless network must give higher priority to a handover connection request than to new connection requests. Terminating an established connection from a node that has just arrived to the cell is less desirable than rejecting a new connection request. The key problem is to calculate effectively the amount of bandwidth necessary to guarantee the handover management.

The *explicit advanced reservation* signaling strategies offer good service guarantees in a mobile environment, allowing the mobile host to make resource reservations at all the locations it may visit during the lifetime of the connection. The most important proposals in this approach are: Mobile RSVP and ITSUMO.

Pre-handover negotiation and context transfer solution

In this approach, when a mobile node is going to make a handover, it can request some indication of resource availability from neighboring access routers. This would allow to reduce the time during which the mobile has no specific resources allocated to it, but requires support from the convergence layer between the IP-layer and link-layer.

In this field, the IETF Seamoby is studying context transfer as part of providing seamless mobility.

II. PART

5 IP QoS Implementation issues

In this chapter we present a brief overview of the most significant congestion control mechanisms, buffer management strategies and queuing disciplines, finally we introduce one example of IntServ over DiffServ implementation on a Linux operating system

5.1 TCP Congestion Control mechanisms

The basic congestion control mechanisms rely on Transmission Control Protocol (TCP)[42] to achieve this function. The TCP is a reliable connection-oriented stream protocol in the Internet Protocol suite. To maintain this virtual circuit, TCP at each end needs to store information on the current status of the connection, e.g., the last byte sent. TCP is called connection-oriented because it starts with a 3-way handshake, and because it maintains this state information for each connection. TCP is called a stream protocol because it works with units of bytes.

TCP is responsible for delivering error-free in-order data to the application at the other end. This reliability is achieved by assigning a sequence number to each byte of data that is transmitted. It is also required of the receiving TCP to send positive acknowledgments (ACKs) back to the data sender. This acknowledgment mentions the next byte of data expected by the receiver. Only when the sender receives ACKs does it continue sending more data, this is called a self-clocking mechanism. Data is divided into segments (or packets) not exceeding a maximum segment size (MSS) with default value 536 bytes.

The sequence numbers also help in eliminating duplicate segments. Corruption of data is detected by a checksum, which is part of each transmitted segment. The receiver handles corrupted data segments by discarding them. If the segment is corrupted or lost, the sender eventually times out and retransmits the corrupted/lost segment.

Slow Start

Early TCP implementations followed a simple go-back-n model without any elaborate congestion control techniques. In modern TCP implementations, every TCP connection starts off in the “slow start” phase. The key concept to understand all the following congestion control mechanisms is that the TCP sender assumes congestion in the network when it times out waiting for an ACK.

The slow start algorithm uses a new variable called congestion window (*cwnd*). The sender can only send the minimum of *cwnd* and the receiver advertised window which we call rwnd (for receiver flow control). Slow start tries to reach equilibrium by opening up the window very quickly. The sender initially sets *cwnd* to 1 (in RFC 2581 [43] is suggested an initial window size value of 2 and in RFC 2414 is suggested min(4MSS,max(2MSS,4380 bytes))), and sending one segment. For each ACK that it receives, the *cwnd* is increased by one segment. The sender always tries to keep a window's worth of data in transit. Increasing *cwnd* by one for every ACK results in an exponential increase of *cwnd* over round trips. In this sense, the name slow start is a misnomer.

Congestion Avoidance

TCP uses another variable ssthresh, the slow start threshold, to ensure *cwnd* does not increase exponentially forever. Conceptually, ssthresh indicates the "right" window size depending on the current network load. The slow start phase continues as long as *cwnd* is less than ssthresh. As soon as it crosses ssthresh, TCP goes into "congestion avoidance." In congestion avoidance, for each ACK received, *cwnd* is increased by 1/ *cwnd* segments. This is approximately equivalent to increasing the *cwnd* by one segment in one round trip (an additive increase), if every segment (or every other segment) destination is acknowledged.

As said above, the TCP sender assumes congestion in the network when it times out waiting for an ACK. ssthresh is set to max(2, min(*cwnd* /2, rwnd)) segments, *cwnd* is set to one, and the system goes to slow start. The halving of ssthresh is a multiplicative decrease. The Additive Increase Multiplicative Decrease (AIMD) system has been shown to be stable. Note that increasing the window only stops when a loss is detected.

Retransmission Timeout

To manage these congestion control techniques, TCP maintains an estimate of the round trip time (RTT), the time it takes for the segment to travel from the sender to the receiver plus the time it takes for the ACK (and/or any data) to travel from the receiver to the sender. The variable RTO (Retransmit Time Out) maintains the value of the time to wait for an ACK after sending a segment before timing out and retransmitting the segment. If RTO is too small, TCP will retransmit segments unnecessarily. If the estimate is too large, the actual throughput will be low because even if a segment gets lost, the sender will not retransmit until the timer goes off. An estimate of the mean and mean deviation round trip time is maintained via a low-pass filter.

Duplicate Acknowledgments and Fast Retransmit

If a TCP receiver receives an out of order segment, it immediately sends back a duplicate ACK (dupack) to the sender. The duplicate ACK indicates the byte number expected (thus it is easy to infer the last in-order byte successfully received). For example, if segments 0 through 5 have been transmitted and segment 2 is lost, the receiver will send a dupack each time it receives an out of order segment, i.e., when it receives segments 3, 4, and 5. Each of these dupacks indicates that the receiver is expecting segment 2 (or expecting byte 1024 assuming segments 0 and 1 are 512 bytes each).

The fast retransmit algorithm uses these dupacks to make retransmission decisions. If the sender receives 3 dupacks (was chosen to prevent spurious retransmissions due to out-of-order delivery), it assumes loss and retransmits the lost segment without waiting for the retransmit timer to go off. TCP then reduces ssthresh as previously described (to half *cwnd*), and resets *cwnd* to one segment. TCP Tahoe was one of the first TCP implementations to include congestion control. It included slow start, congestion avoidance and fast retransmit.

Fast Recovery and TCP Reno

TCP Reno [44] retained all the enhancements in TCP Tahoe, but incorporated a new algorithm, the fast recovery algorithm. Fast recovery is based on the fact that a dupack indicates that a segment has left the network. Hence, when the sender receives 3 dupacks, it retransmits the lost segment, updates ssthresh, and reduces *cwnd* as in fast retransmit (by half). Fast recovery, however, keeps track of the number of dupacks received and tries to estimate the amount of outstanding data in the network. It inflates *cwnd* (by one segment) for each dupack received, thus maintaining the flow of traffic. Thus, fast recovery keeps the self-clocking mechanism alive. The sender comes out of fast recovery when it receives an acknowledgment for the segment whose loss resulted in the duplicate ACKs. TCP then deflates the window by returning it to ssthresh, and enters the congestion avoidance phase.

If multiple segments are lost in the same window of data, on most occasions, TCP Reno waits for a retransmission timeout, retransmits the segment and goes into slow start mode. This happens when, for each segment loss, Reno enters fast recovery, reduces its *cwnd* and aborts fast recovery on the receipt of a partial ACK. (A partial ACK is one which acknowledges some but not all of the outstanding segments.) After multiple reductions of this sort, *cwnd* becomes so small that there will not be enough dupacks for fast recovery to occur and a timeout will be the only option left.

TCP New Reno

In Reno, partial ACKs bring the sender out of fast recovery resulting in a timeout in case of multiple segment losses. In NewReno TCP [44], when a sender receives a partial ACK, it does not come out of fast recovery. Instead, it assumes that the segment immediately after the most recently acknowledged segment has been lost, and hence the lost segment is retransmitted. Thus, in a multiple segment loss scenario, NewReno TCP does not wait for a retransmission timeout and continues to retransmit lost segments every time it receives a partial ACK. Thus fast recovery in NewReno begins when three duplicate ACKs are received and ends when either a retransmission timeout occurs or an ACK arrives that acknowledges all of the data up to and including the data that was outstanding when the fast recovery procedure began. Partial ACKs deflate the congestion window by the amount of new data acknowledged, and then add one segment and re-enter fast recovery.

TCP with Selective Acknowledgments (SACK)

Based on the assumption that the receiver may not acknowledge every individual segment, another way to deal with multiple segment losses is to tell the sender which segments have arrived at the receiver. Selective Acknowledgments (SACK) TCP does exactly this. The receiver uses each TCP SACK block to indicate to the sender one contiguous block of data that has been received out of order at the receiver. When the sender receives SACK blocks, they are used to maintain an image of the receiver queue, i.e., which segments are missing and which have made it to the receiver. Using this information, the sender retransmits only those segments that are missing, without waiting for a retransmission timeout. Only when no segment needs to be retransmitted, new data segments are sent out [45].

The main difference between SACK and Reno is the behavior in the event of multiple segment losses. In SACK, just like Reno, when the sender receives 3 dupacks, it goes into fast recovery. The sender retransmits the segment and halves *cwnd*. SACK maintains a variable called a pipe to indicate the number of outstanding segments, which are in transit. In SACK, during fast recovery, the sender sends data, new or retransmitted, only when the value of the pipe is less than *cwnd*, i.e., the number of segments in transit are less than the congestion window value. The value of the pipe is incremented by one when the sender sends a segment (new or retransmitted) and is decremented by one when the sender receives a duplicate ACK with SACK showing that new data has been received. The sender decrements a pipe by 2 for partial ACKs . As with NewReno, fast recovery is terminated when an ACK arrives that acknowledges all of the data up to and including the data that was outstanding when the fast recovery procedure began.

Forward Acknowledgments (FACK)

Forward Acknowledgments (FACK) also aims at better recovery from multiple losses. The name "forward ACKs" comes from the fact that the algorithm keeps track of the correctly received data with the highest sequence number. In FACK, TCP maintains 2 additional variables: (1) *fack*, that represents the forwardmost segment that has been acknowledged by the receiver through the SACK option, and (2) *retran data*, that reflects the amount of outstanding retransmitted data in the network. Using these 2 variables, the amount of outstanding data during recovery can be estimated as forward-most data sent forward - most data ACKed (fack value) + outstanding retransmitted data (retran data value). TCP FACK regulates this value (the

amount of outstanding data in the network) to be within one segment of *cwnd*. *cwnd* remains constant during the fast recovery period. The fack variable is also used to trigger fast retransmit more promptly [46].

TCP Vegas

TCP Vegas [47] was presented in 1994 before NewReno, SACK and FACK were developed. Vegas is fundamentally different from other TCP variants in that it does not wait for loss to trigger congestion window reductions. In Vegas, the expected throughput of a connection is estimated to be the number of segments in the pipe, i.e., the number of bytes traveling from the sender to the receiver. To increase throughput, the congestion window must be increased. If congestion exists in the network, the actual throughput will be less than the expected throughput. Vegas uses this idea to decide if it should increase or decrease the window.

Vegas keeps track of the time each segment is sent. When an ACK arrives, it estimates RTT as the difference between the current time and the recorded timestamp for the relevant segment. For each connection, Vegas defines BaseRTT to be the minimum RTT seen so far. It calculates the expected throughput as:

$$\textit{Expected} = \textit{Window Size} / \textit{Base RTT}$$

where WindowSize is the size of the current congestion window. It then calculates the actual throughput (every RTT) by measuring the RTT for a particular segment in the window and the bytes transmitted in between.

The difference between expected and actual throughputs is maintained in the variable Diff. If Diff < a a linear increase of *cwnd* takes place in the next RTT; else if Diff > ß , *cwnd* is linearly decreased in the next RTT. The factors a and ß (usually set to 2 and 4) represent too little and too much data in the network, respectively. This is the Vegas congestion avoidance scheme.

Vegas uses a modified slow start algorithm. The modified slow start tries to find the correct window size without incurring a loss. This is done by exponentially increasing its window every other RTT and using the other RTT to calculate Diff, when there is no change in the congestion window. Vegas shifts from slow start to congestion avoidance when the actual throughput is lower (by some value ?) than the expected throughput. This addition of congestion detection to slow start gives a better estimate of the bandwidth available to the connection.

Vegas also has a new retransmission policy. A segment is retransmitted after one duplicate ACK (without waiting for 3 dupacks) if the RTT estimate is greater than the timeout value. This helps in those cases where the sender will never receive 3 dupacks because lots of segments within this window are lost or the window size is too small. The same strategy is applied for a non-duplicate ACK after a retransmission. Table 5.1 summarizes the different TCP variants.

	Tahoe	**Reno**	**NewReno**	**SACK**	**FACK**	**Vegas**
In slow start, cwnd updated with every ACK as	cwnd +1	cwnd +1	cwnd +1	cwnd +1	cwnd +1	Increase every other RTT
In congestion avoidance, cwnd updated with every ACK as	cwnd +1/cwnd	Cwnd +1/cwnd	cwnd +1/cwnd	cwnd +1/cwnd	cwnd +1/cwnd	Linear increase if expected-actual<a ; linear decrease if >ß
Change from slow start to congestion avoidance	Cwnd =ssthresh	Cwnd =ssthresh	Same, but ssthresh may be estimated	Cwnd =ssthresh	Cwnd =ssthresh	Expcted-actual < ?
Fast recovery	None	Terminate with partial or full ACK	Continue with partial ACK	Continue with partial SACKs and send if pipe<cwnd	Send as long as out-standing data<cwnd	Retransmit with ACK if RTT> timeout
ACK format required	ACK	ACK	ACK	SACK	SACK	ACK

Table 5.1TCP overall

5.2 Buffering management

A scheduling discipline of a router has two fundamental components: a queuing component, controlling the order in which packets are serviced, and a discard component, controlling the timing and selection of packet discard. Packet discard happens when the input rate exceeds the output capacity for a period of time such that queue resources are insufficient to absorb this large mismatch. Packet discard is considered as last-resort response to congestion situations, because any packet that is discarded will have already consumed some level of resources in its transit through the network. However buffer management mechanisms, like the Random Early Detection (RED) reported below, prefer discard packets before the congestion event is cleared, rather than wait until all resource are fully consumed, resulting in a complete discard.

The default packet discard algorithm, normally used in conjunction with FIFO queuing, is used to discard packets when there is no further queue space left. This is called *tail drop,* as packets are discarded from the logical tail of the queue. Packets continue to be discarded until space is available on the queue. This discard algorithm does not exhibit fair discard behaviour. One way to protect well-behaved sources from more aggressive sources is to extend the service class structure used within the queuing strategy.

Per-service class queue management is one instance of *early drop,* where the discard is triggered prior to complete resource exhaustion. The intent of an early drop discard strategy is to obtain a form of fair sharing; in fact the heaviest bursting flows are discarded prior to exhaust queue resources in order to leave space for other small flows.

Before starting with the description of the two most important queue active management mechanisms, we introduce two other concepts: one is that discard of the most recently arrived packets is not the only approach and an alternative approach can be used to discard at the head of the queue, or from a random position in order to implement a fair discard function through probability. The other is tied with the *drop precedence* mechanism. The typical applications of this mechanism in IP networks imply all traffic outside the configured profile is marked with an elevated drop precedence by the admission filter, causing discard of these packets when the overall network load exceeds a fixed threshold.

5.2.1 Random Early Detection

Floyd and Jacobson proposed Random Early Detection (RED) [48] gateways in 1993 in order to solve the problems caused by the Drop Tail (DT) queue such as global synchronization and ultimately congestion collapse. Global synchronization occurs when a significant number of TCP senders slows down at the same time and leads to underutilization of scarce network resources (e.g. the bottleneck link). Global synchronization is likely to happen when a router drops many consecutive packets in a short period of time. In this case each TCP sender detects loss and reacts accordingly, going into slow start reducing the overall transmission efficiency.

RED refined the random early drop algorithm by Hashem in such a way that small bursts can pass through such a filter without experiencing elevated drop probability, while larger overload conditions will trigger increasingly higher discard rates.

RED can be classified as an Active Queue Management (AQM) mechanism, which is not designed to operate with any specific protocol, but performs better with protocols that perceive drops as indications of congestion (e.g. TCP). RED gateways require the user to specify five parameters: the maximum buffer size or queue limit (QL), the minimum ($\min_{th}$) and maximum ($\max_{th}$) thresholds of the "RED region", the maximum dropping probability ($\max_p$), and the weight factor used to calculate the average queue size (w_q). QL can be defined in terms of packets or bytes. Note that when DT is implemented at a router, QL is the only parameter that needs to be set. A RED router uses early packet dropping in an attempt to control the congestion level, limit queuing delays, and avoid buffer overflows. Early packet dropping starts when the average queue size exceeds $\min_{th}$. RED was specifically designed to use the average queue size (avg), instead of the current queue size, as a measure of incipient congestion, because the latter proves to be rather intolerant of packet bursts, as we will see shortly. If the average queue size does not exceed $\min_{th}$ a RED router will not drop any packets. avg is calculated as an exponentially weighted moving average using the following formula:

$$avg_i = (1 - w_q) \times avg_{i-1} + w_q \times q$$

where the weight w_q is commonly set to 0.002 [49], and q is the instantaneous queue size. This weighted moving average captures the notion of long-lived congestion better than the instantaneous queue size. Had the instantaneous queue size been used as the metric to determine whether the router is congested, short-lived traffic spikes would lead to early packet drops. So a rather underutilized router that receives a burst of packets can be deemed "congested" if one uses the instantaneous queue size. The average queue size, on the other hand, acts as a low pass filter that allows spikes to go through the router without forcing any packet drops (unless, of course, the burst is larger than the queue limit). The user can configure w_q and $\min_{th}$ so that a RED router does not allow short-lived congestion to continue uninterrupted for more than a predetermined amount of time. This functionality allows RED to maintain high throughput and keep per-packet delays low.

An interesting point about the RED algorithm is that it separates the decision processes of when to drop a packet, and from which packet to drop. As noted earlier, RED uses randomization to decide which packet to drop and, consequently, which connection will be notified to slow down. This is accomplished by making Bernoulli trials (e.g. coin flips) using a probability p_a, which is calculated according to the following formulae:

$$p_b = \max_p \times (avg - \min_{th}) / (\max_{th} - \min_{th})$$

and

$$p_a = p_b / (1 - count \times p_b)$$

where $\max_p$ is a user-defined parameter, usually set to 2% or 10% [49], and count is the number of packets since the last packet drop. count is used so that consecutive drops are spaced out over time. Notice that p_b varies linearly between 0 and $\max_p$, while p_a, i.e., the actual packet dropping probability increases with count. Originally, $\max_{th}$ was defined as the upper threshold; when the average queue size exceeds this limit, all packets have to be dropped (Figure 5.1.a). Later on, however, Floyd proposed a modification to the dropping algorithm (the "gentle option"), under which packets are dropped with a linearly increasing probability until avg exceeds $2\times \max_{th}$; after that all packets are dropped (Figure 5.1.b). Although $\max_{th}$ can be set to any value, a rule of thumb is to set it to three times $\min_{th}$, and less than QL [49].

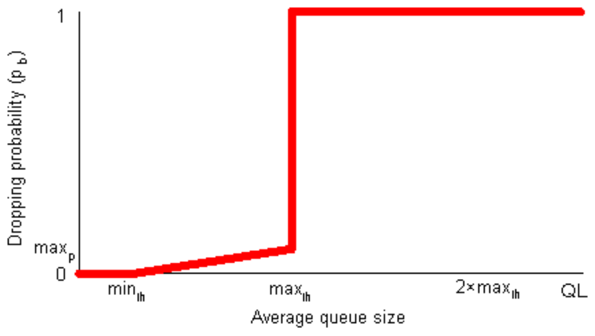


Figure 5.1 (a) Standard RED

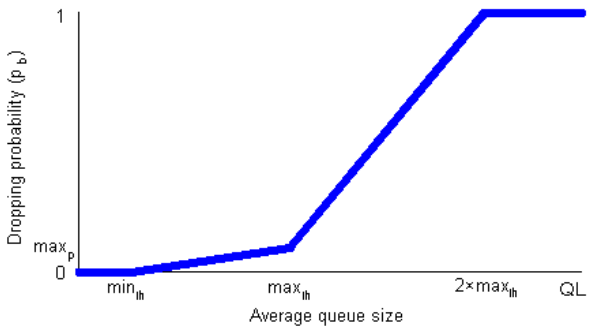


Figure 5.1 (b) Gentle option RED

Fairness is an important feature of any AQM mechanism, especially in the current Internet incarnation where all connections are treated equally. Clearly, in times of congestion, one would like to notify those connections that use most of the resources. In other words, the router should drop packets that belong to flows currently using most of the bandwidth (and buffer space). RED does not keep per-connection state, i.e., exact statistics of resource utilization, so it cannot be absolutely accurate in dropping packets according to the exact proportion of the resources that a connection is using. Nevertheless, its inventors show that the number of packets that RED drops from a single flow is roughly proportional to its share of the bandwidth, and that it can be effective in notifying the connections with the higher percentage of bandwidth utilization. In contrast, DT tends to cause packet drops in batches without taking into account the proportion of the resources a flow is using [49].

RED can easily be deployed on the Internet because it is supported by router vendors and is implemented in a number of products. Nonetheless, RED is not currently widely used, despite the eight years that have passed since its introduction, the subsequent studies of its efficiency in controlling congestion, and the IETF (Internet Engineering Task Force) recommendation for the wide deployment of RED some years ago [48].

5.2.2 Explicit Congestion Notification

The RED algorithm already has a notion of “marking” packets to signal congestion. However, the design of RED only knows to drop marked packets in order to provide congestion feedback to the sender. As previously mentioned, dropping packets are nearly always a less-than optimal signal of congestion. We would rather treat marked packets in a way which delivers them yet still conveys congestion.

Explicit Congestion Notification [51] is an extension proposed to RED, which marks a packet instead of dropping it when the average queue size is between $\min_{th}$ and $\max_{th}$. Since ECN marks packets before congestion actually occurs, this is useful for protocols like TCP that are sensitive to even a single packet loss. After receipt of a congestion marked packet, the TCP receiver informs the sender (in the subsequent ACK) about incipient congestion, which will in turn trigger the congestion avoidance algorithm at the sender. ECN requires support from both the router as well as the end hosts. Packets from flows that are not ECN capable will continue to be dropped by RED.

Router side support for ECN can be added by modifying last RED implementations. For packets from ECN capable hosts, the router marks the packets rather than dropping them (if the average queue size is between $\min_{th}$ and $\max_{th}$). It is necessary that the router identifies that a packet is ECN capable, and should only mark packets that are from ECN capable hosts. This uses two bits in the IP header. The ECN Capable Transport (ECT) bit is set by the sender end system if both the end systems are ECN capable (for a unicast transport, only if both end systems are ECN-capable). Packets encountering congestion are marked by the router using the Congestion Experienced (CE) (if the average queue size is between $\min_{th}$ and $\max_{th}$) on their way to the receiver end system (from the sender end system), with a probability proportional to the average queue size following the procedure used in RED routers, described above. Bits 10 and 11 in the IPV6 header are proposed respectively for the ECT and CE bits. Bits 6 and 7 of the IPV4 header DSCP field are also specified for experimental purposes for the ECT and CE bits respectively.

The TCP interaction is slightly more complicated. In order to add ECN to TCP it has been necessary to specify two new flags in the reserved field of the TCP header. Bit 9 in the reserved field of the TCP header is designated as the ECN-Echo (ECE) flag and bit 8 is designated as the Congestion Window Reduced (CWR) flag. These two bits are used both for the initializing phase, in which the sender and the receiver negotiate the capability and the desire to use ECN, as well as for the subsequent actions to be taken in case there is congestion experienced in the network during the established state.

The initial TCP SYN handshake include the addition of ECN-echo capability and CWR flags to allow each system to negotiate with its peer as to whether it will properly handle packets with the CE bit set during the data transfer. The sender sets the ECT bit in all packets sent. If the sender receives a TCP packet with the ECN flag set in the TCP header, the sender will adjust its congestion window *cwnd* if it has undergone fast recovery from a single lost packet. The next sent packet will set the TCP CWR flag, to indicate to the receiver that it has reacted to the congestion. The additional limitation is that the sender will react in this way at most once every RTT interval. Further, TCP packets with the ECN-Echo flag set will have no future effect on the sender within the same RTT interval. The receiver will set the ECN-Echo flag in all packets once it receives a packet with the CE bit set. This will continue until it receives a packet with the CWR bit set, indicating that the sender has reacted to the congestion. The ECT flag is set only in packets that contain a data payload. TCP ACK packets that contain no payload should be sent with the ECT bit clear.

Simulation of ECN using a RED marking function indicates slightly superior throughput in comparison with RED as a discard packet function. The most notable improvements indicated in ECN simulation experiments occur with short TCP transactions (e.g. Web transactions), where a RED packet drop of the initial data packet may cause a six second retransmit delay. Comparatively, the ECN approach allows the transfer to proceed without lengthy delay. However, widespread deployment of ECN is not considered highly likely in the near future, at least in the context of Ipv4, because there has been no explicit standardization of the field within Ipv4 header to carry this information.

5.3 Queuing Disciplines

This section presents the queuing disciplines which can be used to implement Integrated and Differentiated Services and which we will use in the next chapters. In a packet switched network, such as the Internet, queuing discipline has a very important role especially when quality of service is concerned. A router is a shared resource within a network, where each application generates requests for a share of this resource to packets through the network. Each data packet may be considered a request for service from the router, when multiple requests arrive at the same time or the time taken to service requests overlap with the arrival of other requests there will be contention of resources. Queuing is the act of storing packets into a place where they are held for subsequent processing. A typical router has one process for each input and output port. Forwarding process between the ports determines the destination interface to which a packet should be passed. Thus the role of the router is to bind the input processes to the forwarding and scheduling processes and then to the output processes. This is done by the help of the input and output queues, as shown in Figure 5.2

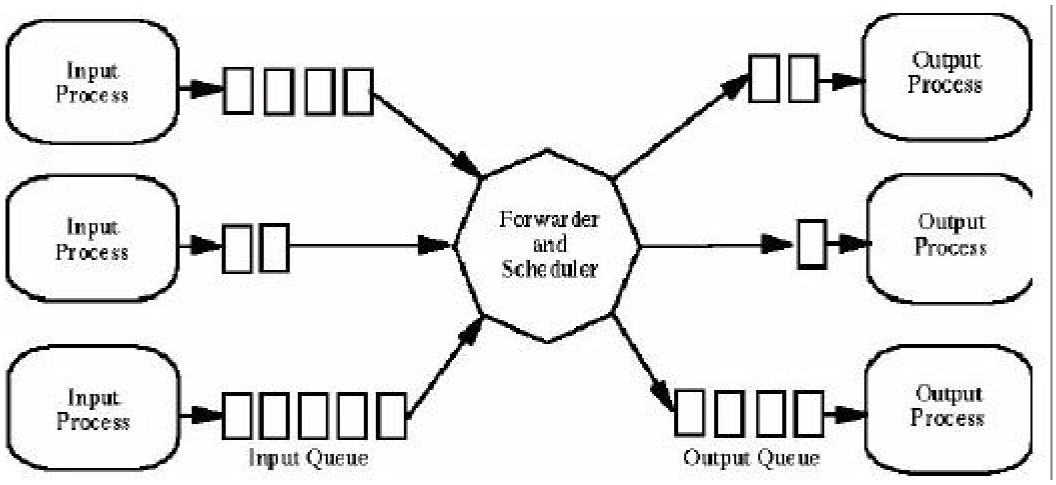


Figure 5.2 Router queuing and scheduling

Queuing disciplines can be characterized as either work-conserving or non-work-conserving. A work-conserving discipline ensures that the server is busy whenever the service queue contains queued requests, whereas a non-work-conserving discipline may allow the server to be idle in such a situation. All work-conserving queuing disciplines that implement differentiated service responses obey a law of conservation of resources. The law of conservation states that any improvement in the average delay characteristics provided to one class of traffic will be matched by a corresponding decline in the delay characteristics of other classes of traffic; for non-work-conserving queuing instead this rule can be less strict [55].

The choice of the scheduling algorithm is one of the most fundamental decision for providing quality of service. Each queuing discipline has its own strengths and drawbacks. In addition, the configuration of the scheduler and queue sizes can be extraordinarily difficult. If the queue is too deep, latency and jitter will increase. On the other hand, too shallow a queue may discard a significant amount of packets. For example, for efficient operation of TCP flows, the router queues should be of a comparable size to the delay bandwidth product of the output link, with which the queue is associated; but for UDP flows the design choice could be nless simple [55].

5.3.1 FIFO Queuing

FIFO (First In, First Out) queuing is a simple work-conserving queuing discipline and is considered to be the standard method for store and forward traffic handling. Packets that enter the input interface queue are placed into the appropriate output interface queue in the order in which they are received. FIFO queuing is fast, since it does not cause computational overhead for the packet processing, because FIFO does not reorder packets, and the maximum jitter added by a FIFO queue is proportionate to the configured queue size.

If the network operates with a sufficient level of transmission capacity and traffic bursts do not cause packet discard, FIFO queuing is very efficient, because as long as the queue depth remains sufficiently short, the average packet-queuing delay is an insignificant fraction of the end-to-end packet transmission time.

When the load on the network increases, the traffic bursts cause a queuing delay and if the queue is fully populated, all subsequent packets are discarded. Further, FIFO generally favors non-rate-adaptive applications over those that adapt their rate to the prevailing network load, so that an uncontrolled or bursting source may consume the entire queue, causing other active flows to experience varying levels of jitter and loss. Since FIFO cannot offer service differentiation, real-time data will suffer from network congestion and adequate QoS cannot be achieved. A FIFO queuing system may, for example, delay or discard real-time voice or video streaming data and allow the queue system to fill with noontime-critical TCP background bulk data transfers.

Finally, we can conclude saying that FIFO queuing represents the obvious solution for a network without QoS policy, generally applied at the edge of the network; where the traffic density is lower and the packet transit rates are lower, permitting a higher processing overhead per packet to be added without impairing the total capacity of the network. The interior of the network uses FIFO queuing instead, because is computationally efficient within the high-speed, high-volume network core [55].

5.3.2 Priority Queuing

Priority queuing is considered a primitive form of traffic differentiation. Priority queuing is based on the concept that certain types of traffic are always transmitted ahead of other types of traffic. Ordering is based on user-defined criteria and several levels of priorities can also exist.

The scheduling algorithm is relatively straightforward: a packet is scheduled from the head of service queue q as long as all queues of higher priority are empty. The associated discard algorithm can be preemptive or non-preemptive. A preemptive discard algorithm will attempt to discard a packet of a lower priority to make space to queue a higher-priority packet; using a non-preemptive discard, a packet is placed into the relevant service queue whenever there are available queue resources. Typical implementations of priority queuing can be visualized as a collection of FIFO output queues, and also include the ability to define queue lengths for each priority level.

Priority queuing offers very efficient service differentiation and it does not need complex computing. Within a priority queuing discipline, the granularity in identifying traffic to be classified into each queue is quite flexible. For example TCP traffic can be easily prioritized ahead of UDP packets.

In theory, many levels of priority are possible, while in practice a more restricted set of priority is desirable, such as a three-level approach of high, medium, and low. Increasing the number of priority levels does not necessarily create an increased number of predictable service outcomes; in fact as long as the higher priority queues are serviced to exhaustion before lower priority queues, the normal traffic has to wait in queue, which can cause starvation for best-effort traffic. The main problem of this queuing discipline is in fact tied with the buffer starvation of normal traffic, occurring when the rate of arrival of lower priority packets exceeds the service rate available for this priority level or high-priority traffic is unusually high.

Nevertheless, priority queuing has been used for a number of years as a simple method of differentiating traffic into various classes of service. This mechanism, however, does not scale adequately to provide the desired performance at higher speeds. A solution to protecting the integrity of the network when using priority queuing is to associate admission controls for high-priority traffic, ensuring that the amount of the premium traffic admitted into the network will not consume all available network resources [55].

5.3.3 Class Based Queuing

Class-Based Queuing (CBQ) or custom queuing (also referred to as weighted round-robin WRR) is a variation of priority queuing, and like priority queuing, several output FIFO queues can be defined. Further, for each of these, we can define the preference, the amount of queued traffic (typically measured in bytes) that should be drained on each pass in the servicing rotation, and the length of the queue. The system processes each queue individually, visiting the set of queues in round-robin or priority order. Each packet class is allocated a certain number of bytes or packets per visit, regardless of how many bytes or packets were transmitted from the class in previous rotations.

CBQ is an attempt to provide some semblance of fairness by prioritizing queuing services for certain types of traffic, while not allowing any one class of traffic to monopolize system resources and bandwidth. With CBQ, at least one packet will be removed from the component queues within each service cycle, so that all queues will receive a quantum of service. Additional packets will be de-queued from each queue until the sum of the packet sizes exceeds the queue's service amount.

CBQ provides a basic mechanism that ensures that a packet within a specific class does not sit in the outbound queue indefinitely. In addition, within bounds of variance determined by maximum packet size, the latency experienced by any queued packets in any service class is bounded to a predictable maximum delay, of course, packet loss probability is not similarly bounded because packets loss depends also on external factors (i.e. ingress rate).

The CBQ approach generally is considered a method of allocating a dedicated portion of bandwidth to specific types of traffic. The fundamental assumption made by CBQ is that resource denial is far worse than resource reduction, in fact not only data transfer but also any form of signaling is influenced by resource denial. This represents a problem for TCP applications, for example, that lose the feedback control signals coming from the receiver and can cause congestion collapse.

For several years, CBQ has been considered a reasonable method of implementing a technology that provides link sharing for classes of service and an efficient method for resource queue management. That said, CBQ is not a perfect resource-sharing algorithm, due to the variable packet sizes that exist in an IP network. Within a variable packet size regime, the CBQ service routine will continue to dequeue packets from a service class while there is still credit of bytes to transmit from this class. Nevertheless the final packet chosen can exceed the available credit, and the service interval will therefore consume a greater proportion of bandwidth than the configured allocation. A fair output can be achieved by increasing the total number of bytes dequeued in each service rotation or, in effect, slowing the service rotation time. Unfortunately, while such a measure increases the accuracy of the method, so that the packet-based actual service levels correspond more accurately to the configured service weightings, the increased rotation latency adds a large jitter factor to all traffic.

CBQ is only fair within an interval of time greater than the service rotation time. The service rotation time can be computed as the sum of the service quantum intervals for each class, divided by the egress bandwidth. For low-speed lines, or where a large number of service classes are used, or where large service quantum values are used, the service rotation interval increases and the minimum time interval where fairness can be observed will extend. For this reason, when using relatively low-speed circuits, CBQ is not the optimal solution to provide network services that include factors of bounded latency or bounded jitter. Therefore, CBQ is a reasonable choice as a bandwidth allocation queuing mechanism when allocating absolute levels of bandwidth to aggregated service flows within the core of a high-speed backbone network [55].

5.3.4 Weighted Fair Queuing

Weighted Fair Queuing (WFQ) is a flow-based algorithm, which provides fair bandwidth allocation. WFQ is based on the fair queuing scheme, which operates according to the round-robin principle. Nevertheless the scheduler uses a different service algorithm to that of strict round-robin, in order to avoid a drawback of the round-robin scheme where short packets are penalized: more capacity are consumed by the flows with longer average packet size compared to flows with shorter average packet size. In other words, the scheduler has to process an elevated service class queue more frequently than a lower priority service class queue.

Before starting to explain in more details this queuing discipline, we need to introduce the min-max weighted fair share algorithm. In this algorithm, requests are sorted in order of increasing weighted resource requirements and each request is serviced in order with a weighted proportion of the remaining resource; and if this amount is greater than the amount requested, the unused is placed back into the resource pool. To add some precision, consider the following example. Consider n requests of size s_1, s_2, …, s_n, with relative normalized weights associated w_1, w_2, …, w_n and a resource of size R. The sum of the normalized weights is W. Initially the server will attempt to allocate ($w_1 \times R / W$) amount of resources to the first request. If this amount is greater than s_1 then the demand will be fully satisfied and the balance of resources remaining is (R-s_1). If this amount is less than s_1 then the demand will only be partially met, and the balance of resources remaining is (R- ($w_1 \times R/W$)). In either case, there is now a new available resource amount to be shared over the remaining (n-1) requests, and the same process is applied again.

The ideal work-conserving weighted fair sharing model is the generalized processor sharing (GPS) proposed by Keshav in 1997. GPS implements min-max weighted fair sharing resource allocation, using an infinitely small service quota; of course, this is an unimplementable algorithm in that it ignores distortion of the resource-sharing algorithm caused by data being quantized into packets. However GPS is important because it defines a metric of effectiveness to examine implemented queuing disciplines, by measuring how close the queuing disciplines come to the ideal GPS behaviour for any particular set of conditions.

WFQ is an approximation behavior at the packet level. WFQ attempts to service a collection of queues in a min-max fair-shared manner. WFQ is min-max fair to a time level equal to the mean packet size. WFQ can be a considerable improvement in fairness accuracy and performance guarantees over CBQ. Predictability of service is a critical outcome in queuing. WFQ offers a way for a network operator to guarantee the minimum level of resources allocated to each defined service class, and this guarantee is independent of the activity level in other service classes. When used in conjunction with some form of admission control, the result will be bounded delay as well as guaranteed throughput.

However, even with admission filters, WFQ will not intrinsically bound the jitter level to any useful value. If smaller jitter bounds are required for some service classes, then some element of priority queuing must be reintroduced into the queuing discipline, as must admission control. Jitter is not the only issue with WFQ. The implementation itself is not without any additional complexity, which will have performance implications for router implementations. Generally WFQ is implemented using a GPS simulator, which involves complex computation every few microseconds on a high-speed router when there are large numbers of active classes and a large packet-forwarding rate. WFQ in fact requires a per-flow state and iterative scans of all per-flow states on packet arrival and packet departure.

The practical conclusion is that WFQ, when operating in a per-flow manner, simply does not scale to provide the desired performance in concentrated high-volume, high-speed routing environments. Instead WFQ has potential application when configured to operate within a bounded domain of a fixed number of aggregated service classes. For such reasons, per-flow WFQ is best regarded as an edge control mechanism for low-speed circuits, where bandwidth management and jitter are of fundamental importance to effective multiservice delivery. When operating in a mode of a bounded number of aggregate service classes, WFQ offers applicability to determinate per-aggregate resource allocation in larger-scale environments [55].

5.4 IntServ/DiffServ implementation on Linux

The open source UNIX–like and the most popular operating system freely available, Linux is a phenomenon on the Internet. Programmers around the world are continually implementing new features of the code or improving or fixing the bugs of the current ones, contributing in this way to the Linux development and open source movement at the same time. Linux was developed by the student Linus Torvalds as a "hobby project" and until some years ago it was not more than a "student's" written software. These days, however, Linux is a stable and effective operating system and it represents a strong competition to other UNIX versions and especially to Windows NT in the small to medium server market.

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

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

5.4.1 Network traffic control in Linux

Linux is providing various traffic control functions starting from the 2.1.x kernel versions, which provide a foundation for Linux support of IP QoS mechanisms. The kernel traffic control functions and also the user-space programs to control them are implemented by Alexey Kuznetsov, one of the key developers of the networking code for Linux.

As is shown in figure 5.3, the traffic control functions are implemented just before the packets are sent to the device drivers. Once the appropriate interface is selected by the forwarding block, the traffic control will perform queuing, and set up classes and filters that control the packets that are sent on the output interface.

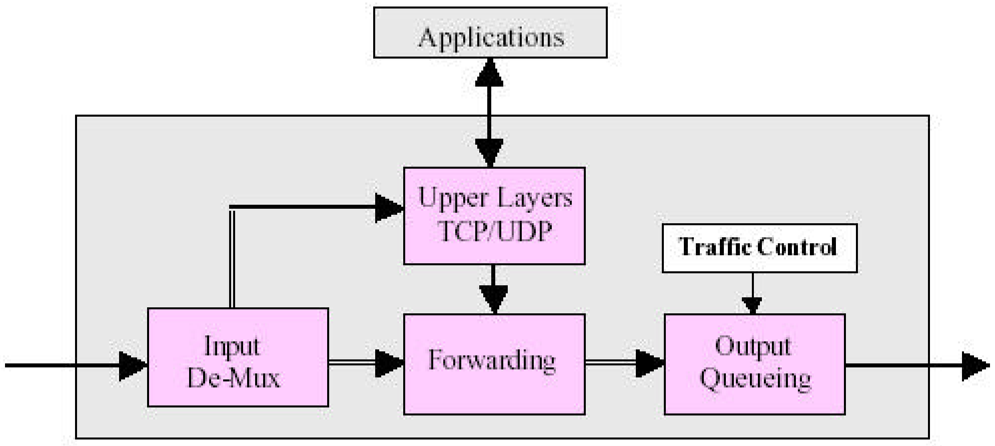


Figure 5.3 kernel processing of incoming and outgoing packets

The traffic control consists of the following major building blocks (see Figure 5.4):

- ***Queuing discipline***

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

Simple queuing disciplines such as First In First Out (FIFO) have no classes or filters related to it, while more complicated queuing disciplines may use filters to distinguish among different classes of packets. Packets can be stored in classes based on assigned priorities; i.e. a traffic class with a higher priority will be favoured to the class with lower priority. Packets are sent out to the transmission medium according to these classes and priorities. A refined queuing discipline as given in [57] with multiple classes is depicted in Figure 5.4.

- ***Classes***

Queuing discipline and classes are closely coupled. Classes and their semantics represent key properties of a queuing discipline. Classes themselves do not store packets, instead they use another queuing discipline for such a purpose. These queuing disciplines can be one of the above-mentioned queuing disciplines, a simple one with no classes or a more refined one containing classes as well. Usually each class owns a queue, but a queue can be also shared by several classes, depending on the queuing discipline used (see Figure 5.4). The packets themselves do not carry any class-related information.

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

As depicted in Figure 5.4 filters are used to classify the enqueued packets into classes. They use different packet properties such as source address, destination address, port numbers, and TOS byte in the IP header, etc as classification parameters. There are several types of classifiers implemented in Linux: fw – based on the firewall marking of the packets, u32 – based on packet header fields, route – based on the IP destination address, rsvp/rsvp6 – more refined classifiers using the five tuples RSVP.

- ***Policing***

The purpose of policing is to ensure that the traffic does not exceed the set bounds, e.g. delay or bandwidth bounds. Policing is a function which can be implemented together with filtering / classification. Filter will classify only packets matching the classification criteria. Policing is implemented either as a classification criterion of a filter, or as criterion for enqueuing packets into a class, or dropping packets within an inner class.

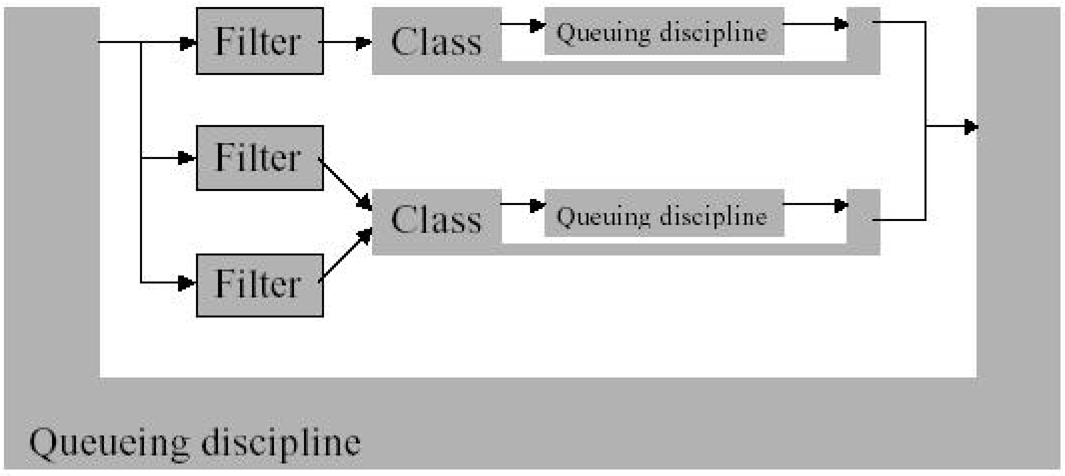


Figure 5.4 Queuing discipline with multiple classes

For building and configuring the traffic control, Alexey Kuznetsov also provided a user space tc tool [58], which enables communication with the kernel. The tc tool and the kernel communicate via the netlink socket. The tc tool is implemented in such a flexible way that one can, depending on the requirements, create different scripts to build specific traffic control on the desired network interface.

5.4.2 RSVP and Integrated Services in Linux

RSVP is a signaling protocol that does not transport any application data, rather it can be characterized as an Internet control protocol such as ICMP or IGMP. However, in Linux RSVP is implemented on the user space on top of the Kernel. The well-known implementation of RSVP is the ISI RSVP daemon, which is free software available from the Internet [59].

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

The exact functionality of the RSVP daemon will depend on whether it is running on a host or on a router. Its role in both cases is depicted in Figure 5.5. The RSVP daemon on a host will have to communicate with the applications while it may not require any traffic control-related functionality, i.e. the host may rely on over-provisioning of its outgoing link. The RSVP daemon running on a router will have to communicate with the kernel via its traffic control interface, in order to establish reservations for IntServ style flows.

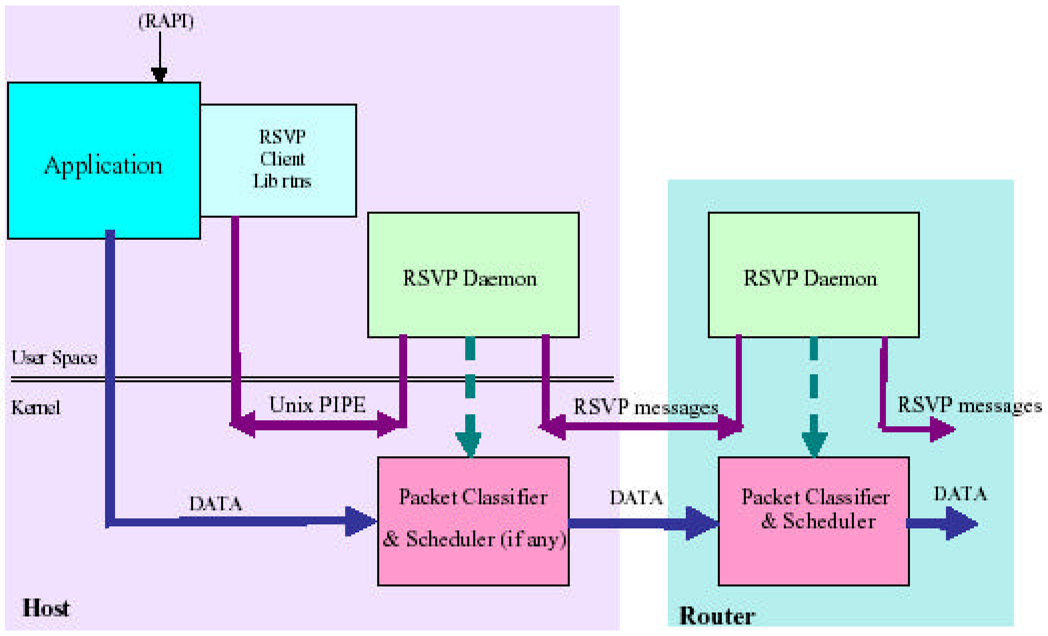


Figure 5.5 RSVP daemon roles in the host and router

RSVP daemon under Linux

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

As mentioned CBQ is the one used as a queuing discipline on the outgoing IntServ interface. Using the tc tool [58], a shell script is applied to the outgoing network interface of the network element (router and sometimes hosts), so that it has the CBQ scheduler enabled. The total amount of the link bandwidth is shared between the BE effort traffic class and RSVP traffic class. The RSVP class is further separated into CL classes and for each new GS flow accepted a new GS class is added (see Figure 5.6).

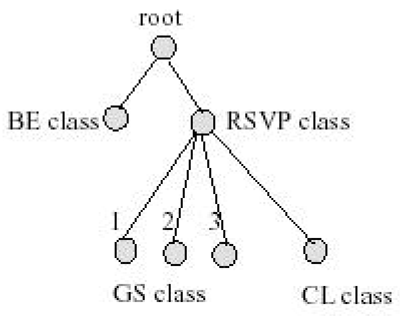


Figure 5.6IntServ CBQ classes

5.4.3 Differentiated Services under Linux

The support for Differentiated Services under Linux [56] is a design and implementation extension of the existing traffic control code. The implementation supports classification based on the DS field and configuration of the EF and AF PHB as defined in the DiffServ working group (WG). The relation of the Differentiated Services node functionality blocks and traffic control building blocks is depicted in Figure 5.7. The Classifier is implemented by means of a filter while the meter functions are also performed by filters with policing abilities (e.g. fw, u32, rsvp). The marking, shaping and dropping functions and consequently the PHBs are implemented as part of the queuing disciplines.

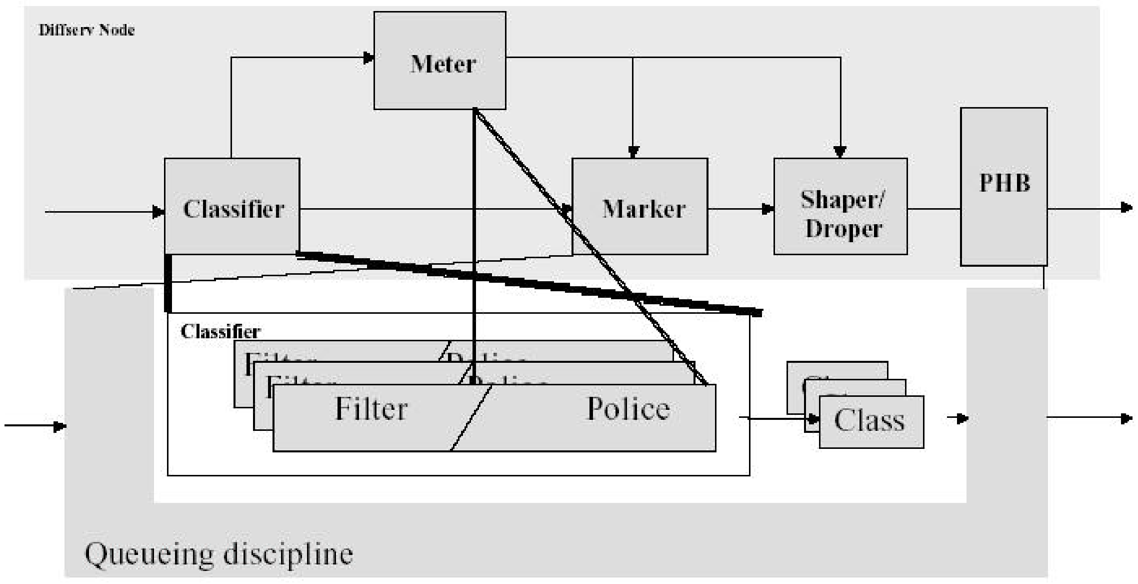


Figure 5.7 Relationship between the Linux traffic control and DiffServ node building blocks.

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

- The queuing discipline DSMark (sch_dsmark [56]) responsible for the retrieval of the DSCP
- A classifier which uses this information for aggregated classification TCindex (cls_tcindex [56])
- A new queuing discipline Generalised RED (GRED) in order to support different drop priority levels for AF PHB.

DSmark and TCindex

The DSMark queuing discipline is specific to Diffserv, while the other ones can also be used for other purposes. The sch_dsmark is responsible for the following actions on the enqueued packets:

- The extraction and the storage of the DSCP into the storage buffer field "skb->tc_index". skb-> tc_index is a new field in packet buffer descriptor sruct sk_buff,. Note that the storage buffer skb-tc_index relates a predefined class ID to a DSCP. This relation between class ID and DSCP is also stored into the skb->tc_index.
- The identification of the class that belongs to this DSCP
- Remarking and changing the DSCP of the enqueud packets.

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

GRED

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

GRED provides the possibility to use multiple RED queues, i.e., create more than one RED queues within a physical queue. Each RED queue, called Virtual Queue, operates as a standard RED queue. Each Virtual Queue is selected based on predefined configuration parameters passed on by the user and stored in the storage buffer skb->tcindex. A user can configure preferential treatment of virtual queues by setting the physical queue limit as parameter for example, or the user can assign preferences by means of the priorities.

6 Experimental design

In this chapter and in the next one we present test environments, network configurations, metrics adopted and measurements, which explore what kind of quality of service the combination of Integrated and Differentiated Services can offer in a radio access network. The scenario used in the experiments assumes that RSVP and IntServ are used in access networks and DiffServ is used in the core network.

6.1 Test Environment

The basic test network adopted, depicted in figure 6.1, contained three routers and two workstations. All five machines are Intel Pentium 1 GHz and are running the Linux Operating System (kernel version 2.4.17). For the most general configuration, all the machines are RSVP enabled (release 4.2a4-3 of RSVP ISI daemon) [59], further, besides kemi-9, all the machines are DiffServ enabled. We used an external workload generator MGEN to generate real-time traffic and the TTCP traffic generator for best-effort one; we adopted different network tools (i.e. Tcpdump or Mcalc) to gather information in order to calculate packet delay, throughput and loss statistics of analyzed flows. Time synchronization of the PCs was obtained using Network Time Protocol (NTP) [61].

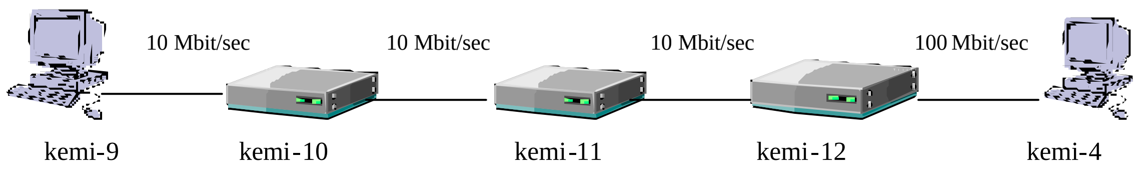


Figure 6.1 General network topology

All links, except for the link between kemi-4 and kemi-12 are 10Base-T Ethernet links belonging to an isolated LAN and these links simulate a high-speed core network, or in our case, a typical radio access network link. The link between kemi-4 and kemi-12 is a 100Base-T Ethernet link belonging to the same LAN and it simulates a very high-speed network trunk inserted in order to create a congestion situation at the edge of the core network, DiffServ or IntServ domain depending on the network configuration adopted.

6.2 IntServ/DiffServ Services mapping options

Let us assume that the DiffServ domain in the testbed can support a maximum of 10 Mbit/sec bandwidth. A possible configuration for the DiffServ domain could be to support two AF classes (AF1 and AF2) with one-drop priority level and one EF class. Thus the available bandwidth is divided between the Best Effort, AFs and EF classes. Traffic differentiation between EF, AF1, AF2 and BE could be implemented by means of CBQ or simple PQ, while AF1 and AF2 support could be implemented by means of GRED.

Imagining a static subdivision and management of the available bandwidth, the amount of bandwidth could be divided as follows: 1Mbit/s for EF, 3Mbit/s for AF1 and 3Mbit/s for AF2, and 3 Mbit/s for BE. Also the mapping between IntServ and DiffServ service classes is static and the most logical association is of the Controlled Load flows (CL) to the AF PHB and of Guaranteed Service (GS) to EF PHB (see table 6.1).

Table 6.1 IntServ/DiffServ mapping

DiffServ service class	Class bandwidth		Tspec parameters
EF	1 Mbit/s		GS [gx R S r b p m M]
AF	AF 1.1	3 Mbit/s	CL 1 [cl r b p m M]
	AF 2.3	3 Mbit/s	CL 2 [cl r b p m M]

Another important task of the edge devices (kemi-10 and kemi-12 in our scenario) between IntServ and DiffServ domains is related to the admission control procedure. In fact, before the Edge Device takes a mapping decision and it selects a DiffServ traffic class, the availability of resources should be checked. The result of the admission control procedure provides the configuration set up parameters of the per-flow traffic control elements in the ED: classifiers, policers and schedulers, but this task is beyond the scope of this thesis. In our experiments we maintained the mapping options described above except for two aspects. We have chosen in fact to map the EF service to a controlled load service and we skipped the admission control procedure supposing to study an over provisioned case.

Regarding the application side, the non-rate adaptive real-time application as e.g. VoIP that do not generate bursty traffic and are not tolerant for the jitter will be assigned to the EF PHB or highest AF PHB. Rate-adaptive real-time applications also not-bursty but jitter-tolerant like video audio-streaming will also be assigned on the lower-level class of AF PHB, instead applications more bursty like web browsing will be mapped on the lowest level of AF PHB or handled as best-effort. We have chosen a limited number of service levels because increasing the number of priority levels does not necessarily create an increased number of predictable services.

6.3 Test Cases

Preliminarily we studied the wired part of a radio access network in order to observe the behavior of different QoS network configurations without any wireless link. Finally a base station and a laptop connected by a wireless link were added to our reference topology to study QoS management in the presence of a mobile device.

The measurements for the wired part of the network consisted of three different cases. In one case, we tested in a network lightly loaded and with a small percentage of real-time traffic referred to its link bandwidth. In the second and third case the network was progressively overloaded with an increasing percentage of real-time traffic in order to create congestion at the edge router (kemi-12). A fourth test case including a wireless link is finally presented in the next chapter in order to observe the validity of our QoS network architecture in a mobile environment.

The real-time and best-effort traffic that was used in the measurements was generated by a public domain software package MGEN (version 3.3a6) [60], which understands RSVP messages and another simple workload generator: TTCP. All UDP flows were generated by MGEN, which generates real-time traffic patterns to unicast and multicast IP destinations over time according to a script file. Packet sizes and transmission rates for individual information flows can be controlled and varied and also the TOS byte can be set up using MGEN's script file; further PATH and RESV messages can be generated using MGEN in scripted mode. TCP bulk traffic was instead generated by TTCP, one of the simplest and oldest traffic generators.

The real time data emulated IP phone calls and IP video streams [62]. The characteristics of the IP phone streams were collected from IP phone measurements and two different voice codecs were considered: adpcm and gsm. The speeds of the video streams were 200kbit/s on the average and they represent bursty streams. Best-effort traffic was composed of a TCP bulk transfer and a variable (depending on the test case) number of UDP flows without any reservation generating the congestion situation. The traffic characteristics are summarized in tables 6.2 and 6.3.

Table 6.2 Generated flows characteristics

STREAM	Packet size (bytes)	Protocol	Bit-rate type	Packet rate (packets/sec)	Data rate (kbit/sec)	Number of buffer
VoIP(adpcm)	104	UDP	CBR	50	41.6	
VoIP (gsm)	36	UDP	CBR	50	14.4	
Video	300	UDP	VBR	84	200	
BE	250	UDP	VBR	300	600	
BE	1024	TCP				20000

Where the packet size indicates the UDP packet payload size for UDP flows and the buffer's length of TCP bulk transfer. The packet rate and data rate parameters for VBR flows indicate the average transmitted packet rate and the average transmitted data rate of a Poisson process packet generation. A complex workload was chosen with several kinds of flow ranging in a set of different packet sizes and packet rates in order to examine network behavior in quite a realistic situation. The next table presents a general view of traffic generated in each test case.

Table 6.3 Workload description used in the test cases

Data flows	Case 1	Case 2	Case 3
Number of VoIP (adpcm) flows	5	5	5
Number of VoIP (gsm) flows	5	5	5
Number of Video flows	5	5	5
Number of BE/UDP flows	2	20	5
Number of (prioritized) 600 Kbit/sec flows	0	0	15
Number of bytes of TCP bulk transfer (MB)	20,48	20,48	20,48
Total amount of real-time data (Kbit/sec)	2280	2280	10280
Total amount of traffic in the EF class (Kbit/sec)	208	208	808
Total amount of traffic in the AF1.1 class (Kbit/sec)	72	72	6072
Total amount of traffic in the AF2.3 class (Kbit/sec)	1000	1000	3400
Total amount of BE/UDP traffic (Kbit/sec)	1200	12000	3000

6.4 Network Configurations

In order to observe advantages and disadvantages of different kinds of QoS architectures and to compare quantitatively these different solutions, in each test case the test network was configured to represent four different types of QoS architecture:

- Best Effort
- DiffServ
- IntServ
- IntServ/DiffServ Combination

Best-effort Configuration

In the best effort configuration pfifo_fast queuing was used with a queue size of 100 packets (default value). All flows are equal and no reservation for real-time flows were made in this configuration.

DiffServ Configuration

In the DiffServ configuration two different sub-cases were taken into account: in the first one only the edge router (kemi-12) was DiffServ enabled by means of the TC tool [58], in the second one every router in the path between source and destination were DiffServ capable. We adopted the Dsmark queuing discipline (described in section 5.4.3) to implement our DiffServ routers. According to the generated workload, this queuing discipline differentiates traffic in four different classes: EF, AF1.1, AF2.3 and BE. A simple PRIO qdisc first separates the EF priority class from AFs and BE ones. The EF class is then handled with a token bucket filter (parameters EF_BURS='200Kb'; EF_MTU=1.5kb); a class-based queuing is used instead to differentiate AFs and BE traffic, and a red queuing management is employed for BE traffic and two gred qdiscs for the AFs classes respectively. The following figures describe the TC model of the queuing discipline in the DiffServ network node and the network configuration adopted in the following test cases.

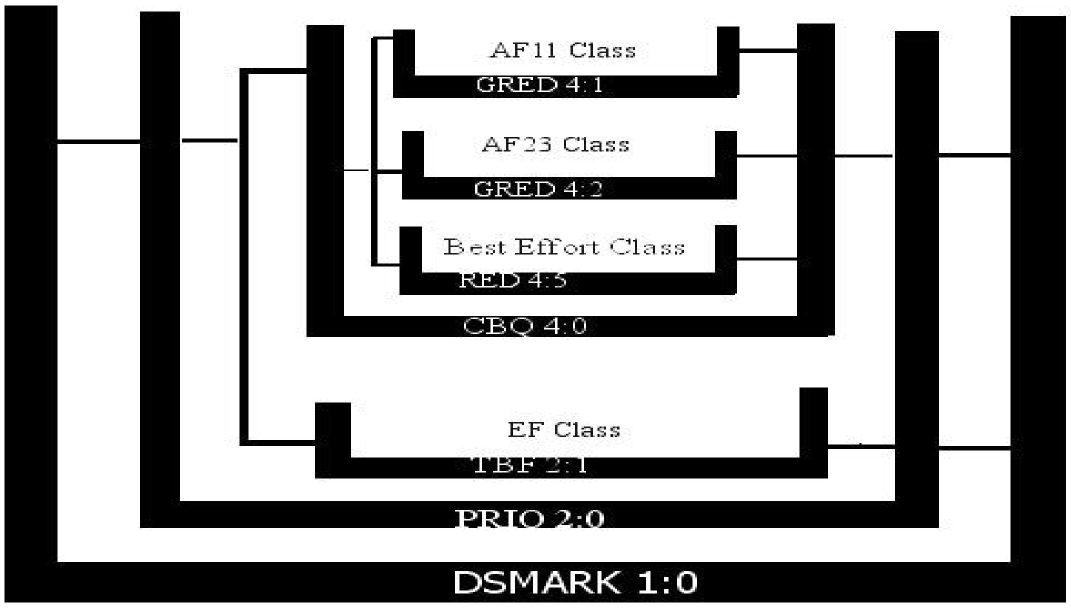


Figure 6.2 Queuing Discipline in the DiffServ Node

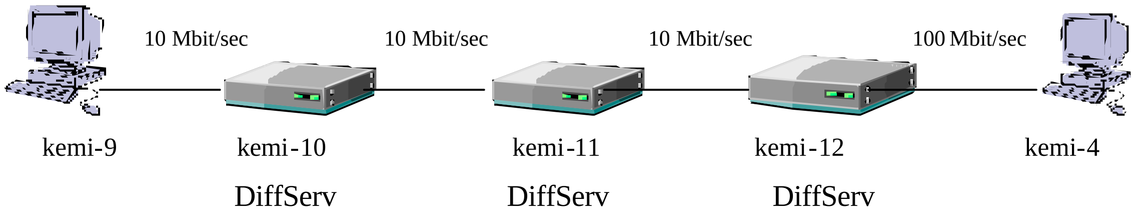


Figure 6.3 DiffServ Configuration

IntServ Configuration

In the IntServ configuration all machines are running RSVP daemon (release 4.2a4-3 of RSVP ISI daemon). A CBQ scheduling algorithm was enabled by means of the TC tool in each routers. CBQ isolates each flow and therefore builds a separate queue for each flow. Isolation of each flow (even best-effort ones) would cause considerable processing overhead particularly in the core network, then a good solution for this scalabity issue could be to isolate RSVP flows from the best-effort ones that will share the same queue. In fact we finally choose to isolate RSVP flows from the best-effort ones, which used one shared stochastic fair queue. Further the RSVP class includes split option in order to handle real-time packets which violated policing filters or exceeded reshaping buffers. The next figures describe the TC model of the queuing discipline in the IntServ network node and the network configuration adopted.

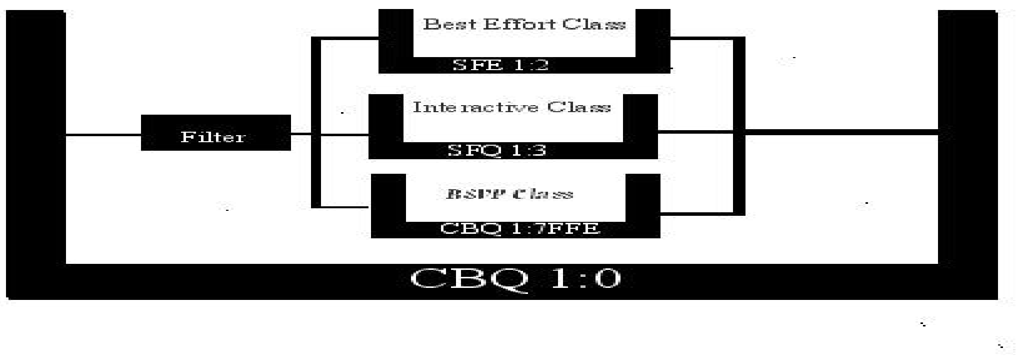


Figure 6.4 Queuing Discipline in the IntServ Node

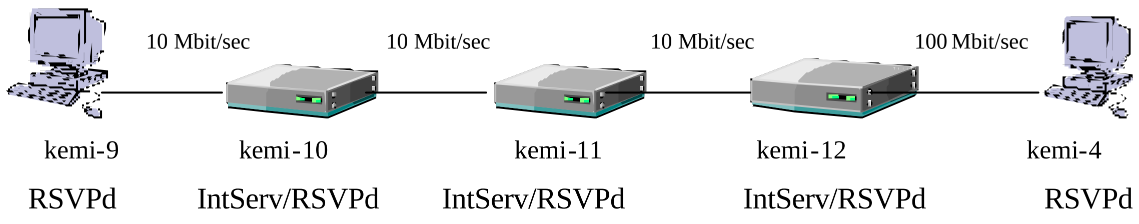


Figure 6.5 IntServ Configuration

IntServ/DiffServ Configuration

The network configuration set up for IntServ/DiffServ interoperability is as following:

- Kemi-9 and kemi-4 are configured as RSVP aware hosts, running a RSVP daemon, and in the following experiments they represented the sender and receiver of the traffic respectively in the network test bed.
- kemi-12 is configured as border router and is running a version of RSVP daemon in order to handle the IntServ/DiffServ interoperability. Its output interface is configured as the IntServ model establishes.
- Kemi-11 and Kemi-10 are configured as DiffServ core routers, so that they check the DS field of the incoming packets in order to retrieve the DSCP byte and assign them to the appropriate class.

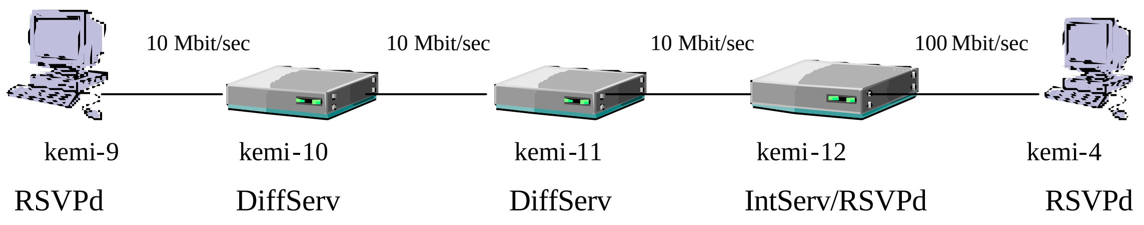


Figure 6.4 IntServ/DiffServ Configuration

6.4 Metrics

Fundamentally, the two most important telecommunication standard organizations have defined two different QoS parameters schemes: IETF IP Performance Metric (IPPM) and ITU-T proposal. The two entities essentially agree on the list of parameters that can be used to gauge the performance of an IP link and hence its quality, but they use similar or identical acronyms with different meanings. The most significant difference is that ITU-T proposes a statistical definition of QoS parameters, while IETF is willing to provide a precise measurements procedure for each parameter. The agreed parameters for IP Performance QoS are one-way delay, instantaneous packet delay variation (or Jitter), bandwidth and packet loss.

In our experiments we used a metrics system composed of the above-mentioned and other parameters (i.e. Throughput, Transfer Time and Packet Interarrival Time) in order to extrapolate the key parameters for each kind of traffic. We considered for example: bandwidth, delay and jitter for data stream traffic, RTT for interactive data and throughput or transfer time for data bulk traffic. But the two most important parameters for application that have stringent QoS demands are delay and Jitter, and for these we report a standard definition.

One-way delay is defined formally in RFC 2679. This metric is measured from the wire time of the packet arriving on the link observed by the sender to the wire time of the last bit of the packet observed by the receiver. The difference of these two values is the one-way delay.

Instantaneous Packet Delay Variation (IPDV) is formally defined by the IPPM working group draft. It is based on the one-way delay measurements and it is defined for a consecutive pair of packets. If we say D_i is the one-way delay of the I^{th} packet, then the IPDV (or jitter) of a packet pair is defined as $D_i - D_{i-1}$.

$$\text{IDPV\_jitter} = | D_i - D_{i-1} |$$

Results

The results were collected by monitoring one flow for each class of traffic, which travelled from kemi-4 to kemi-9, source and destination of all flows, respectively. The measurements for the wired part of the network consisted of three cases and four different combinations of QoS architectures. In each test case the achieved data rate, delay value and packet loss rate for each flow were measured by the Mcalc tool included in the MGEN package. The presented values are averages (rounded) of at least ten individual measurements, whose duration was 100 seconds. The following tables and figures show the results for the three test cases.

7.1 Test Case 1: Network lightly loaded

In this case the network was lightly loaded with a simple workload composed of 15 real-time UDP flows described in the test environment section, which start all together and end after 100 sec, at the same a TCP bulk transfer is active from the same source to the same destination. A more detailed flow description is reported in the next table.

Table 7.1 Workload description used in this test case

Data flows	Number	TOS byte
VoIP (adpcm) flows	5	EF (0xb8)
VoIP (gsm) flows	5	AF11 (0x28)
Video flows	5	AF23 (0x58)
BE/UDP flows	2	

As mentioned, MGEN, the workload generator used in our experiments, permits to set up the TOS value of the IP header in order to originate flows belonging to designated service classes. In this preliminary test case we studied only the comparison between BE and DiffServ network configuration and the following tables and figures show the results obtained as described above for four different kinds of flows analyzed: one TCP bulk transfer and three real-time streams. The first table illustrates the TCP bulk transfer results in the two network configurations compared.

Table 7.2 TCP bulk transfer results

Flow analyzed	**Network Configuration**	
TCP flow	**BE**	**DiffServ**
Average Transfer time (sec)	40,82	54,72
Average Throughput (kB/sec)	501,71	374,27

Regarding the TCP transfer bulk, the DiffServ configuration did not introduce any improvement in the transfer time because best-effort traffic was limited by the priority queuing discipline adopted by the DiffServ routers. Best-Effort configuration without any traffic differentiation guarantees a smaller transfer time for a best effort flow like a TCP transfer bulk because it did not suffer of any bandwidth restriction. Next tables present the results for the real-time traffic.

Table 7.3 VoIP (adpecm) flow results

VoIP (adpcm) Flow	**BE**	**DiffServ (EF)**
Recv date rate (kbit/s)	41,6	41,56
Packets Received	5000	4994
Packets Dropped	0	6
Ave. Delay (ms)	6	4
Max. Delay (ms)	42	136
Min. Delay (ms)	1	8

Table 7.4 VoIP (gsm) flow results

VoIP (gsm) Flow	**BE**	**DiffServ (AF11)**
Recv date rate (kbit/s)	14,4	14,4
Packets Received	5000	5000
Packets Dropped	0	0
Ave. Delay (ms)	6	4
Max. Delay (ms)	42	137
Min. Delay (ms)	1	1

Table 7.5 Video (200 kbit/s) VBR flow results

Video Flow (VBR)	BE	DiffServ (AF23)
Recv date rate (kbit/s)	202,45	201,25
Packets Received	8435	8385
Packets Dropped	0	0
Ave. Delay (ms)	5	3
Max. Delay (ms)	42	135
Min. Delay (ms)	4	0

In this test case we analyzed in fact three kinds of UDP flows, whose packets were marked as EF, AF11 and AF23 respectively, representing a VoIP (adpcm codec) phone call, a VoIP (gsm codec) phone call and a 200 kbit/sec (VBR) medium-high quality video stream. As previous tables show, when the network is lightly loaded the two mechanisms provided almost equal services in terms of throughput and packet loss for each real-time flow, with a light improvement for the average delay and a worse performance for the maximum delay in the DiffServ network. The delay values obtained from the mcalc tool are calculated from application to application, thus including not only network delay but also processing delay caused by the end hosts, nevertheless the absolute values are not important here, because we are interested in the relative changes due to different network configurations and workloads.

Some high spurious values for the max Delay parameter in the DiffServ configuration can be due to the Linux Operating System process scheduling mechanism that delays some packets because of processing overhead or to high delay function granularity around 10 ms in our version.

The following figures show the delay and delay distribution function of a VoIP (adpcm) flow whose packets were marked as EF service class in the two network configurations. These figures represent the best case between ten individual replications for both configurations. A detailed analysis of time parameters, especially the delay distribution function will better explain the apparent misbehavior regarding the maximum delay registered in the DiffServ network, confirming that only few packets suffered of this high delay values in some replications, but as expected the DiffServ management of the EF flow analyzed offers lower delay and jitter values generally.

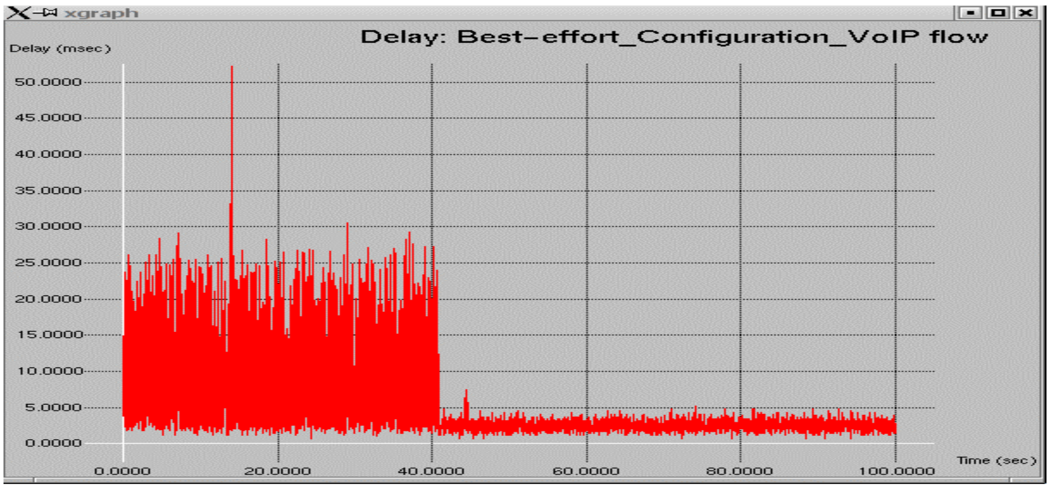


Figure 7.1 Delay of a VoIP (adpcm) Flow in the Best-Effort Configuration

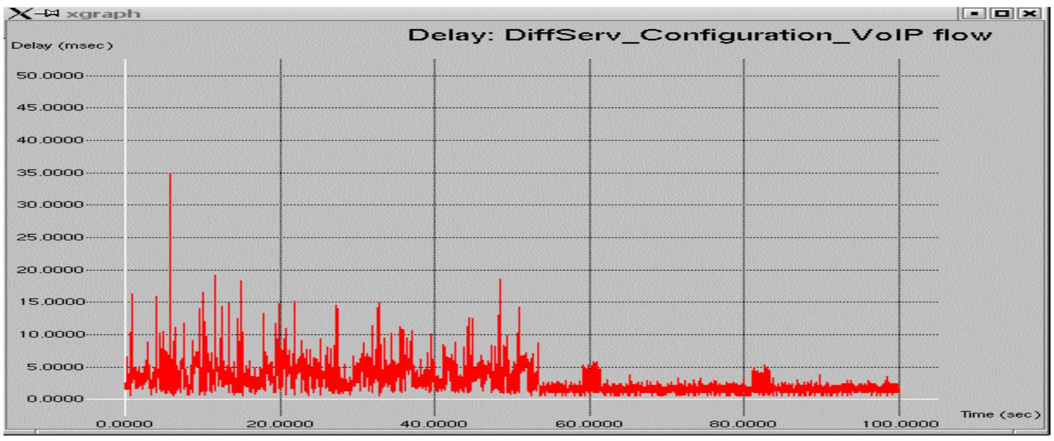


Figure 7.2 Delay of a VoIP (adpcm) Flow in the DiffServ Configuration

In the previous figures we can individuate two phases in each experiment. Until about 40 sec (duration of TCP transfer bulk, see table 7.2) in the best-effort configuration the delay of real-time flow is really variable presenting an elevate jitter. The mean value and average deviation for the all experiment duration are: 5,63 ms and 4,53 ms respectively. Considering that we are not interested in absolute values in this analysis a jitter so high compared to the delay could represent a serious problem for a jitter intolerant application.

We can notice a similar shape for the delay in the DiffServ configuration. Until approximately 54 sec, in fact, it suffers of a jitter phenomenon but drastically reduced in respect to the previous case, as the mean value (2,85 ms) and average delay variation (1,49 ms) demonstrate. The second phase of the two graphs show instead that in an over-provisioned case the FIFO queuing offers a satisfactory service level to this flow almost comparable with the DiffServ management because of its faster scheduling mechanism.

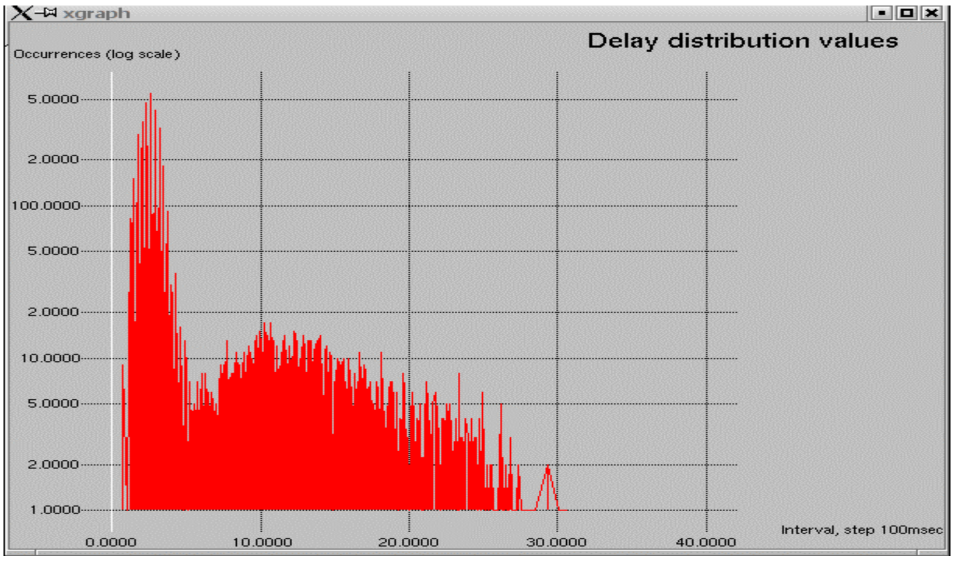


Figure 7.3 Delay Distribution of a VoIP (adpcm) Flow in a Best-effort network

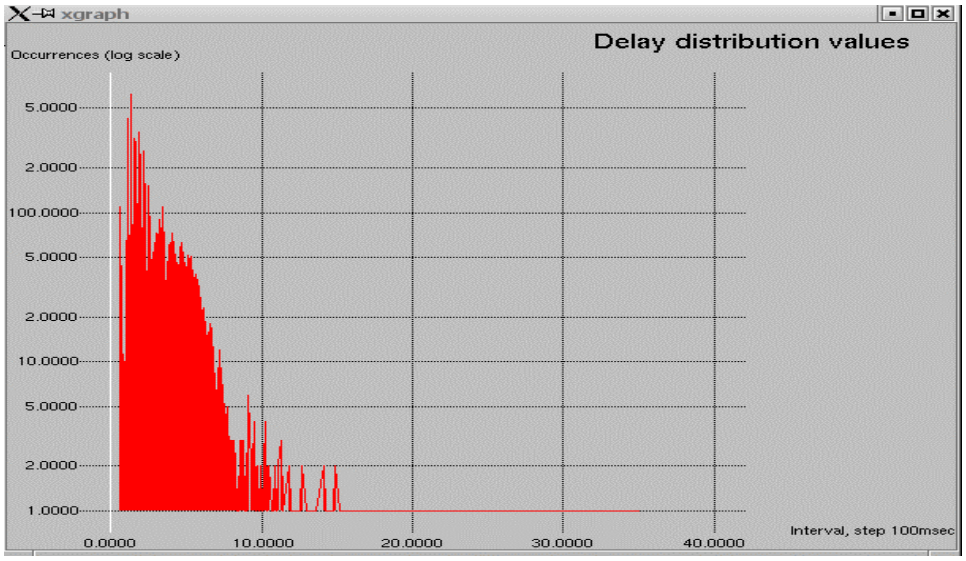


Figure 7.4 Delay Distribution of a VoIP (adpcm) Flow in a DiffServ enabled network

The delay distribution function (figures 7.3 and 7.4) better explains the previous figures and tables, confirming what we said before. In the best effort configuration a peak value of 500.000 occurrences is around 3 ms (values deriving from the last part of the delay shape) but delay values ranging from 2000 to 10000 occurrences are spread until 30 ms. In DiffServ configuration all delay values are concentrated until 15 ms confirming that the Token Bucket Filter management of an EF flow guarantees a lower jitter than an undifferentiated best-effort handling of packets belonging to this real-time flow at least when the TCP transfer bulk is active. Finally DiffServ management offers a superior level of service for real-time application also in a lightly loaded network, particularly in the presence of a TCP bulk transfer.

7.2 Test Case 2: Network overloaded

In this case the network was loaded whit a more complex workload composed of 35 UDP flows described in the test environment section, which were active for 100 sec and as in the previous test case a TCP transfer bulk was active from the same source to same destination at the same time. We compared four different network configurations: Best-effort, DiffServ/Best-effort, DiffServ and IntServ. The best effort configuration represents the reference for our comparison, in the DiffServ/BE network only the edge router (Kemi-12) was DiffServ enabled and the two other cases were described in the network configuration section. For more clearness we report UDP flow characteristics in the following table showing TOS values and Tspec (RESV messages) parameters used in DiffServ and IntServ configuration respectively.

Table7.6 Workload description used in this test case

Data flows	Number	TOS byte	CL Tspec
VoIP (adpcm) flows	5	EF (0xb8)	[CL1 5200 256 5200 64 128]
VoIP (gsm) flows	5	AF11 (0x28)	[CL2 1800 256 1800 64 128]
Video flows	5	AF23 (0x58)	[CL3 25000 256 25000 64 128]
BE/UDP flows	20		

The following four tables show the comparative results for one flow belonging to each service class, but before we should describe the tests methodology used in this case in more detail. Particularly, we need to describe how we generated RSVP flows. As mentioned before, MGEN permits to originate flows observing the RSVP specification model. In our experiments we generated for each RSVP flow a PATH message at t=0 sec from the source and a relative RESV message at t=5 sec from the destination, at t=10 sec flows started sending data and finally at t=110 sec flows end. PATH and RESV messages were sent every 30sec as the IntServ/RSVP model establishes. The first parameter analyzed was again the TCP bulk transfer time.

Table 7.7 TCP transfer bulk results

Flow analyzed	Network Configuration			
TCP flow	**BE**	**DiffServ / BE**	**DiffServ**	**IntServ**
Average Transfer time (sec)	139,36	161,76	168,84	124,94
Average Throughput (kB/sec)	146,96	126,6	121,13	164,42

The previous table shows the average transfer time and relative throughput of a TCP flow in each QoS configuration. As expected in the Best-effort one the TCP flow gets a good result because it did not suffer any bandwidth restriction. Between the QoS enabled, IntServ with its CBQ mechanism coupled with a stochastic fair queuing management shows a better performance for this flow than the PQ coupled with a red management offered by the DiffServ network to the best-effort class, even superior of the Best-Effort handling

Table 7.8 VoIP (adpcm) Flow results

VoIP (adpcm) Flow	BE	DiffServ / BE	DiffServ (EF)	IntServ(CL1)
Recv date rate (kbit/s)	18,6	41,6	41,6	41,57
Packets Received	2236	5000	5000	4995
Packets Dropped	2764	0	0	5
Ave. Delay (ms)	45	2	4	9
Max. Delay (ms)	75	8	10	67
Min. Delay (ms)	5	1	1	1
Delay variation (ms)	70	7	9	66

Table 7.9 VoIP (gsm) Flow results

VoIP (gsm) Flow	BE	DiffServ / BE	DiffServ(AF11)	IntServ(CL2)
Recv date rate (kbit/s)	6,09	14,4	14,4	4995
Packets Received	2114	5000	5000	41,57
Packets Dropped	2886	0	0	5
Ave. Delay (ms)	45	2	4	10
Max. Delay (ms)	75	9	11	65
Min. Delay (ms)	5	1	1	1
Delay variation (ms)	70	8	10	63

Table 7.10 Video Flow results

Video Flow (VBR)	BE	DiffServ / BE	DiffServ(AF23)	IntServ(CL3)
Recv date rate (kbit/s)	117,1	201,4	202,1	201,55
Packets Received	4881	8390	8420	8398
Packets Dropped	3481	0	0	2
Ave. Delay (ms)	43	2	4	9
Max. Delay (ms)	74	10	15	58
Min. Delay (ms)	5	1	1	1
Delay variation (ms)	69	9	14	56

The previous tables present the performance results for three different real-time flows, when the network was overloaded with a high percentage of real-time traffic. First of all we noticed that the pure best-effort network could not provide sufficient service for real-time traffic anymore.

Significant improvements are instead shown from the two different DiffServ configurations and also the IntServ. Every parameter gets an advantage from these QoS network configurations. The two CBR flows, emulating IP phone calls, get the target data rate without any losses or only a few packets lost in the IntServ network, with an average improvement of about 40% in terms of throughput and also the time parameters registered a considerable improvements reducing delay and delay variation even twenty times in respect to the Best-effort configuration, but it is clear that it is secondary matter the delay analysis when flows register a so high number of packet loss. The VBR video stream shows similar results for each parameter.

A light improvement is introduced instead enabling DiffServ treatment only at the edge router (kemi-12 in our network topology). Making a correct traffic differentiation and traffic shaping in the most critical link (i.e. access link) guarantees a service level even higher than a traffic reshaping at every hop between source and destination, because packet reordering in the core network introduces additive delay components, which are useless when the traffic was already shaped at the edge of the network. This enhancement regards especially the maximum delay and delay variation, as we will show in the next figures.

The IntServ enabled network also provides an important improvement in terms of service offered to real-time flows that made a reservation, guaranteeing the target data rate for each reserved stream and smaller delay and delay variation compared to the BE network. Nevertheless the high value for delay and especially delay variation shown by the IntServ network confirm that the CBQ mechanism adopted for this kind of traffic does not represent an ideal solution in a congested network. Instead the RED and GRED management inside a priority queuing of the DiffServ network offers an excellent solution for a real-time application in such a situation of moderate congestion and variable packet size regime.

A more detailed delay analysis is showed in the following figures, which analyzes first the packet interarrival time of a selected VoIP (adpcm codec) flow marked as EF in the DiffServ network, whose packets were dumped by the Tcpdump tool and then post-processed in order to obtain the time parameter characterizing the flow analyzed. After that we proposed a Time analysis of a VoIP (gsm codec) flow marked as the AF11 service class in the DiffServ network. In this case graphs were obtained by the Mcalc log file. The two test methodologies provided coherent results validating each other.

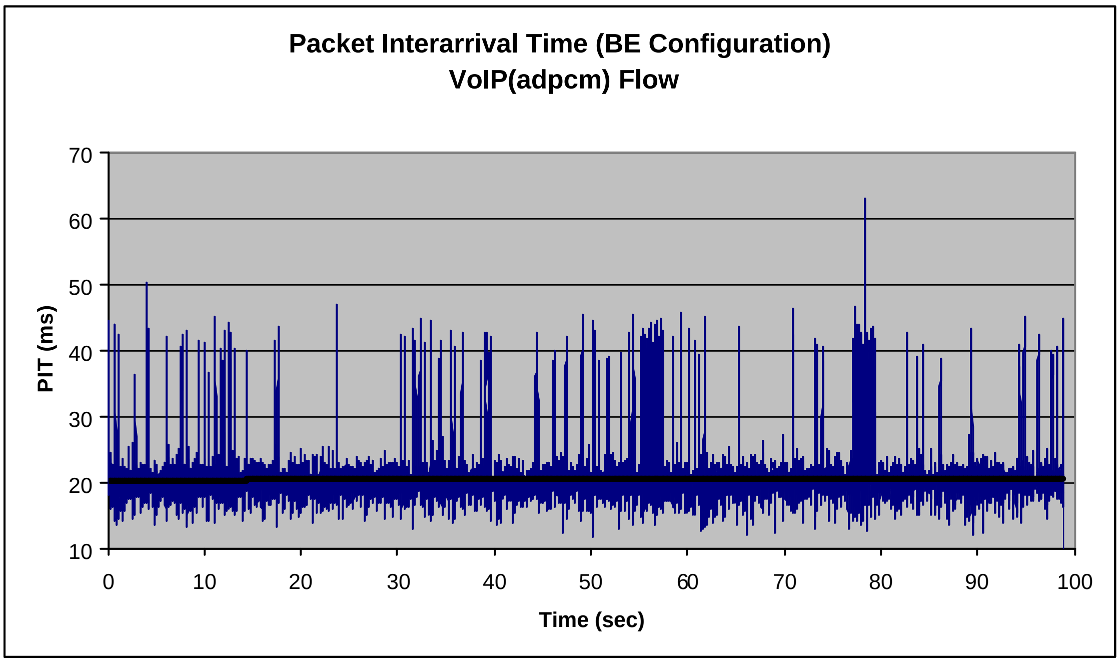


Figure 7.5 Packet Interarrival Time (Best-effort Configuration) of a VoIP (adpcm) Flow

Mean value = 20,59737 (ms) Average deviation = 2,236817 (ms)

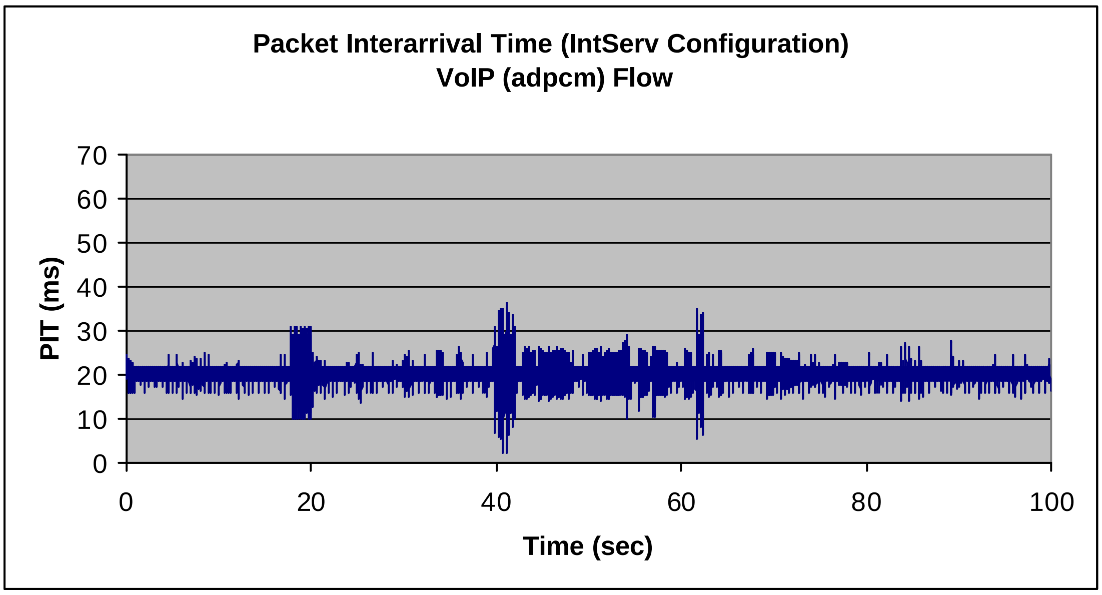


Figure 7.6 Packet Interarrival Time (IntServ Configuration) of a VoIP (adpcm) Flow

Mean value = 20,00154 (ms) Average deviation = 1,672384 (ms)

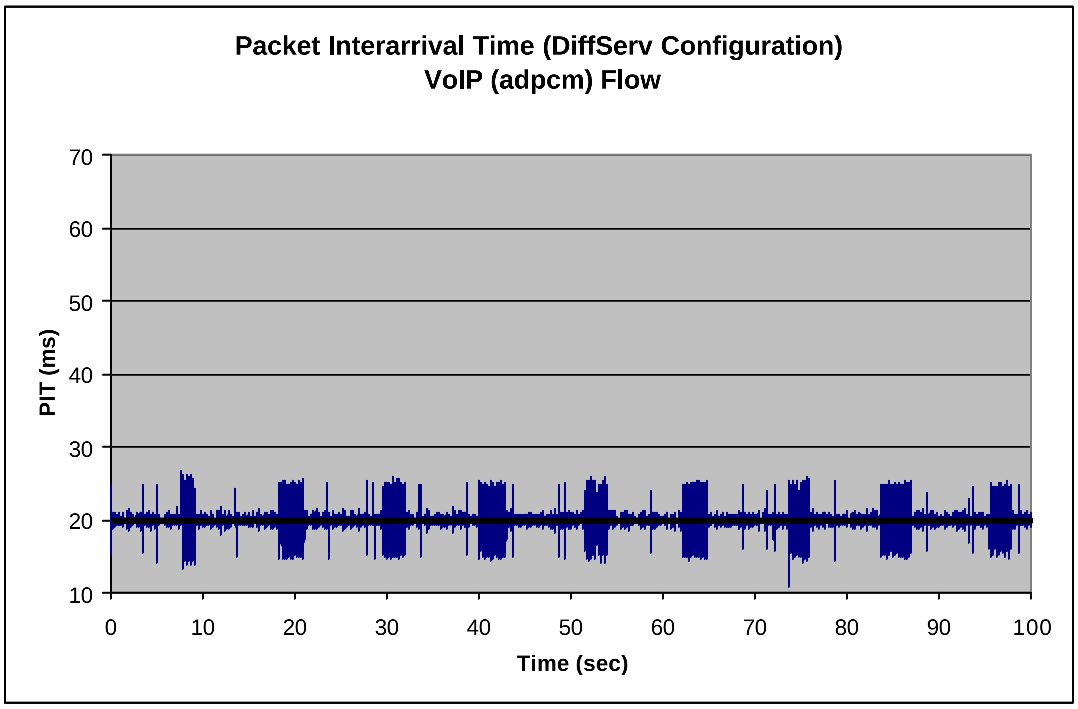


Figure 7.7 Packet Interarrival Time (DiffServ configuration) of a VoIP (adpcm) Flow

Mean value = 20,00019 (ms) Average deviation = 0,850946 (ms)

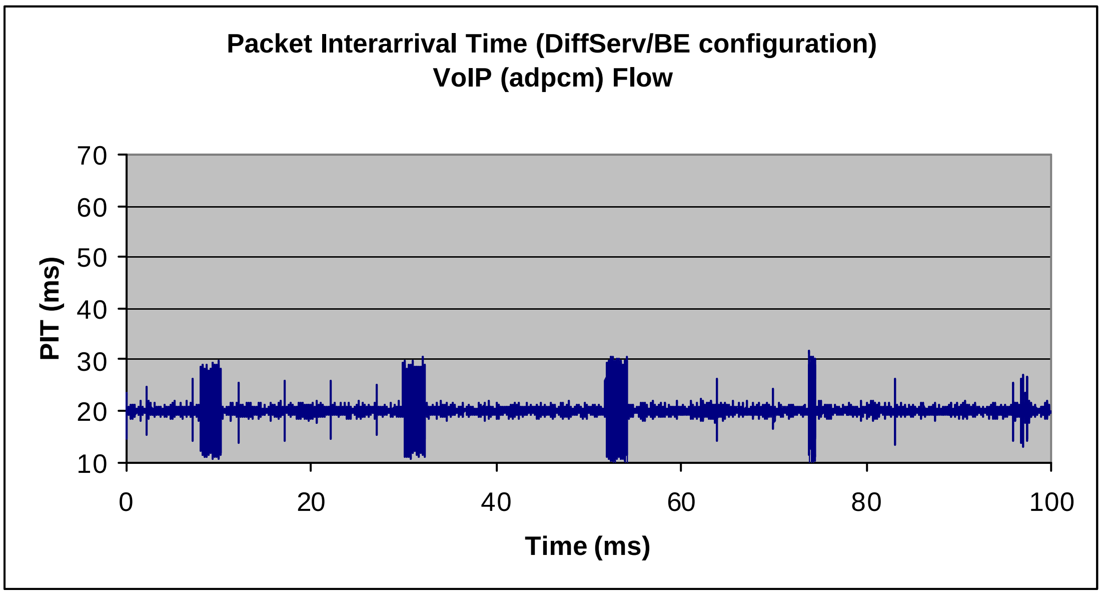


Figure 7.8 Packet Interarrival Time (DiffServ/BE Configuration) of a VoIP (adpcm) Flow

Mean value = 19,99 (ms) Average Deviation = 0,840634 (ms)

The previous figures show the instantaneous packet interarrival time for a selected adpcm VoIP flow in each network configuration. This parameter is directly related to the network delay and it should coincide with the inter packet transmission time in a lightly loaded network. In this case the sending rate for the flow analyzed was 50 packet/sec (table 6.2), which corresponds to a 20 ms inter-packet transmission time. As expected the mean value and average deviation of the packet interarrival time shows an increasing improvement in the QoS enabled configurations. In each configuration the mean value is around 20 ms, but the best-effort network shows an average deviation 40% worse than the IntServ network and almost three times larger than DiffServ networks. The best performance was obtained in the DiffServ/BE one that guarantees the smallest mean value and average deviation for the instantaneous packet interarrival time and in general a more flat shape reducing the number of bursts of packets delayed thanks to the token bucket filter handling of EF packets.

The next figures present a Time analysis (Delay and Delay distribution) for a gsm IP phone call, we examined a selected flow making comparison between four different configurations. Results for this flow are even more significant in respect to the previous ones. These figures represent the best case between ten individual replications for each configuration. We should remember that the delay function examined came from the log file provided by Mcalc, and it represents the sum of two components: network delay and application delay.

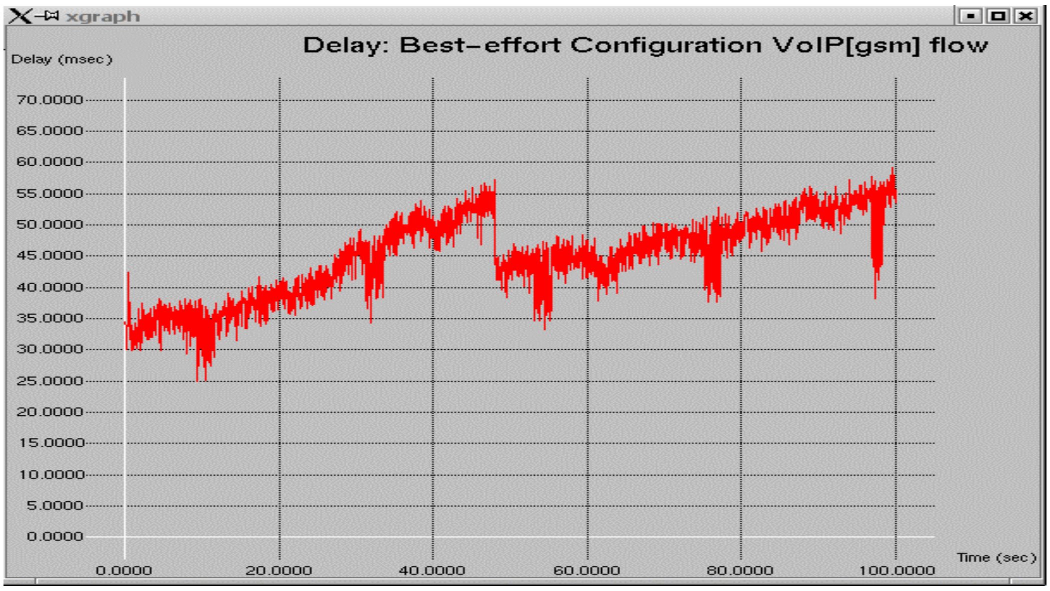


Figure 7.9 Delay of a VoIP (gsm) Flow in a loaded Best-Effort Network

Figure 7.9 represents the delay function for the selected gsm flow in a Best-Effort network, the delay was never under 25 ms and it grows until about 55ms exhibiting a periodic shape, increasing because of the increasing queue length until packets get lost. The mean value and average delay variation are 44,91 ms and 5,51 ms respectively, at least ten times higher than values for QoS enabled networks.

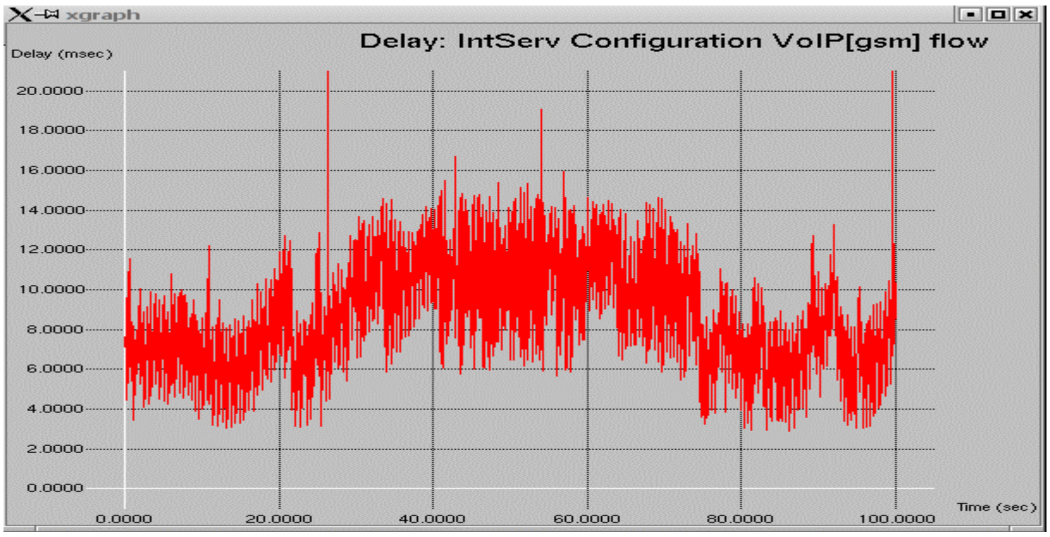


Figure 7.10 Delay of a VoIP (gsm) Flow in the IntServ Network Configuration

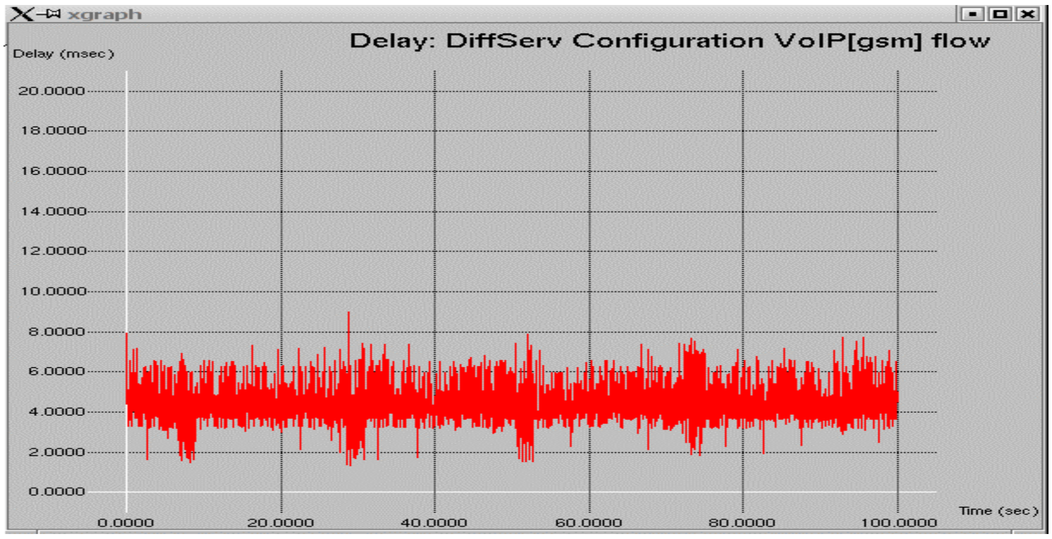


Figure 7.11 Delay of a VoIP (gsm) Flow in the DiffServ Network Configuration

The two previous figures represent the delay function for the same flow in the two most well known QoS network approaches. As already outlined when commenting flow characteristics tables, the DiffServ approach exhibits more flatness and a smaller mean value for the delay, 3,76 ms for the DiffServ network and 8,45 ms for the IntServ one respectively. Also the average delay variation is smaller in the DiffServ network: 0,31 ms against 2,14 ms.

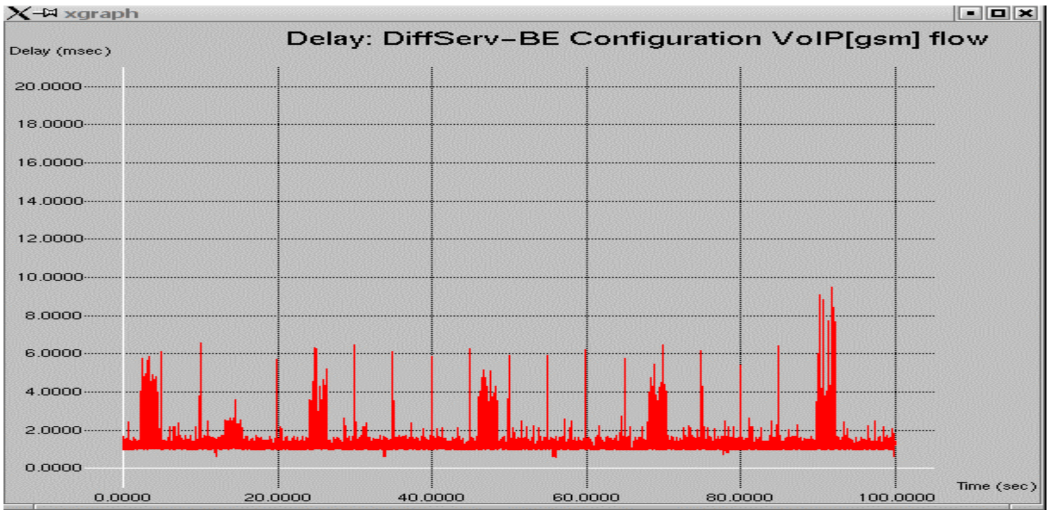


Figure 7.12 Delay of a VoIP (gsm) Flow in the DiffServ/Best-Effort Network Configuration

According to what we obtained in the previous summarizing tables, the combination of a DiffServ router at the edge of the network and best-effort handling of marked packets in the core routers shows the best results at least in a not heavily congested network. The mean value and average delay variation in this case are: 1,19 ms and 0,27 ms.

The following figures illustrate the delay distribution function of the flow examined in each QoS network architectures analyzed. Particular attention should be paid to the scale ranges; the first one represents the delay distribution in the Best-Effort network: the maximum number of occurrences is concentrated around the mean value and delay values range approximately from 30 to 60 ms.

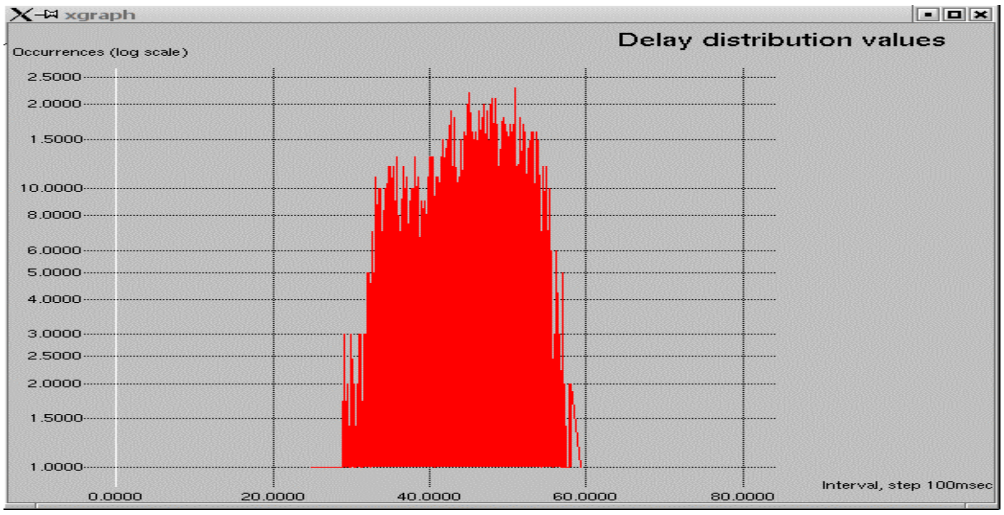


Figure 7.13 Delay Distribution of a VoIP (gsm) Flow in a Best-effort network

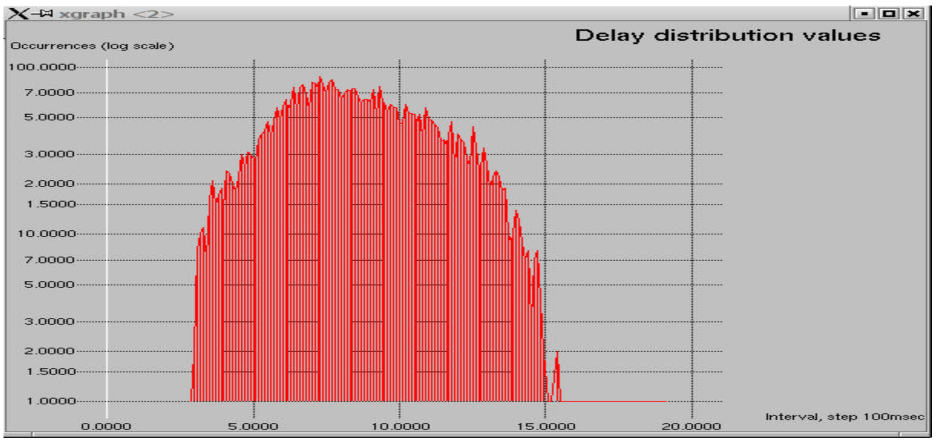


Figure 7.14 Delay Distribution of a VoIP (gsm) Flow in the IntServ Network Configuration

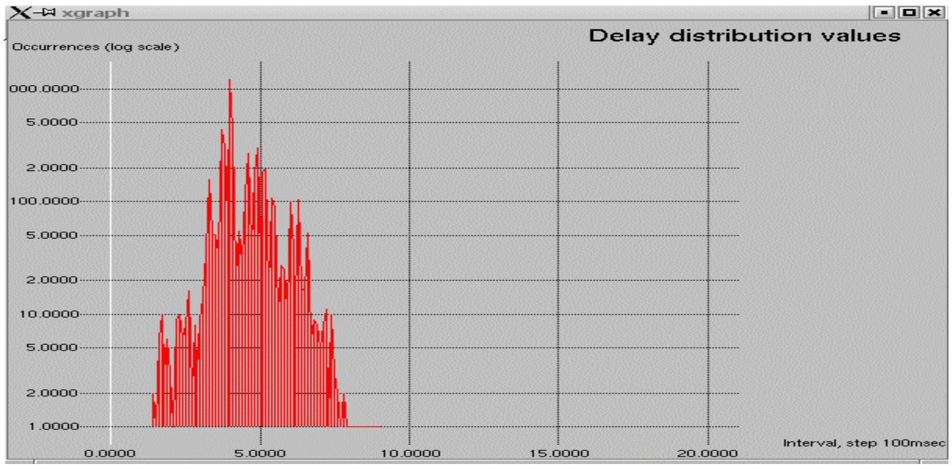


Figure 7.15 Delay Distribution of a VoIP (gsm) Flow in the DiffServ Network Configuration

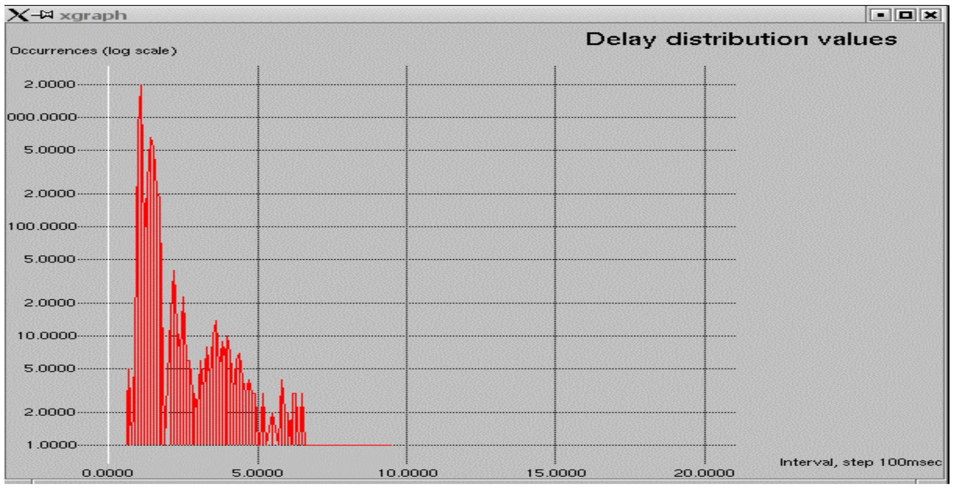


Figure 7.16 Delay Distribution of a VoIP (gsm) Flow in the DiffServ/Best-Effort Network Configuration

The previous three figures show the delay distribution value for the same flow in the three different network QoS architectures, and according to our earlier explanation, the combination of DiffServ and Best-Effort represents the best solution for this flow in this condition of network loads. Most delay sample values in the last figure are in fact concentrated under 5 ms, exhibiting a very low jitter.

7.3 Test Case 3: Network heavily overloaded

The third test case represents the most complex and critical one. We have created in fact a heavy congestion situation in the core network, in order to observe network behavior for the proposed QoS architecture in this extreme situation. The workload was again composed of 35 UDP flows described in the test environment section, which were active for 100 sec, and as in the previous test case a TCP transfer bulk was active from the same source to the same destination. But in this case 30 of these UDP flows were prioritized setting up the TOS byte in the experiments in DiffServ configuration and 20 flows made a reservation for the IntServ network. The following table illustrates in more detail the flow parameters used in our experiments.

Table 7.11 Workload description used in this test case

Data flows	Number	TOS byte	CL Tspec
VoIP (adpcm) flows (41,6Kbit/s)	5	EF (0xb8)	[CL1 5200 256 5200 64 128]
VoIP (gsm) flows (14,4 Kbit/s)	5	AF11 (0x28)	[CL2 1800 256 1800 64 128]
Video flows VBR (200 Kbit/s)	5	AF23 (0x58)	[CL3 25000 256 25000 64 128]
UDP (600Kbit/s) VBR flows	1	EF (0xb8)	[CL4 75000 256 75000 64 128]
UDP (600Kbit/s) VBR flows	4	AF11 (0x28)	[CL4 75000 256 75000 64 128]
UDP (600Kbit/s) VBR flows	6	AF11 (0x28)	
UDP (600Kbit/s) VBR flows	4	AF23 (0x58)	

The test methodology of this test was similar to the previous one, at least regarding the experiments in the IntServ configuration. Particular attention should be paid to the last experiments regarding the combined QoS network architecture, where real-time flows made a reservation according to the RSVP model and were also marked according to the DiffServ model in order to get advantage from the combination of the two network QoS approaches. The following tables show a comparative analysis for each kind of flow in the three network QoS architectures analyzed: DiffServ, IntServ and our proposal of an IntServ/DiffServ inter-operation model.

Table 7.12 Overall Throughput

Number of Flows Analyzed = 30	DiffServ	IntServ	Int/Diff
Overall Throughput (Kbit/sec)	8039	6235	6597

The first parameter we took into account was the overall throughput on the receive side, and as the previous table shows the DiffServ configuration with its priority queuing mechanism offers the best performance, because it did not suffer from any scalability problem and guaranteed a faster packet scheduling avoiding buffer overflow. The DiffServ QoS mechanism then guarantees a better utilization of the link bandwidth close to the maximum available bandwidth considering that the other five UDP flows and a TCP bulk transfer are not included in this parameter. The IntServ mechanism shows the worst performance, because the increased number of reservations raises a scalability trouble and the overall efficiency of the network degrades significantly; but the most interesting result is that concerned with the combination of the two network QoS mechanisms that shows a significant improvement for this parameter in respect to the IntServ configuration and as we will demonstrate in the next tables take advantages from the reservation style proper of the RSVP mechanism. Next tables show the results for each kind of flow analyzed.

Table 7.13 TCP transfer bulk results

Flow analyzed	**Network Configuration**		
TCP flow	**DiffServ**	**IntServ**	**Int/Diff**
Average Transfer time (sec)	192	165	162
Average Throughput (kB/sec)	106,6	124,12	126,64

The second parameter analyzed was the TCP bulk transfer time. Also for this parameter the combination of the two mechanisms guarantees the optimal solution. Confirming what we have demonstrated in the test case 2, the stochastic fair queue management inside the CBQ mechanism offered in the IntServ router (kemi-12) to this best-effort flow, avoiding the resources starvation of lowest priority queue typical of PQ shows even better performance when coupled with the DiffServ management in the core network. The following tables exhibit in more detail the network service offered to the real-time flows in each network QoS mechanism.

Table 7.14 VoIP (adpcm codec) Flow results

VoIP (adpcm) Flow	DiffServ (EF)	IntServ (CL1)	Int/Diff
Recv date rate (kbit/s)	41,49	41,6	41,57
Packets Received	4986	5000	4997
Packets Dropped	14	0	3
Ave. Delay (ms)	13	12	16
Max. Delay (ms)	18	33	46
Min. Delay (ms)	6	2	4
Delay variation (ms)	12	30	42

Analyzing these results we should notice that, according to a real case, a major level of congestion was created for the lower DiffServ classes of service. In fact as, we will show later, only the AFs classes suffer a significant degradation in terms of service level from the DiffServ management.

For the EF-CL1 flow (table 7.14), representing an IP phone call in adpcm codec, all three QoS mechanisms offer almost equal service at least in terms of throughput in fact only few packets get lost in the DiffServ network; but as we will show later in the delay analysis, the token bucket filter handling in the DiffServ configuration offers better results for time parameters than the IntServ management of this flow. A behavior close to the IntServ one is shown in the combination of IntServ and DiffServ, with a slightly worse mean value and average deviation of delay with respect to the IntServ case. This behavior is due to the fact that EF flows did not get any privilege in the first link (the more congested) in respect to the other reserved flows, but the level of service offered by the combination is still reasonable for a real-time application as we will show in the time analysis of this flow.

Table 7.15 VoIP (gsm codec) Flow results

VoIP (gsm) Flow	DiffServ (AF11)	IntServ (CL2)	Int/Diff
Recv date rate (kbit/s)	11,75	14,4	14,31
Packets Received	4082	5000	4968
Packets Dropped	918	0	32
Ave. Delay (ms)	82	13	27
Max. Delay (ms)	97	33	84
Min. Delay (ms)	7	3	4
Delay variation (ms)	90	30	42

The second flow analyzed was a VoIP (gsm) flow (table7.15), and the results for this flow better exhibit that the DiffServ mechanism is not sufficient anymore for guaranteeing bandwidth requirements for a CBR flow (about 20% of packets dropped) when the rate of packets marked with the same TOS byte (belonging then to the same class of traffic) overcomes the bandwidth available for this class. In this case in fact packets exceeding the available bandwidth partially (about 30%) go to the best-effort class sharing bandwidth from a class with lower priority and remaining packets get lost in the AF class. The DiffServ architecture then presents its limit tied with its class full approach that does not permit to differentiate between flows belonging to the same class. The IntServ configuration shows the best results for this flow, but the combined solution, too, guarantees bandwidth requirements and sufficient response in terms of the time parameters.

Table 7.16 Video Flow results

Video Flow (VBR)	**DiffServ (AF2.3)**	**IntServ (CL 3)**	**Int/Diff**
Recv date rate (kbit/s)	170,62	201,6	201,57
Packets Received	7115	8400	8389
Packets Dropped	1216	0	1
Ave. Delay (ms)	118	11	19
Max. Delay (ms)	142	33	42
Min. Delay (ms)	7	3	4
Delay variation (ms)	135	30	38

As expected, also the AF23 (200Kbit/sec mean data rate) flow (table 7.16) shows a behavior similar to the previous flow analyzed, manifesting again that DiffServ QoS management does not offer any guarantee in such a congestion situation. For this flow with a bigger packet size and higher data rate, the combination of the two mechanisms offers a quite satisfactory level of service close to the service offered by the IntServ architecture.

The last two tables show the results for two VBR streams of 600 Kbit/sec, respectively marked as EF and AF11 classes of service in the DiffServ tests. Also for these streams the Int/Diff configuration exhibits an acceptable level of service guaranteeing a satisfactory receive data rate for the EF one and a better performance in terms of average delay and delay variation at least for the last one in respect to the IntServ configuration.

Table 7.17 Flow 16 results

Flow 600 Kbit/s VBR	DiffServ (EF)	IntServ (CL4)	Int/Diff
Recv date rate (kbit/s)	594,99	597,9	601,075
Packets Received	24793	24914	25045
Packets Dropped	159	1	18
Ave. Delay (ms)	13	12	16
Max. Delay (ms)	18	34	36
Min. Delay (ms)	6	3	5
Delay variation (ms)	13	31	32

The network response to this high quality video stream in the three QoS architectures is almost similar to the first UDP real-time flow analyzed (see table 7.14). Showing that network handling of EF flows is quite independent from flow packet size and data rate at least in a not over-congested situation for this class.

Table 7.18 Flow 21 results

Flow 600 Kbit/s VBR	DiffServ (AF11)	IntServ (CL4)	Int/Diff
Recv date rate (kbit/s)	427,86	242,31	276,06
Packets Received	17840	10099	11509
Packets Dropped	7191	15026	13469
Ave. Delay (ms)	85	90	96
Max. Delay (ms)	99	217	195
Min. Delay (ms)	8	6	12
Delay variation (ms)	91	211	183

The last flow analyzed is the first one that did not make any reservation. We can easily observe how DiffServ management still offers some level of service to this flow in such a congestion situation, and how the combination of the two QoS mechanisms guarantees an improvement around 10-15% in terms of throughput in respect to the IntServ architecture, demonstrating more efficient link utilization. The following figures present a time analysis (Delay and Delay distribution) for an adpcm IP phone call and a gsm VoIP flow. We examined two selected flows making comparisons between the three different QoS architectures. These figures represent the best case between ten individual replications for each configuration.

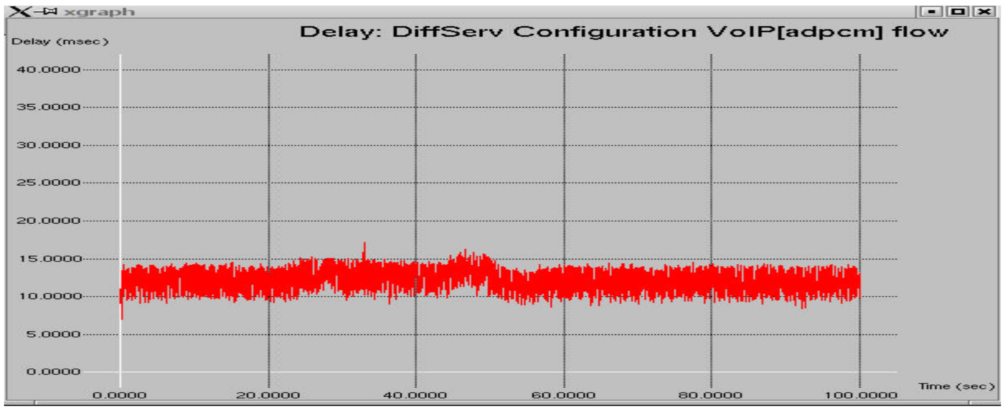


Figure 7.17 Delay of a VoIP (adpcm) Flow in the DiffServ Network Configuration

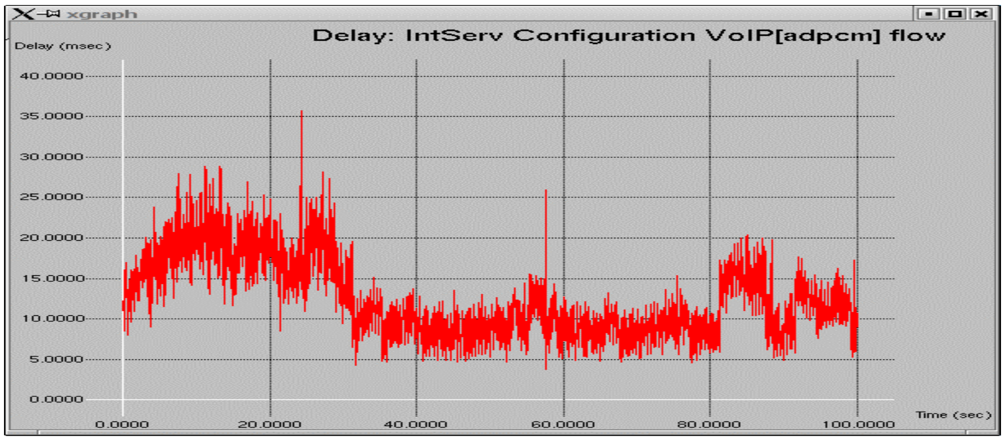


Figure 7.18 Delay of a VoIP (adpcm) Flow in the IntServ Network Configuration

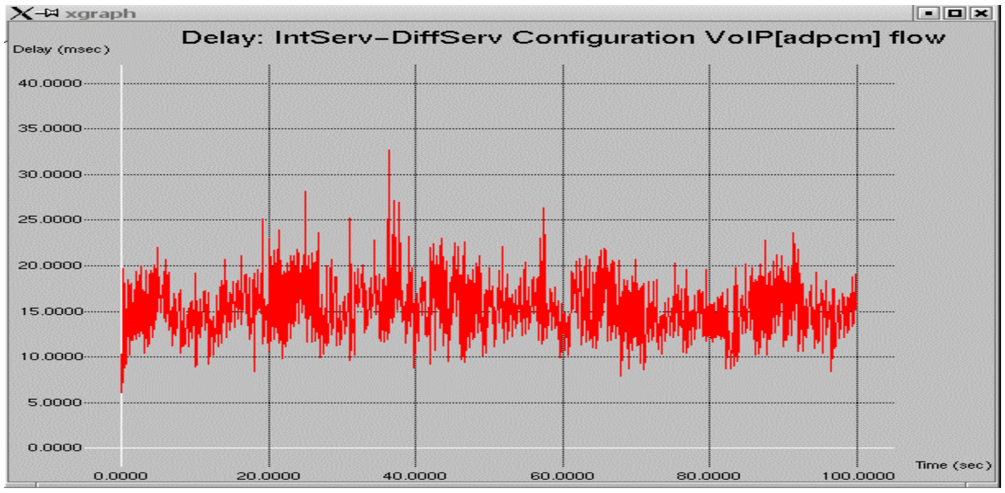


Figure 7.19 Delay of a VoIP (adpcm) Flow in the IntServ-DiffServ Network Configuration

The figures on the previous page illustrate the delay function of an adpcm VoIP flow in each network configuration. As already outlined when commenting single flow parameters, the DiffServ architecture offers the best response to this EF flow in terms of delay and jitter with quite a flat delay shape ranging from 10 to 15 ms, mean value 12,14 ms and average delay variation 1,19 ms. We remind the reader that these delay values include application delay and not only network delay, but they are still interesting when compared to the values obtained in the same manner for the other configurations.

A more irregular shape presents the delay function in the IntServ configuration, oscillating between 5 and 35 ms with a mean value and average delay variation of 12,49 and 3,9 ms respectively. The last graph represents the delay function for the same flow in our IntServ/DiffServ cooperation proposal, it clearly appears as a combination of the two previous ones, taking advantage in terms of jitter (1,96 ms) from the DiffServ architecture and keeping the mean value reasonably low (14,83 ms), representing a very good solution for a real-time application intolerant to the jitter.

The next three pictures represent the delay distribution function for the same flow in the three QoS mechanisms compared. The reader should pay attention to the logarithmic scale of the y-axis. Confirming what we found in the table regarding this flow and in the previous figures, delay distribution in the DiffServ network is mostly concentrated around its mean value, with a very small jitter typical of its scheduling mechanism: TBF.

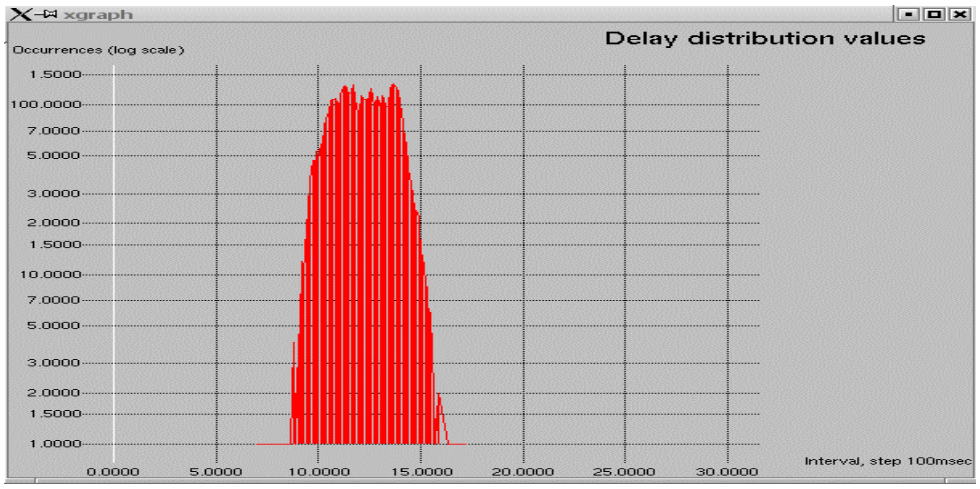


Figure 7.20 Delay Distribution of a VoIP (adpcm) Flow in the DiffServ Network Configuration

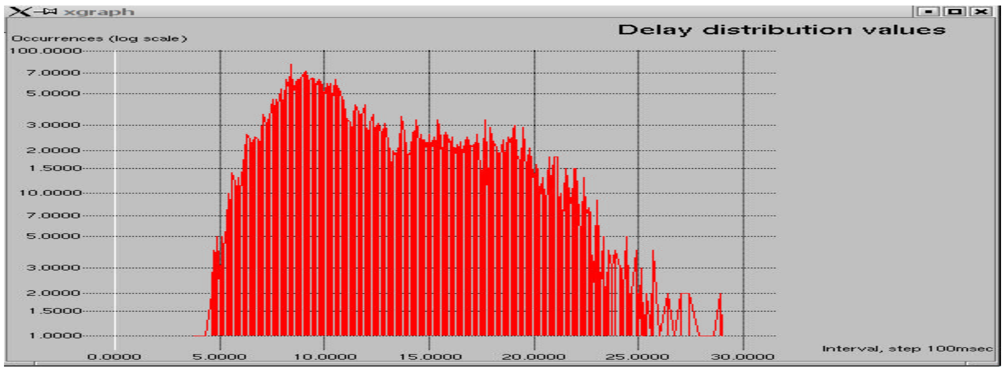


Figure 7.21 Delay Distribution of a VoIP (gsm) Flow in the IntServ Network Configuration

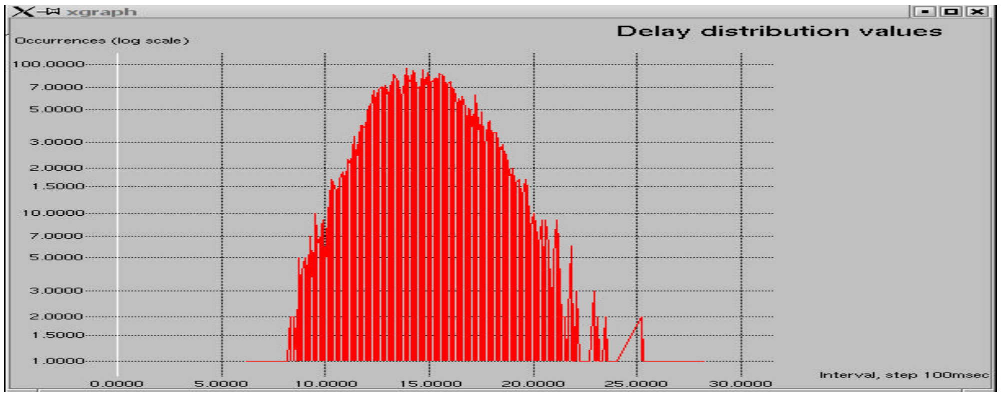


Figure 7.22 Delay Distribution of a VoIP (gsm) Flow in the IntServ-DiffServ Network Configuration

Figures 7.21 and 7.22 clearly explain how the combined network QoS architecture really represents a combination of the two approaches taking advantages of each other. Also delay distribution takes advantage from this solution, reducing significantly the wider range shown from the IntServ configuration.

The time analysis of a gsm IP phone call is now presented. Results for this flow are even more significant respect to the previous ones. Figure 7.23 presents the delay function in the DiffServ network. Packets belonging to this flow were market with AF11 codepoint and in presence of such a degree of congestion the delay introduced by the network is not tolerable anymore if compared to the two other network configurations and also to the inter-packet sending time: 20ms in this case. Only a couple of packets were transferred with an acceptable delay as we can notice from the minimum peak value around 45ms, but almost all packets of this flow suffer from too high delay values. Nevertheless, it should be noticed that even in this extreme congestion situation, the DiffServ network shows the smallest value for the jitter: 1,85 ms, with a mean delay of 83,15 ms.

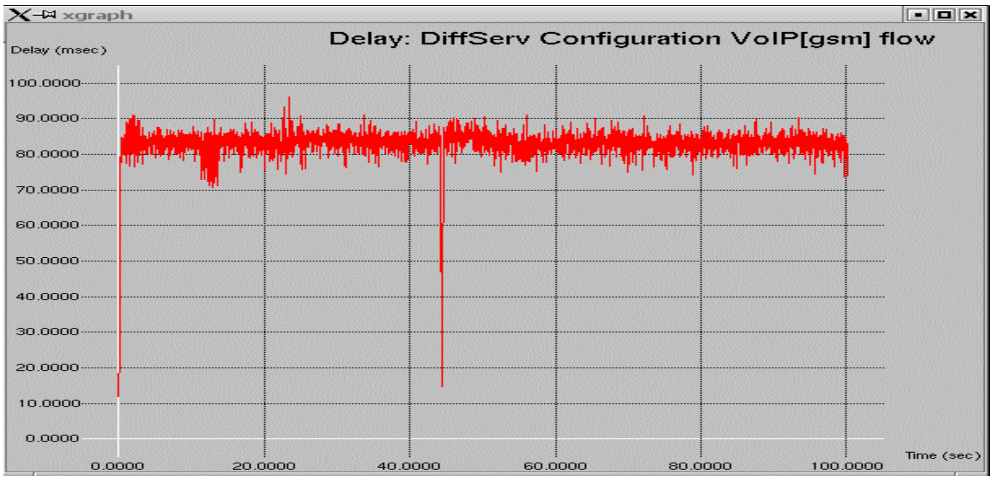


Figure 7.23 Delay of VoIP (gsm) Flow in the DiffServ Network Configuration

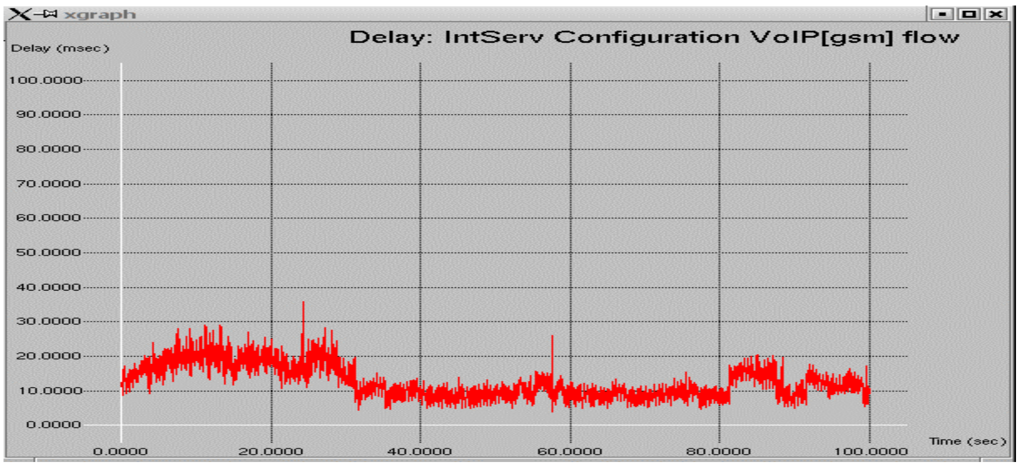


Figure 7.24 Delay of VoIP (gsm) Flow in the IntServ Network Configuration

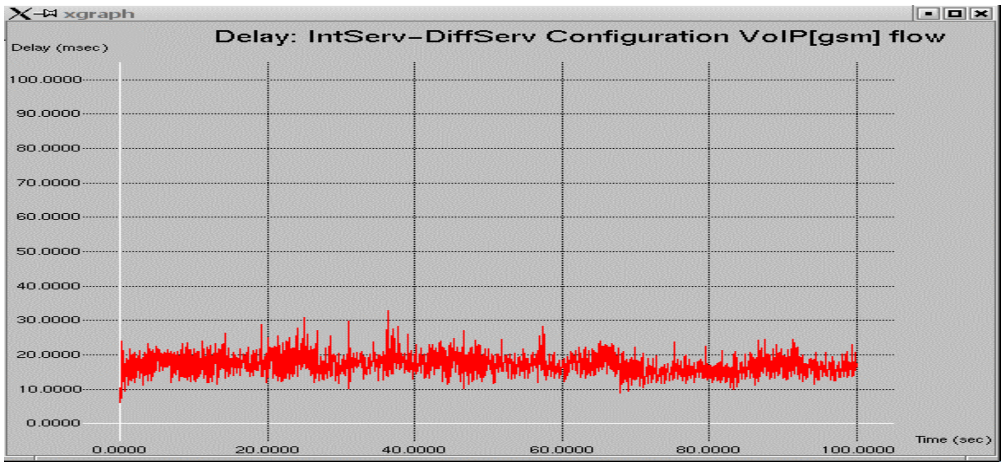


Figure 7.25 Delay of VoIP (adpcm) Flow in the IntServ-DiffServ Network Configuration

Figures 7.24 and 7.25 illustrate instead the delay for the two other QoS architectures analyzed. The results are concurrent with the preceding analysis: the IntServ handling of packets belonging to this flow presents a smaller mean value: 12,58 ms but a bigger average delay variation (3,91 ms) than the values obtained in our IntServ/DiffServ architectural proposal: 16,92 ms and 1,99 ms respectively.

The last figures present the delay distribution functions for each network, showing what is quite evident from the delay function. The delay values are much more concentrated around the mean value in the DiffServ network that however presents the biggest mean value. The IntServ network and especially the combination IntServ/DiffServ proposal move back this peak to a reasonable value maintaining a peaked shape that means a low jitter.

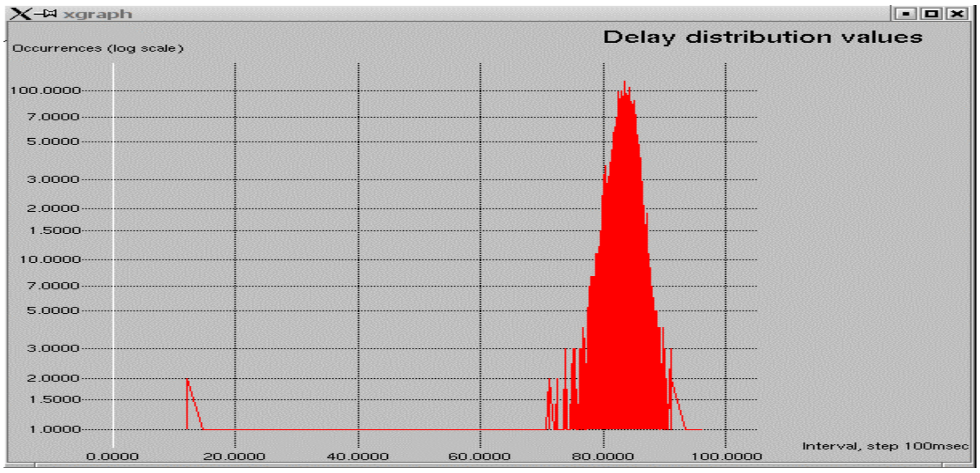


Figure 7.26 Delay Distribution of a VoIP (gsm) Flow in the DiffServ Network Configuration

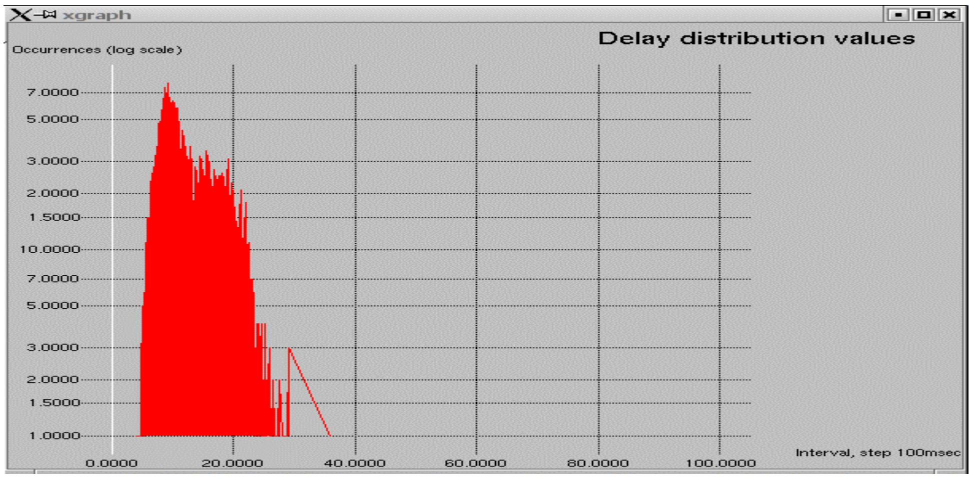


Figure 7.27 Delay Distribution of a VoIP (gsm) Flow in the IntServ Network Configuration

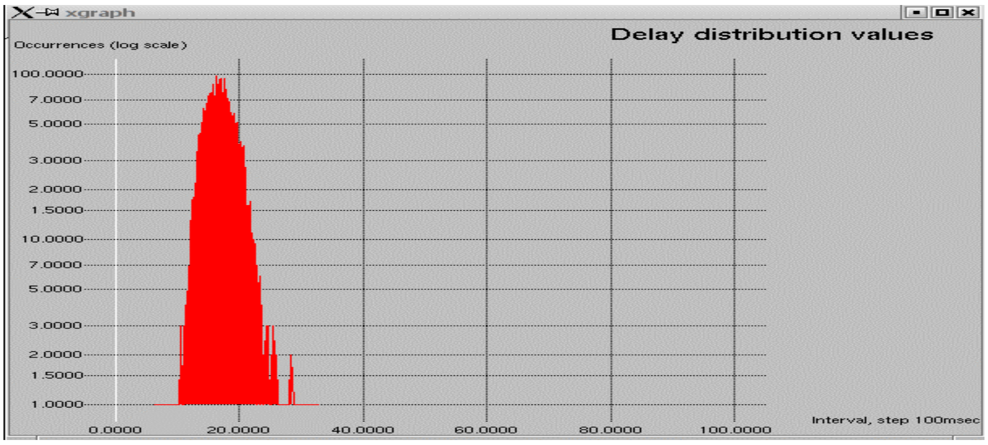


Figure 7.28 Delay Distribution of a VoIP (gsm) Flow in the IntServ-DiffServ Network Configuration

The measurements presented in this test case showed that the combination of IntServ and DiffServ architectures could provide the same level or even superior QoS than pure IntServ or DiffServ models when used alone. Even in this very simple test network adopted some problems tied with scalability issues of the IntServ/RSVP model and static class full traffic aggregation issues of the DiffServ approach were realized demonstrating the validity of our combined solution in this network environment.

7.4 Test Case 4: Network heavily overloaded including a wireless link

The network configuration for the final test case was modified introducing a base station and a laptop connected by a wireless link to the previous network topology. The new network topology is depicted in figure 7.29.

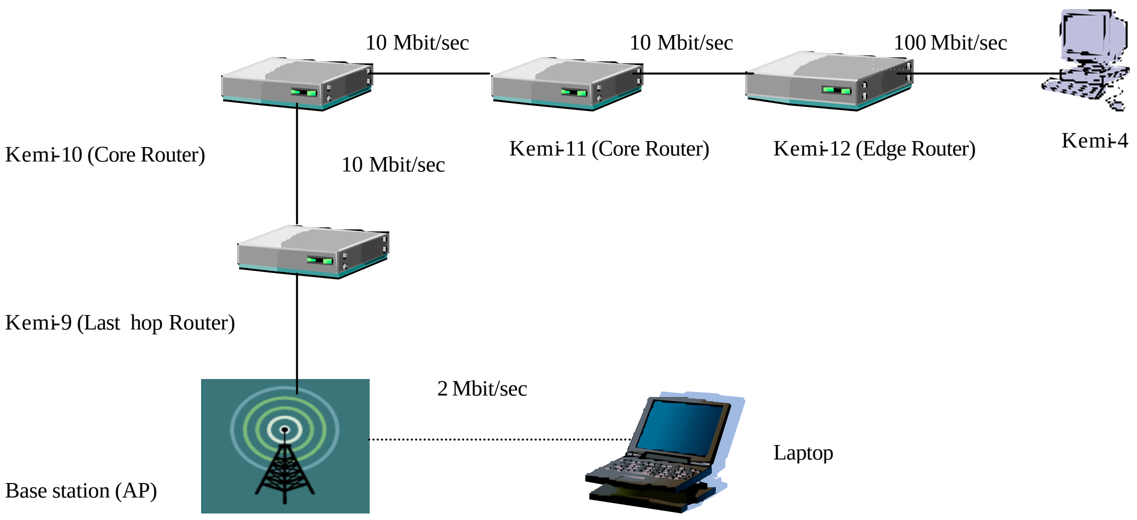


Figure 7.29 Network topology including the wireless link.

In this new network topology, kemi-9 is the destination of most of the traffic generated by kemi-4 and acts as router for traffic addressed to the mobile terminal linked by a 2 Mbit/s (MAC level capacity) wireless link to the network previously studied. The Wireless Access Point (AP) (BayStack 660) is a wireless-to-wired bridge that enables the mobile terminal (laptop) equipped with the BayStack 660 Wireless PC card to access the network wirelessly. The BayStack 660 Wireless LAN products use 2 Mbit/s Direct Sequence Spread Spectrum (DSSS) radio technology to deliver the maximum point-to-point performance under the IEEE 802.11 wireless LAN standard. The laptop was running the Linux Operating System (kernel version 2.4.17) and was RSVP enabled (release 4.2a4-3 of RSVP ISI daemon) [59] in the IntServ configuration. For more clearness we report in the following table the three network configuration set-ups used in this test case. Particular attention should be paid to the IntServ/DiffServ co-operation model, as we will show later in this chapter we choose for this QoS network architecture a combination of different queuing mechanisms and QoS approaches trying to get advantages from each of them.

Table 7.19 Network Configuration Set-ups

QoS Architecture	Laptop	Kemi-9	Kemi-10	Kemi-11	Kemi-12	Kemi-4
DiffServ		DIFF	DIFF	DIFF	DIFF	BE
IntServ	RSVP	RSVP/INT	RSVP/INT	RSVP/INT	RSVP/INT	RSVP/INT
Int/Diff	RSVP	BE	DIFF	DIFF	RSVP/INT	RSVP/INT

The workload used in the last test case is almost the same as that used in the previous test case (see table 7.11), only three flows, one adpcm VoIP phone call, one gsm VoIP phone call and one medium-high quality video stream whose characteristics were already described reach the mobile terminal. We analyzed in this final test case the QoS offered from the three network QoS architectures to these flows in a mobile environment.

The results obtained in this test case confirmed what we have demonstrated previously and particularly show that in a mobile environment with a slower link and a higher transmission error probability, the combination of IntServ and DiffServ introduces even more benefits than in a fixed one. The following tables illustrate the results for the flows reaching the mobile device. In respect to the previous case, as expected, the delay values are higher since the path between source and destination now count two more hops, and the number of packets lost is slightly increased at least for the flows with smaller packet size.

Table 7.20 VoIP (adpcm codec) Flow Results

VoIP (adpcm) Flow	DiffServ (EF)	IntServ (CL1)	Int/Diff
Recv date rate (kbit/s)	41,28	41,39	41,46
Packets Received	4962	4976	4983
Packets Dropped	38	24	17
Ave. Delay (ms)	15	24	20
Max. Delay (ms)	90	108	82
Min. Delay (ms)	3	7	5
Delay variation (ms)	87	102	77

For the EF-CL1 flow (table 7.20), representing an IP phone call in adpcm codec, all the three QoS mechanisms offer almost equal service at least in terms of throughput, in fact, only few more packets get lost in the DiffServ network; but as we will show later in the delay analysis, the token bucket filter handling of the DiffServ configuration offers better results for time parameters than stochastic fair queuing IntServ management of this flow. A behavior closer to the DiffServ one is shown in the combination of IntServ and DiffServ QoS approaches, with a slightly worse mean value but a lower average delay variation in respect to the DiffServ network.

Table 7.21 VoIP (gsm codec) Flow Results

VoIP (gsm) Flow	DiffServ (AF11)	IntServ (CL2)	Int/Diff
Recv date rate (kbit/s)	11,55	14,37	14,3
Packets Received	4012	4990	4967
Packets Dropped	988	10	33
Ave. Delay (ms)	90	27	27
Max. Delay (ms)	164	148	106
Min. Delay (ms)	6	8	7
Delay variation (ms)	158	141	99

The second flow analyzed was a VoIP (gsm) flow (table7.21), and the results for this flow as in test case 3 exhibit that the DiffServ mechanism is not sufficient anymore to guarantee bandwidth requirements for a CBR flow (about 20% of packets dropped) when the rate of packets marked with the same TOS byte (belonging then to the same class of traffic) overcame the bandwidth assigned to this class in the routers. The DiffServ architecture presents then its limit tied with its class full approach that does not permit to differentiate between flows belonging to the same class. The IntServ solution guarantees bandwidth requirements and sufficient response in terms of time parameters but the IntServ/DiffServ configuration shows the best results for this flow maintaining the same average delay and reducing the delay variation. The reason for this behaviour is tied with the last hop router packet handling, as we have demonstrated in the test case 2, best-effort with FIFO management reduces jitter and delay variation in an un-congested link because of its faster packet scheduling mechanism. Thus the proposed combination of the two mechanisms, guaranteeing bandwidth in the access link, DiffServ management in the core routers and best-effort handling in the last-hop router, overcomes the two QoS architectures used alone in this case.

Table 7.22Video Flow Results

Video Flow (VBR)	DiffServ (AF2.3)	IntServ (CL 3)	Int/Diff
Recv date rate (kbit/s)	171,8	201,33	201,48
Packets Received	7164	8392	8395
Packets Dropped	1248	8	5
Ave. Delay (ms)	128	27	25
Max. Delay (ms)	213	149	93
Min. Delay (ms)	16	7	6
Delay variation (ms)	197	142	87

As expected, the AF23-CL3 (200Kbit/sec mean data rate) medium-high quality Video stream (table 7.22) also shows a behavior similar to the previous flow analyzed, manifesting again that DiffServ QoS management does not offer any guarantee in such a congestion situation. Also for this flow with a bigger packet size and higher data rate, the combination of the two mechanisms again offers the best service level overcoming even the service level offered by the IntServ architecture, because, as already shown the RED management of this flow offered by the DiffServ core routers and best-effort handling at the last hop router guarantees lower delays and especially lower jitter values. The following figures represent a time analysis (Delay and Delay distribution) for the adpcm VoIP phone call, the gsm VoIP flow and the Video stream. We examined the three flows making comparison between the three different QoS architectures. These figures represent the best case between ten individual replications for each network configuration.

The results obtained in this analysis are consistent with what was previously exposed in the test case 3, for each flow as we could have expected the mean delay and average delay variation are larger because of the presence of the wireless link between kemi-9 and the mobile terminal that introduces an additive delay component and a more jittered delay shape, but we should keep in mind that the values from the next figures represent the results from one single replication and even if they are related to the previous summarizing tables (average values) could present some differences, especially for the jitter values, considering further that we made comparison between the best cases for each network configuration.

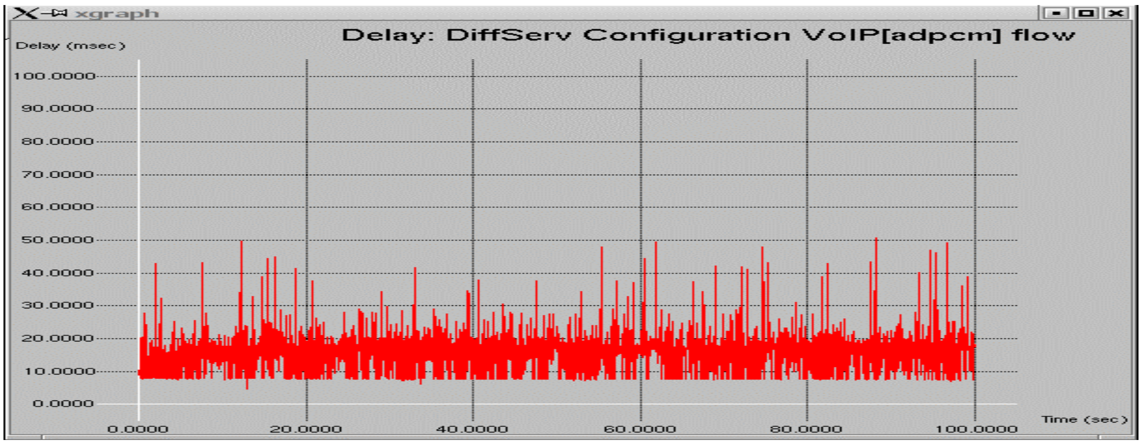


Figure 7.30 Delay of VoIP (adpcm) Flow in the DiffServ Network Configuration

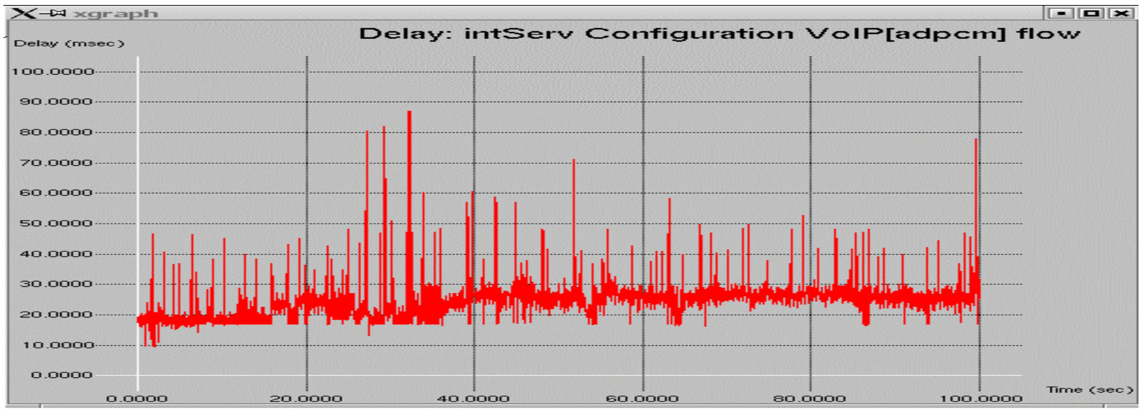


Figure 7.31 Delay of a VoIP (adpcm) Flow in the IntServ Network Configuration

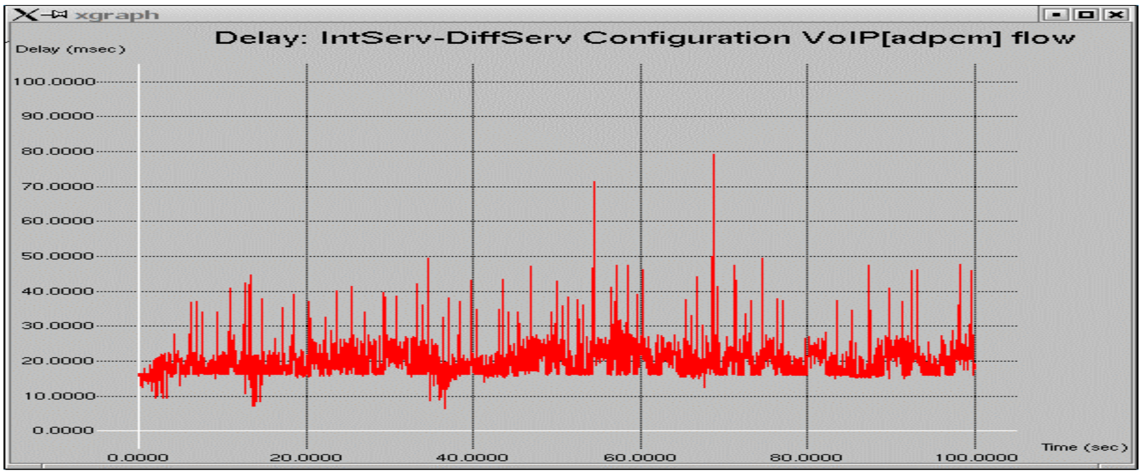
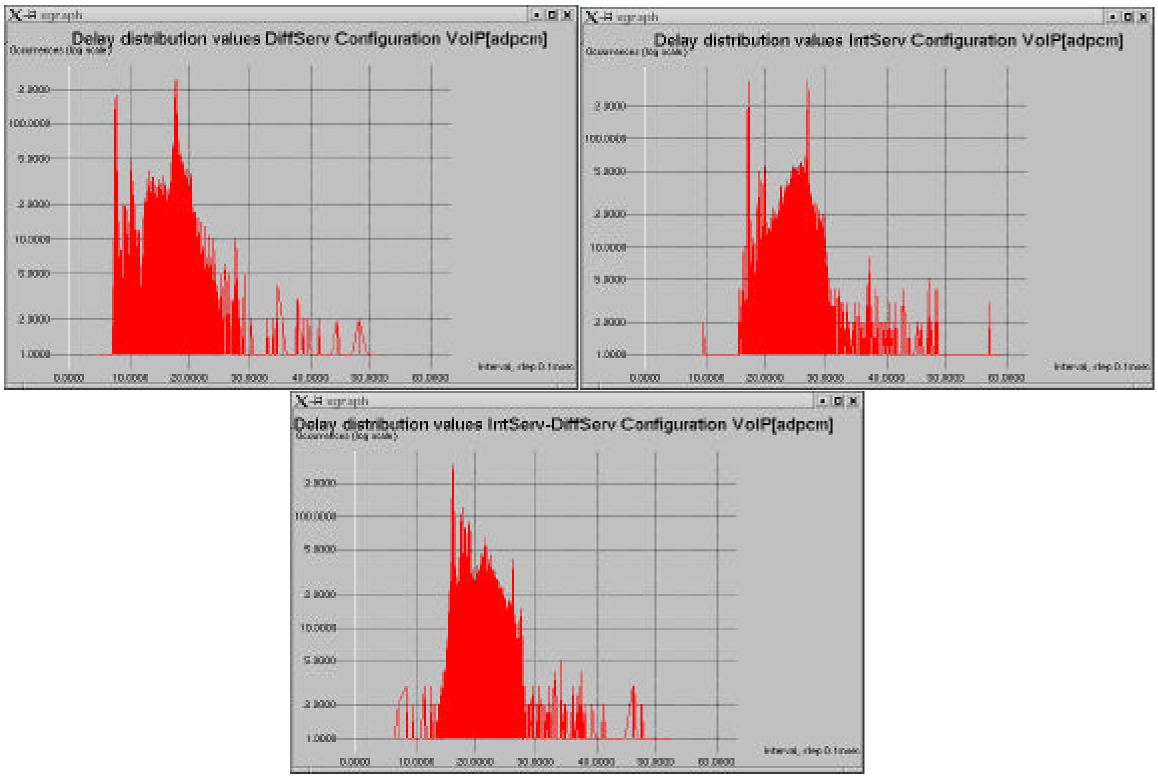


Figure 7.32 Delay of a VoIP (adpcm) Flow in the IntServ/DiffServ Network Configuration

The figures on the previous page illustrate the Delay function for the adpcm VoIP flow in each network configuration. The DiffServ architecture offers the best response to this EF flow at least in terms of mean delay presenting a mean delay of 15,9 ms and average delay variation of 3,96 ms.

A more irregular shape presents the delay function in the IntServ configuration, oscillating between 10 and 85 ms with a mean value and average delay variation of 24,59 and 3,9 ms respectively. We remind the reader that these values represent the best case for each network configuration and are not so important in absolute term for our comparison. The last figure represents the delay function for the same flow in our IntServ/DiffServ cooperation proposal, it clearly appears as a combination of the two previous ones, taking advantage in terms of jitter (3,37 ms) from both the two QoS architectures and keeping the mean value reasonably low (20,05 ms) representing then a very good solution for a real-time application intolerant to the jitter.

The next three pictures represent the delay distribution function for the same flow in the three QoS mechanisms compared. Confirming what we found in the table regarding this flow and in the previous figures, delay distribution in the DiffServ network presents more values concentrated before 20 ms than IntServ and Int/Diff configurations, but IntServ and particularly the combination show a more peaked shape around their mean value, which means low jitter.



7.33 Delay Distribution of a VoIP (adpcm) Flow in each network configuration

The next pictures show the time analysis of the gsm VoIP flow, as in the test case 3, the results for this flow show the benefits introduced by combined network QoS architecture even better.

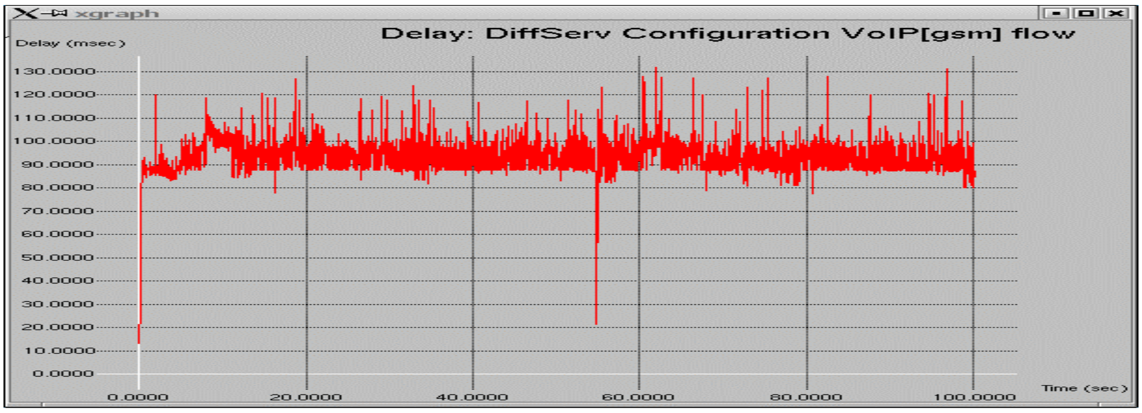


Figure 7.34 Delay of a VoIP (gsm) Flow in the DiffServ Network Configuration

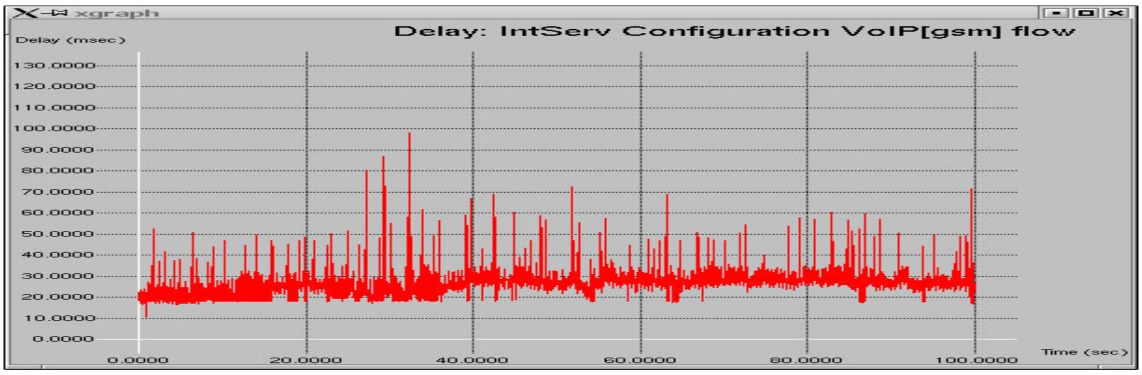


Figure 7.35 Delay of a VoIP (gsm) Flow in the IntServ Network Configuration

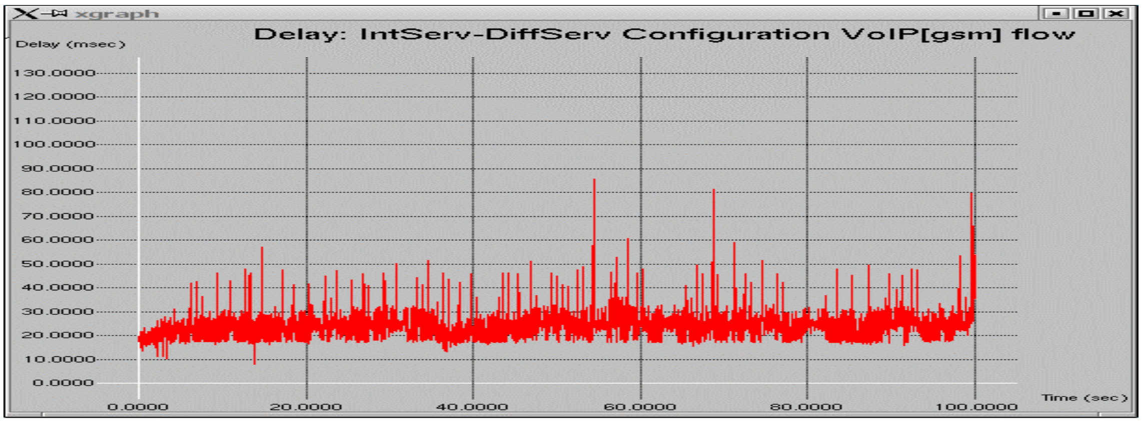
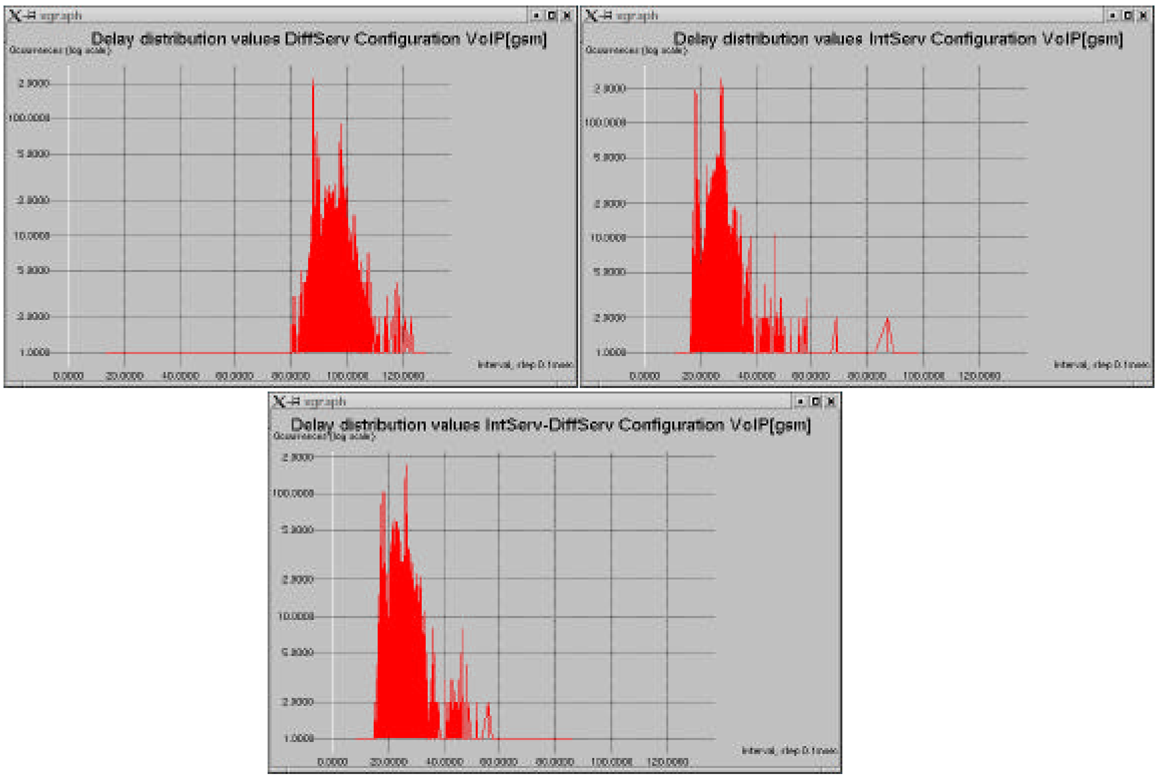


Figure 7.36 Delay of a VoIP (gsm) Flow in the IntServ/DiffServ Network Configuration

Figure 7.34 presents the delay function in the DiffServ network. Packets belonging to this flow were marked with the AF11 codepoint and in presence of such a degree of congestion the delay introduced by the network is not tolerable anymore if compared to the two other configurations examined and also to the inter-packet sending time, 20ms in this case. Nevertheless it should be noticed that even in this extreme congestion situation, the DiffServ network shows an acceptable value for the jitter: 5,24 ms, at least if compared to its mean delay: 93,58 ms.

Instead figures 7.35 and 7.36 illustrate the delay for the two other QoS architectures analyzed. The results are concurrent with the preceding analysis, our IntServ/DiffServ architectural proposal presents a smaller mean value, 26,72 ms, but a slightly larger average delay variation (4,12 ms) than the values obtained in the IntServ handling of packets belonging to this flow, 26,72 ms and 4,01 ms respectively.

Figure 7.37 presents delay distribution functions for each network configuration, showing what is quite evident from the delay function. The delay values are almost concentrated around the mean value in the DiffServ network that, however, presents the largest mean value. The IntServ network and especially the combined IntServ/DiffServ proposal move back this peak to a reasonable value maintaining a peaked shape that always means a lower jitter.



7.37 Delay Distribution of a VoIP (adpcm) Flow in each network configuration

The last pictures finally show the time analysis of the video flow analyzed. Packets belonging to this flow were marked with the AF23 codepoint.

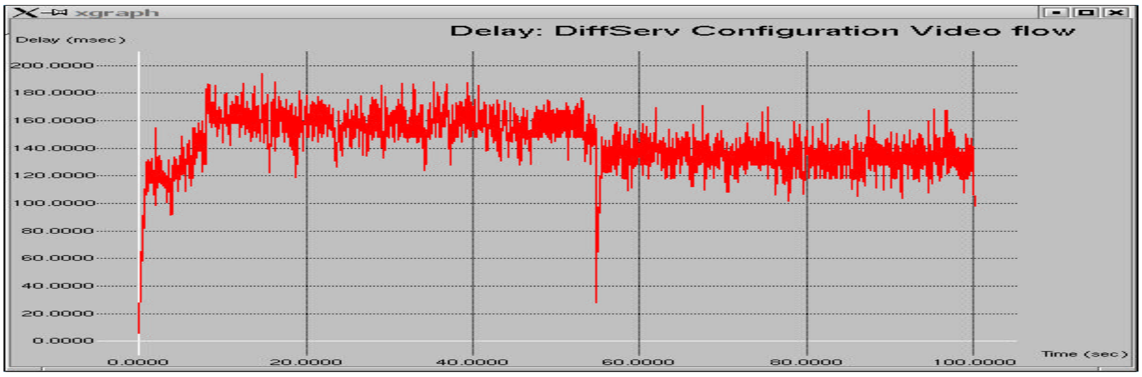


Figure 7.38 Delay of a High Quality Video Stream in the DiffServ Network Configuration

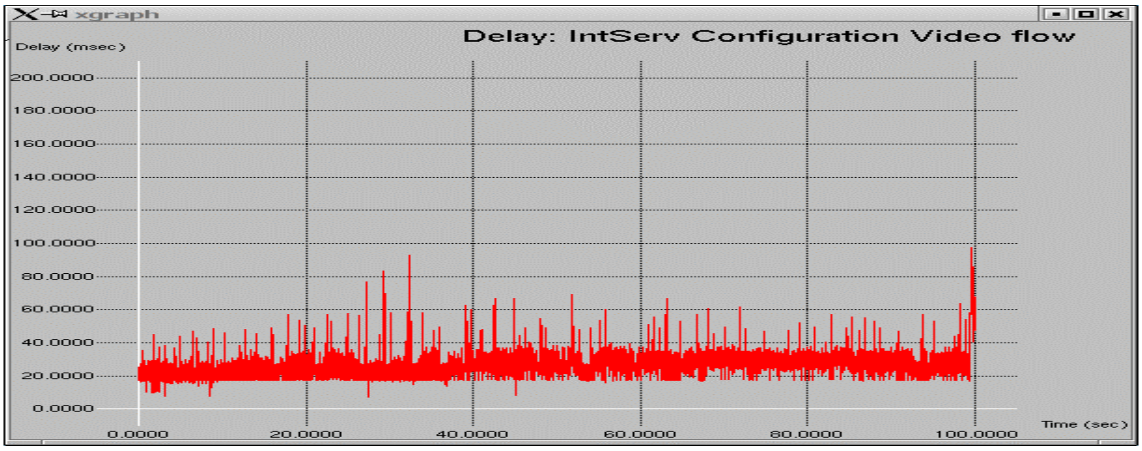


Figure 7.39 Delay of a High Quality Video Stream in the IntServ Network Configuration

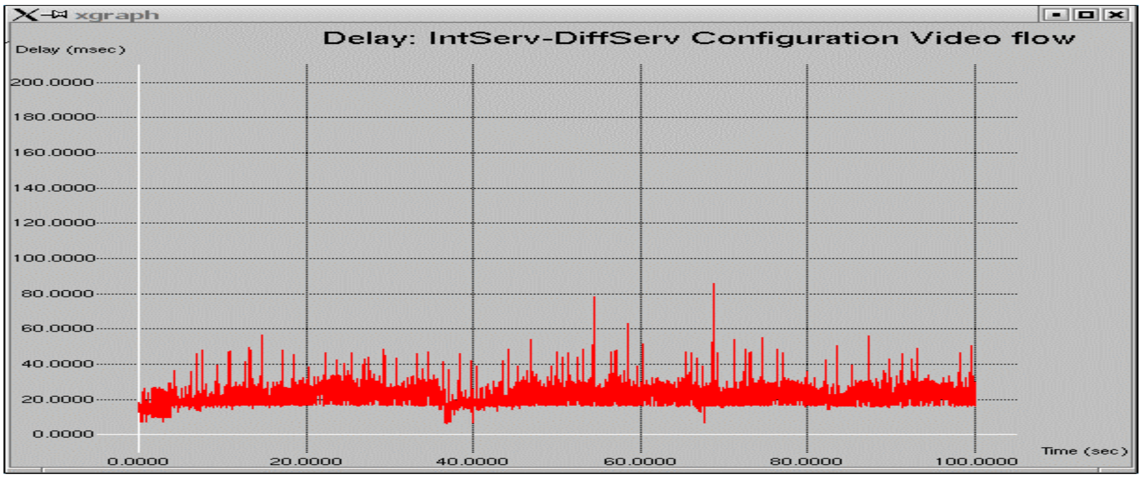


Figure 7.40 Delay of a High Quality Video Stream in the IntServ-DiffServ Network Configuration

The network behaviour for this flow with a bigger packet size and a higher data rate presents results very close to the previous case. In fact in the presence of such a degree of congestion the delay introduced by the DiffServ network is no longer tolerable if compared to the two other network configurations. Confirming the previous results, AF flows get the best results from the combined network configuration because the RED management of these flows only in the core network and a resources reservation in the access link guarantee the ideal solution in this overloaded condition.

Figure 7.41 presents the delay distribution functions of this video flow for each network configuration. As quite evident from the delay function, the delay values are quite spread around the mean value in the DiffServ network that further presents the largest mean value. The IntServ network and especially the combined IntServ/DiffServ architecture proposal as for the gsm flow move back this peak to a reasonable value maintaining a peaked shape guaranteeing a lower jitter.

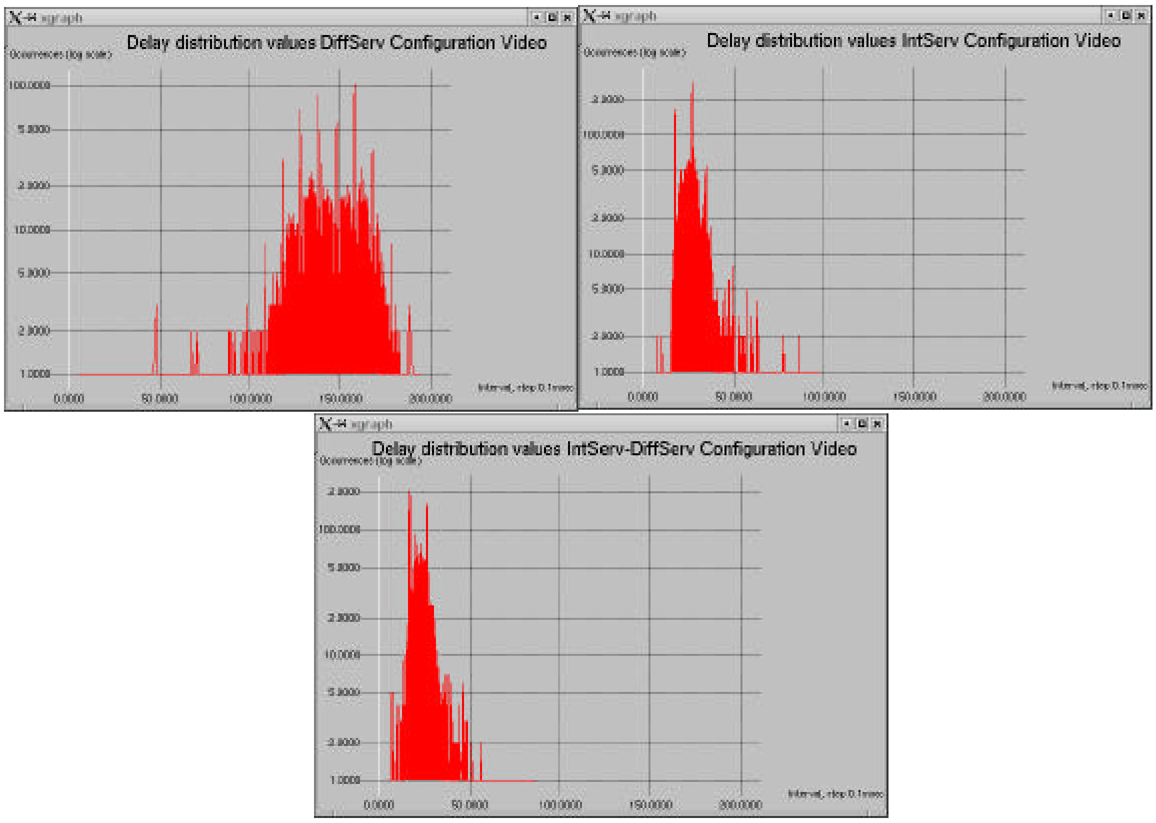


Figure 7.41 Delay Distribution of a VoIP (adpcm) Flow in each network configuration

8 Conclusion

In this thesis we have presented a combination of the two best known IP QoS architectures: Integrated Services and Differentiated Services as a solution for QoS provisioning in an IP radio access network. In a more realistic network topology the co-operation model proposed is based on the assumption that Integrated Services and RSVP are utilized in the access networks where the traffic volume is lower but less predictable and the more robust and scalable DiffServ approach is used in the core network where the traffic volume is higher.

The measurements in the test network did not illustrate clearly the scalability problem because of the simple network topology adopted, but this objective was beyond the scope of this thesis. Instead the limit for the DiffServ configuration was very clear when the network was heavily overloaded because of its static bandwidth subdivision and management. And what was even clearer was the improvement introduced by a mixed use of the two QoS architectures.

From our comparative analysis of different network configurations loaded with diverse kinds of workloads we can conclude that the DiffServ management (TBF, RED and GRED) inside a priority queuing of real-time flows guarantees the best performance in terms of delay and jitter response quite independently from packet size and packet rate - at least until there is available bandwidth for the served traffic class. It does not offer, however, any guarantees in case of congestion. The IntServ approach using the RSVP mechanism can guarantee bandwidth requirements for selected flows even in an extreme congestion situation but causes a worse link bandwidth utilization and also an inferior service level in terms of time parameters.

Our IntServ over DiffServ interoperation model has shown noticeable improvements in respect to the two mechanisms when used alone and provided satisfactory QoS guarantees even in an extreme congestion situation, optimising the link utilization as shown by the overall throughput in the test case 3 and improving the time response of the network for the real-time flows analyzed.

One other important result we showed in our experiments is the effectiveness of the best-effort handling of packets in the last hop router combined with the traffic shaping in the access link. The simple scheduling mechanism (FIFO) of the best-effort model in fact guarantees a faster packet scheduling improving delay and jitter response of a network lightly loaded. Concerning the CBQ scheduling mechanism, we observed worse behavior in terms of delay and delay variation than in other scheduling mechanisms analyzed (e.g. priority queuing) within a variable packet size regime, because of its imperfect resource-sharing algorithm.

Regarding the TCP transfer bulk, we observed that the stochastic fair queuing management of this flow offered inside a CBQ by the IntServ network shows better performance in respect to the RED management offered inside a priority queue by the DiffServ core routers. This is because the CBQ mechanism avoids resources starvation of the lowest precedence queues caused by the simpler pr iority queuing mechanism.

Although the objectives of this thesis are almost fully completed, a lot more research is still required in order to achieve end-to-end QoS in wired and wireless networks and there can be a number of different research directions to improve the QoS mechanisms. In this thesis for example, we only took into account unidirectional real-time streams and we considered the traffic management for the downlink. But a further issue to develop could be the uplink bandwidth management of the wireless link. We also considered only a static mapping option between IntServ and DiffServ models but a more efficient solution could be offered by a dynamic bandwidth management mechanism like Bandwidth Broker. Finally, further work could focus on the integration of other service classes as well as on the introduction of an admission control procedure.

9 Acknowledgements

Many thanks to my supervisors Professors Kimmo Raatikainen and Sergio Palazzo for the opportunity that they gave me to develop this thesis and for their invaluable advice, direction and comments on this thesis.

I would like to thank Jukka Manner for his practical support and suggestions about my thesis, all in the Computer Science Department of University of Helsinki for making this an enjoyable and stimulating period's study.

I am also grateful to Simone Leggio and Davide Astuti for their practical helps and encouragements during this period.

10 References

[1] Braden, R., Clark, D. Shenker, S., “Integrated Services in the Internet Architecture: an Overview “. Internet Engineering Task Force, Request for Comments (RFC) 1633, June1994.

[2] Blake, S., Black, D., Carlson, M., Davies, E., Wang,, Z., Weiss, W., “An Architecture for Differentiated Services”. Internet Engineering Task Force, Request for Comments (RFC) RFC 2475, Dec. 1998.

[3] Wroclawski, J. Charny, A., “Integrated Service Mappings for Differentiated Services Networks”. Internet Draft (work in progress), February 2001 (draft-ietf-issll-ds-map-01.txt).
(expired but rewritten as new draft)

[4] Bernet, Y. et al, “A Framework For Integrated Services Operation Over Diffserv Networks”. Internet Engineering Task Force, Request for Comments (RFC) 2998, November 2000.

[5] Blair, G.S., Davies, N., Friday, A., Wade, S.P.: Quality of service support in a mobile environment: an approach based on tuple spaces *Proceedings of the 5th IFIP International Workshop on Quality of Service 1997* (Chapman & Hall 1997)

[6] Bochmann, G. v., Hafid, A.: Some principles for quality of service management *Distributed Systems Engineering* vol4 p16-27 (IOP Publishing,1997)

[7] Broadband Radio Access for IP-based Networks, IST-1999-10050, http://www.ist-brain.org

[8] Summary: Introduction, ATM network operation, ATM signaling and addressing, routing protocols (IISP, P-NNI), LAN emulation, native mode protocols (IP over ATM, NHRP), multiprotocol over ATM, wide area internetworking, survey of ATM traffic management, status of key ATM standards and specifications. (By Anthony Alles, Cisco Systems, 1995.)

[9] Rosen. E., Viswanathan, A., Callon, R., “Multiprotocol Label Switching Architecture”, Request for Comments (proposed standard) 3031, Internet Engineering Task Force, January 2001

[10] Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S., .Resource ReSerVation protocol (RSVP) -- Version 1, Functional Specification.. Internet Engineering Task Force, Request for Comments 2205, September 1997.

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

[12] Shenker, S., Partridge, C., Guerin, R., "Specification of Guaranteed Quality of Service", RFC 2212,September 1997.

[13] Wroclawski, J., "Specification of the Controlled-Load Network Element Service, RFC 2211, September 1997.

[14] Nichols, K., Blake, S., Baker, F., Black, D., " Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474 December 1998.

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

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

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

[18] QoS Working group: "QBone Bandwidth Broker Architecture" Work in Progress http://qbone.internet2.edu/bb/bboutline2.html

[19] Schulzrinne, H., et al. RTP: A Transport Protocol for Real-Time Applications.. Internet Engineering Task Force, Request for Comments 1889, January 1996.

[20] Manner, J., et al., Mobility Related Terminology. Internet Draft (work in progress), January 2001 (draft-manner-seamoby-terms-00.txt).

[21] Perkins, C., "IP Mobility Support", Internet Engineering Task Force, Request for Comments (RFC) 2002, October 1996.

[22] Johnson, D., Perkins, C., “Mobility Support in IPv6”, Internet Draft (work in progress), November 2000 (draft-ietf-mobileip-ipv6-13.txt).

[23] Jucca Manner, Alberto Lopez, Andrej Mihailovic, Hector Velajos, Eleanor Hepworth, Youssef Khouaja “Evaluation of mobility and quality of service interaction”. Received in revised from 21 June 2001.

[24] Castelluccia, C., Bellier, L., “Hierarchical Mobile IPv6”, Internet Draft (work in progress), July 2000 (draft-castelluccia-mobileip-hmipv6-00.txt).

[25] Malinen, J., Perkins, C., “Mobile IPv6 Regional Registrations”, Internet Draft (work in progress), July 2000 (draft-malinen-mobileip-regreg6-00.txt).

[26] McCann, P., Hiller, T., Wang, J., Casati, A.,Perkins, C., Calhoun, P., “Transparent Hierarchical Mobility Agents (THEMA)”, Internet Draft (work in progress), March 1999 (draft-mccann-thema-00.txt).

[27] El Malki, K., Soliman, H., “Fast Handoffs in Mobile IPv4”, Internet Draft (work in progress), September 2000 (draft-elmalki-mobileip-fasthandoffs-03.txt).

[28] Ramjee, R., La Porta, T., Thuel, S., Varadhan, K., “IP micro-mobility support using HAWAII”, Internet Draft, (work in progress), June 1999 (draft-ietf-mobileip-hawaii-00).

[29] Campbell A.T., Gomez J. & Valko' A. " An Overview of Cellular IP". In : IEEE Wireless Communication and Networking Conference September 21-24, New Orleans, LA USA.

[30] Shelby, Z.D, Gatzounas, D., Campbell, A. Wan, C., “Cellular IPv6”. Internet Draft, (work in progress), November 2000 (draft-shelby-seamobycellularipv6-00).

[31] Mysore, J., Bharghavan, V., “A New Multicasting-based Architecture for Internet Host Mobility”, Proceedings of ACM Mobicom, September 1997.

[32] Mihailovic, A., Shabeer, M., Aghvami, A.H., "Multicast for Mobility Protocol (MMP) for emerging Internet networks", To appear in Proceedings of PIMRC2000, London, UK, September 2000.

[33] O'Neill, G., Corson, S., "Homogeneus Edge Mobility for Heterogeneous Access Technologies" Proceedings of the IPCN 2000, Paris, France, May 2000.

[34] Lee et al., "INSIGNIA: An IP-based quality of service framework for mobile adhoc networks". Journal of parallel and distributed computing, Vol 60 no 4, pp 374-406, April 2000.

[35] Talukdar, A., Badrinath, B. Acharya, A., "MRSVP: A Resource Reservation Protocol for an Integrated Services Packet Network with Mobile Hosts", In Proceedings of ACTS Mobile Summit'98, June 1998.

[36] Chen, J-C, et al. "QoS Architecture Based on Differentiated Services for Next Generation Wireless IP Networks". Internet Draft (work in progress), July 2000 (draft-itsumo-wirelessdiffserv-00.txt).

[37] 3rd Generation Partnership Project, "General Packet Radio Service (Release 1999); Service Description stage 1" Doc. no.: 3G TS 22.060, March 2000.

[38] G. Priggouris, L. Merakos, S. Hadjiefthymiades "Supporting IP QoS in General Radio Packet Service"

[39] RICHARDSON, K.W. UMTS overview *Electronics & Communications Engineering Journal* 12,3 (June 2000), 93-100.

[40] Y. Bernet, S. Blake, D. Grossman, A. Smith , "An Informal Management Model for Diffserv Routers", Internet Draft , draft-ietf-diffserv-model-06.txt, February 2001.

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

[42] Adolfo Rodriguez, John Gatrell, John Karas, Roland Peschke "TCP/IP Tutorial and Technical Overview" August 2001 International Technical Support Organization.

[43] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, April 1999. Also see http://tcpsat.lerc.nasa.gov/tcpsat/papers.html.

[44] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC 2582, April 1999. Also see http://www.aciri.org/floyd/tcp small.html.

[45] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. RFC 2481, October 1996.

[46] M. Mathis and J. Mahdavi. Forward acknowledgment: Refining TCP congestion control. In *Proceedings of the ACM SIGCOMM*, August 1996. Also see http://www.psc.edu/networking/papers/papers.html.

[47] L. Brakmo, S. O'Malley, and L. Peterson. TCP vegas: New techniques for congestion detection and avoidance. In *Proceedings of the ACM SIGCOMM*, pages 24–35, August 1994. http://netweb.usc.edu/yaxu/Vegas/Reference/vegas93.ps.

[48] Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion avoidance". In ACM/IEEE Transactions on Networking, (3)1, August 1993.

[49] Floyd, S., RED: Discussions of Setting Parameters, November 1997, http://www.aciri.org/floyd/REDparameters.txt (June 2001).

[50] Braden, B., D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, Recommendations on Queue Management and Congestion Avoidance, RFC 2309, April 1998.

[51] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. RFC 2481, January 1999.

[52] "Interoperation of Controlled-Load Service and Guaranteed Service with ATM," August 1998, 43 Pages, ftp://ftp.isi.edu/in-notes/rfc2381.txt Talks about the interoperation of Controlled Load Service and Guaranteed Service with ATM

[53] Köhler S., Schäfer U. “Performance Comparison of Different Class-and-Drop Treatment of Data and Acknowledgements in DiffServ IP Networks”, Report No. 237, University of Wuerzburg, August 1999.

[54] Bernet, Y., “Format of the RSVP DCLASS Object”. Internet Engineering Task Force, Request for Comments (RFC) 2996, November 2000.

[55] Huston G., “Internet Performance Survival guide”, Wiley Computer publishing, 2000.

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

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

[58] TC Traffic control tool, available via, available via ftp://ftp.inr.ac.ru/ip-routing/ version iproute2-2.2.4-now-ss991023.tar

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

[60] ftp://manimac.itd.nrl.navy.mil/Pub/MGEN/dist/, January 2000.

[61] NTP (Network Time Protocol) available via http://www.eecis.udel.edu/~ntp/

[62] http://www.radcom-inc.com

[63] http://diffserv.sourceforge.net/

11 Appendix

This Appendix contains the tc shell scripts used in the lab for implementing our DiffServ and IntServ routers. They implement CBQ on Intserv Interface and DSmark on DiffServ interfaces as described in chapter 6, and based on the scripts available on the Iproute2 package [58] and on the DiffServ home page [63] respectively.

DiffServ interface script

```
#!/bin/sh

# DSCP and TOS values used (based on RFC 2597 and 2598)
# ---------------------------------------------------
# AF DSCP values                          AF TOS VALUES
# AF1 1. 0x0a 2. 0x0c 3. 0x0e   AF1 1. 0x28 2. 0x30 3. 0x38
# AF2 1. 0x12 2. 0x14 3. 0x16   AF2 1. 0x48 2. 0x50 3. 0x58
# AF3 1. 0x1a 2. 0x1c 3. 0x1e   AF3 1. 0x68 2. 0x70 3. 0x78
# AF4 1. 0x22 2. 0x24 3. 0x26   AF4 1. 0x88 2. 0x90 3. 0x98

#
# EF DSCP value                 EF TOS value
# 0x2e                          0xb8

# Variables
TC='/root/src/liproute2/tc/tc'
ETH='eth0'
#link capacity
BANDWIDTH='10Mbit'

EF_RATE='1Mbit'
AF_RATE1='3Mbit'
AF_RATE2='3Mbit'
BE_RATE='3Mbit'

EF_BURST='200Kb'
EF_MTU='1.5Kb'
EF_LIMIT='3Kb'

BE_INDEX='0'
EF_QUEUE_LIMIT='5'

#change this
GRIO='grio'
GRIO_FLAG=1

BE_LIMIT='60KB'
BE_MIN='15KB'
BE_MAX='45KB'
BE_BURST='20'
BE_PROB='0.4'
```

```
#AFx1
DP_LIMIT1='60KB'
DP_MIN1='15KB'
DP_MAX1='45KB'
DP_BURST1='20'

#AFx3
DP_LIMIT3='60KB'
DP_MIN3='15KB'
DP_MAX3='45KB'
DP_BURST3='20'

#DSMARK, BA classifier, based on incoming DSCP >>> skb->tc_index

echo 'installing DSMARK'

$TC qdisc add dev $ETH handle 1:0 root dsmark indices 64 set_tc_index

#default_index 0
$TC filter add dev $ETH parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2 pass_on
$TC qdisc add dev $ETH parent 1:0 handle 2:0 prio
$TC filter add dev $ETH parent 2:0 protocol ip prio 1 tcindex mask 0xf00 shift 8

###########################################################################
#  EF priority class, TBF, and its filter
echo 'installing EF'
$TC filter add dev $ETH parent 1:0 protocol ip prio 1 handle 0x01 tcindex classid 1:111
$TC filter add dev $ETH parent 2:0 protocol ip prio 1 handle 1 tcindex classid 2:1
$TC qdisc add dev $ETH parent 2:1 tbf rate $EF_RATE burst $EF_BURST mtu $EF_MTU
limit $EF_LIMIT

# AF & BE CBQ, and its filter
echo 'installing CBQ'

$TC filter add dev $ETH parent 2:0 protocol ip prio 1 handle 2 tcindex classid 2:2
$TC qdisc add dev $ETH parent 2:2 handle 4:0 cbq bandwidth $BANDWIDTH cell 8 avpkt
1500 mpu 64

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

###########################################################################
# BE
echo 'installing BE'
$TC filter add dev $ETH parent 1:0 protocol ip prio 1 handle 0 tcindex classid 1:250
$TC class add dev $ETH parent 4:0 classid 4:5 cbq bandwidth $BANDWIDTH rate
$BE_RATE avpkt 1500 prio 6 bounded allot 1514 weight 1 maxburst 21
$TC filter add dev $ETH parent 4:0 protocol ip prio 1 handle 5 tcindex classid 4:5
$TC qdisc add dev $ETH parent 4:5 red limit $BE_LIMIT min $BE_MIN max $BE_MAX
burst $BE_BURST avpkt 1500 bandwidth $BANDWIDTH probability $BE_PROB
```

```
########################################################################
# AF1 Class
echo 'installing AF1'
$TC class add dev $ETH parent 4:0 classid 4:1 cbq bandwidth $BANDWIDTH rate
$AF_RATE1 avpkt 1500 prio 4 bounded allot 1514 weight 1 maxburst 21
$TC filter add dev $ETH parent 4:0 protocol ip prio 1 handle 1 tcindex classid 4:1
$TC qdisc add dev $ETH parent 4:1 gred setup DPs 3 default 3 $GRIO

# AF11
echo 'installing AF11'
   $TC filter add dev $ETH parent 1:0 protocol ip prio 1 handle 0x02 tcindex classid 1:211
   $TC qdisc change dev $ETH parent 4:1 gred limit $DP_LIMIT1 min $DP_MIN1 max
$DP_MAX1 burst $DP_BURST1 avpkt 1500 bandwidth $BANDWIDTH DP 1 probability 0.02
prio 2

########################################################################
# AF2 Class

echo 'installing AF2'
$TC class add dev $ETH parent 4:0 classid 4:2 cbq bandwidth $BANDWIDTH rate
$AF_RATE2 avpkt 1500 prio 5 bounded allot 1514 weight 1 maxburst 21
$TC filter add dev $ETH parent 4:0 protocol ip prio 1 handle 2  tcindex classid 4:2
$TC qdisc add dev $ETH parent 4:2 gred setup DPs 3 default 3 $GRIO

# AF23
   $TC filter add dev $ETH parent 1:0 protocol ip prio 1 handle 0x03 tcindex classid 1:223
   $TC qdisc change dev $ETH parent 4:2 gred limit $DP_LIMIT3 min $DP_MIN3 max
$DP_MAX3 burst $DP_BURST3 avpkt 1500 bandwidth $BANDWIDTH DP 3 probability 0.06
prio 4
```

IntServ interface script

```
#! /bin/sh
##########################################################################
# Configuration of kemi-12 eth0 interface as classical Intserv interface #
##########################################################################
TC=/root/src/iproute2/tc/tc
ETH=eth0
BANDWIDTH="bandwidth 10Mbit"

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

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

# Create root class with classid 1:1.
$TC class add dev $ETH parent 1:0 classid :1 est 1sec 8sec cbq \
$BANDWIDTH rate 10Mbit allot 1514 maxburst 50 avpkt 1000

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

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

# OPTIONAL.
# Attach "sfq" qdisc to this class, quantum is MTU, perturb
# gives period of hash function perturbation in seconds.
#
$TC qdisc add dev $ETH parent 1:2 sfq quantum 1514b perturb 15
# Interactive-burst class
$TC class add dev $ETH parent 1:1 classid :3 est 2sec 16sec cbq \
$BANDWIDTH rate 1Mbit allot 1514 weight 100Kbit \
prio 2 maxburst 100 avpkt 1000 split 1:0 defmap c0
$TC qdisc add dev $ETH parent 1:3 sfq quantum 1514b perturb 15


# Realtime class for RSVP

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

```
# Reclassified realtime traffic
# New element: split is not 1:0, but 1:7FFE. It means,
# that only real-time packets, which violated policing filters
# or exceeded reshaping buffers will fall to it.
$TC class add dev $ETH parent 1:7FFE classid 1:7FFF est 4sec 32sec cbq \
rate 1Mbit $BANDWIDTH allot 1514b avpkt 1000 weight 10Kbit \
prio 6 maxburst 10 split 1:7FFE defmap ffff
```