A QoS Measurement-Based Analysis of the Expedited Forwarding Per Hop Behaviour

Tiziana Ferrari , Antonia Ghiselli, Cristina Vistoli
Franco Callegati, Giorgio Corazza, Jordi Mongay, Giovanni Pau, Carla Raffaelli

DEIS Technical Report DEIS-NET-99-003

Dicembre 1999

* *Work under progress carried out with the support of the TERENA TF-TANT task force, in the framework of the EC-funded project QUANTUM, and partially funded by MURST in the framework of the project "Techniques for quality of service guarantee in multi-service Telecommunication Networks" .*

A QoS Measurement-Based Analysis of the Expedited Forwarding Per Hop Behaviour

T.Ferrari [*], A.Ghiselli[*], C.Vistoli[*]
F.Callegati[^], G.Corazza[^], J. Mongay [^], G.Pau[^], C.Raffaelli[^]

[*]Istituto Nazionale di Fisica Nucleare (INFN-CNAF)
{ferrari |ghiselli | vistoli}@cnaf.infn.it

[^] DEIS, Dipartimento di Elettronica Informatica e Sistemistica, University of Bologna
{fcallegati | gcorazza | gpau | craffaelli}@deis.unibo.it
jmongay@thesis.deis.unibo.it)

Abstract

The differentiated services (diffserv) architecture defines a new framework for the support of quality of service (QoS) in IP-based networks. One of the diffserv packet forwarding treatments, the Expedited Forwarding PHB, is taken into consideration, and its performance is analysed in terms of one-way delay and instantaneous packet delay variation. Within a diffserv node we study the impact of buffering and we compare the effectiveness of several scheduling algorithms for the treatment of EF traffic. Tests are carried out in different network layouts under a range of different configurations and traffic conditions.
The goal of our study is twofold: EF simulation results need to be validated through a real diffserv playground and many implementation-related issues need to be evaluated. Our paper addresses service-to-arrival-ratio of an EF queue, the effect of buffering on delay and jitter at any stage within a diffserv node and the effect of EF aggregation on the end-to-end behaviour.

1. Introduction

Quality of Service (QoS) is the capability of forwarding packets in a differentiated way by grouping packets into traffic categories called classes. The class can consist of a single application microflow or of the aggregation of multiple microflow instances. Differentiated treatments of data can be deployed to create services: a service addresses the needs of applications in terms of bandwidth, delay, jitter, loss probability etc.

Several different solutions to the QoS problem have been devised: ATM (Asynchronous Transfer Mode) [1], RSVP (Resource ReSerVation Protocol) [2], the integrated services [3] and the differentiated [4] services architectures are examples of complementary and interoperable approaches addressing different needs.

The support of QoS requires performance measurement. Whenever an ISP provides QoS-based services to its customers measurement at different levels has to be introduced: in applications to adapt to the current QoS level, by end users to verify the service provided by the network, by operators for monitoring and validation of services.

The performance measurement problem is addressed by the IP Performance Metrics working group [5] at IETF. Measurement means the availability of standard metrics to quantify performance and of tools and universal measurement methodologies.

In this paper we address measurement when applied to the differentiated services architecture, in particular for the support of the Expedited Forwarding Per Hop Behaviour. In our study measurement is deployed as a tool to gain experience with measurement methodologies, to define and validate the implementation of EF in a real diffserv network, and last but not least, to investigate the applicability of the diffserv architecture in a production environment.

The Differentiated Services architecture [4] is based on few fundamental concepts and components: the identification of the packet QoS group through a label and the differentiated treatment of that packet within a diffserv node as defined by the diffserv conceptual model [9].

Per Hop Behaviours (PHBs) define the forwarding behaviour of a given packet in a diffserv node. Several PHBs have been standardised so far: the Expedited Forwarding PHB - for the support of services requiring time guarantees - and the Assured Forwarding PHB groups – for the packet treatment within each group according to three types of drop precedence. PHBs are identified through a 6 bits label, called Differentiated Services Code Point (DSCP) which is placed into the Diffserv Field of the IP header.

The *classifier* is the component which splits an input stream into a set of output streams by means of traffic filters based on the content of packet headers and/or packet attributes which can be implicitly derived .
According to packet arrival times *meters* verify if a packet conforms to a given pre-defined traffic profile. According to the conformance several test result actions (*action elements*) can be performed. Four types of actions (and a combination of them) can be performed: marking, dropping, shaping, mirroring and monitoring.

A *queue* is the storage area needed to perform scheduling and/or shaping. *Scheduling* is the process of deciding at every transmission time which queue among a set of candidates has to be serviced depending on some queue and/or packet properties. Generally it is deployed in case of resource contention at the output. Shaping modifies the input traffic to enforce a given output profile.

Traffic conditioners are deployed at a given stage in the data path to enforce a given policy. They can be implemented through the combination of one or more of the previous diffserv components (classifiers, meters, action elements and queues), or alternatively through the combination of existing traffic conditioners. Marking is an example of traffic conditioning

The *service* is QoS from the point of view of an application. It can be quantitative or qualitative. In the former case the service is specified through a set of metrics and the corresponding target values, while in the latter case only a high-level definition is provided. In both cases the actual implementation is based on a given combination of the above mentioned diffserv components. This makes the diffserv architecture very flexible.

2. Objectives

According to RFC 2598 [12], EF traffic “*can be used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through the domains*.”. EF is the PHB which explicitly defines time constraints and which can be deployed for the support of delay and/or jitter sensitive applications. Our work focuses on EF, in particular we deploy QoS metrics for the study of the definition, optimisation and validation of the EF implementation both in the local and in the metropolitan area.

We aim at investigating some of the issues related to the design and implementation of EF-based services, like:

1. the identification of appropriate scheduling algorithms for the support of EF traffic,
2. the definition of the optimum service-to-arrival-ratio value for the minimisation of queuing delay and jitter,
3. the effect of buffering at different stages into a single diffserv node on the time properties of an EF stream,
4. the impact of EF traffic aggregation on end-to-end traffic profile,
5. the definition of some guidelines in the design of a core diffserv network for the support of the EF PHB, e.g. resource over-provisioning in the core versus the implementation of traffic conditioning in some diffserv nodes of the core on the data path.

The ultimate goal is the identification of the main issues related to the support of EF so that time guarantees can be effectively provided in a broad range of scenarios, i.e. independently of the EF traffic profile, of the EF aggregation degree and of the instantaneous background traffic volume. The ultimate goal is the validation of the EF PHB.

3. QoS Measurement

One-way delay [6] and instantaneous packet delay variation (ipdv) [7] are the two metrics deployed in our work. The average and the behaviour over time have been considered. In this study the definition of the two metrics and the measurement methodology comply with the standardisation efforts of the IP Performance Metrics (IPPM) working group at IETF [5].

EF traffic requires low packet loss probability and guaranteed bandwidth: In this study the assumption is that scheduling provides good traffic isolation, i.e. no EF packet loss is observed and EF bandwidth guarantees are effectively supported independently of the instantaneous background traffic volume [15] .

3.1 Metrics

3.1.1 One-way Delay

One-way delay is defined in RFC 2679 [6] and corresponds to the definition of parameter **Type-P-One-way-Delay**: "*for a real number dT, >>the *Type-P-One-way-Delay* from Src to Dst at T is dT<< means that Src sent the first bit of a Type-P packet to Dst at wire-time* T and that Dst received the last bit of that packet at wire-time T+dT* ".

3.1.2 Instantaneous Packet Delay Variation

Instantaneous packet delay variation is computed according to metric **Type-P-One-way-ipdv-jitter**, which is based on **Type-P-One-way-ipdv** as defined by the IPPM working group [7]: "*Type-P-One-way-ipdv is defined for two (consecutive) packets from Src to Dst, as the difference between the value of the type-P-One-way- delay from Src to Dst at T2 and the value of the type-P-One-Way-Delay from Src to Dst at T1.*"
Type-P-One-way-ipdv-jitter is computed according to the following formula:

$$\text{Type-P-One-way-ipdv-jitter} = | \text{Type-P-One-way-ipdv} |$$

In our tests we assume that the drift of the sender clock and receiver clock is negligible given the time scales of the tests discussed in this article.

In the following we will refer to Type-P-One-way-ipdv-jitter simply with *ipdv*.

3.2 Precision and Synchronisation

Generally speaking, inter packet gaps can be computed by a given host in software, for example by recording packet timestamps. Precision of this measurement in general depends on the skew and on the resolution of the sender and receiver's clock, and on the difference (if any) between wire and host time – which is the main source of error if present.
Different synchronisation methods can be adopted according to the precision required: the Network Time Protocol and GPS are two examples. Alternatively a single measurement point with two or more interfaces can be deployed so that traffic is originated by a given interface and received on a different one. These approaches will be discussed in the following.

One-way delay measurement is directly affected by synchronisation errors and therefore synchronisation between the source clock and the destination clock is required.

The ipdv is obtained by means of on one-way delay measurements. Nevertheless, since it is a differential measurement it does not require synchronisation of the source and destination clocks, in the assumption that skew and drift [7] of both clocks are negligible in the inter-packet gap time scale. It can be demonstrated that given two subsequent packets P_i and P_{i+1} ipdv can be computed as the absolute value of the difference of the inter packet gap at the sender int_S and the inter packet gap at the receiver int_R:

$$ipdv = | int_S - int_R|$$

3.2.1 NTP

The Network Time Protocol [16] provides the mechanism to synchronise time and co-ordinate time distribution in a large network operating at different speeds. NTP is based on server hierarchy through which timekeeping information is distributed. A number of local network hosts or gateways may act as secondary servers running NTP with one or more primary servers. The secondary servers distribute time via NTP to remaining hosts on local network. In the NTP terminology the accuracy of each server is defined by a number called the stratum, with the topmost level (primary servers) assigned as one.

An NTP server hierarchy based on three stratum-1 servers with GPS clock source was configured in a wide area test network to verify the precision achievable over dedicated links[1]. For each test site a stratum 1 server provides time to the local router, which then distributes this information locally to workstations running NTP version 3 under Solaris 2.7.

According to our tests NTP precision in clock synchronisation is in the order of a few msecs. The clock offset does not vary linearly because of the frequent external adjustments automatically performed by NTP and because of the designated server, which can change over time. For this reason we can conclude that NTP can be deployed as a synchronisation tool only when the uncertainty it introduces is negligible in comparison to one-way delay, which is not the case in high speed WANs and in the local or metropolitan area.

[1] Two streatum-1 servers were by Advanced Systems [17] while the third was provided by RIPE [18].

3.2.2 GPS

This solution requires the deployment of specialised hardware both on the sending and receiving side to get the clock signal from a GPS antenna. GPS is deployed as primary source of timestamps for data packets. The accuracy in clock synchronisation with GPS is in the range [100, 1000] nsec.

3.2.3 Loop

Packets are sent and received by a single node and traverses several diffserv-capable nodes. This test methodology is best suited to test networks in which specific routing can be deployed to build a sort of ring. As figure 1 shows, given a single source/destination measurement point, two different routers have to be deployed locally since the first and the last hop can not coincide. This is the measurement approach adopted for all the experiments presented in this paper.

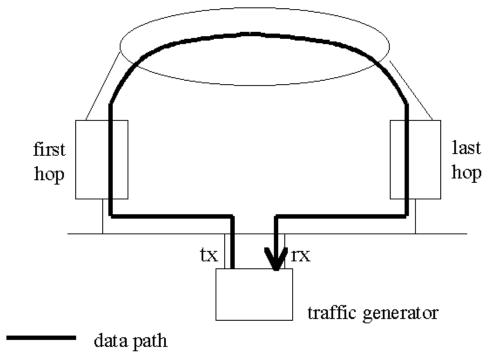


Figure 1: configuration of a data path looped to the traffic source

4. Test Methodology

4.1 Traffic Generators

4.1.1 Host time: mgen

MGEN is a set of tools, which provide the ability to perform IP network performance measurements using UDP/IP unicast and multicast traffic. The toolset generates real-time traffic patterns so that the network can be loaded in a variety of ways. Script files are used to drive the loading patterns over the course of time. It is possible to create script files, which emulate the traffic patterns of unicast and/or multicast UDP/IP applications.

4.1.2 Wire time: SmartBits

In order to improve precision in delay and ipdv measurement, the deployment of specialised equipment capable of implementing time stamping in hardware is recommended. Results presented in this paper were gathered through specialised equipment by Netcom Systems. SmartBits 200 provides an accuracy of 100 nsec. Application SmartWindows v.6.21 was deployed for both one-way delay and ipdv measurement.

4.2 Measurament

In this work one-way delay computation is derived from cut-through latency measures collected by the SmartBits 200 according to RFC 1242 [8]. The standard defines cut through latency as: "*the time interval starting when the end of the first bit of the input frame reaches the input port and ending when the start of the first bit of the output frame is seen on the output port*". One-way delay can be computed from cut-through latency by adding the transmission time of the packet, which is constant for a given packet size, as shown in figure 2.

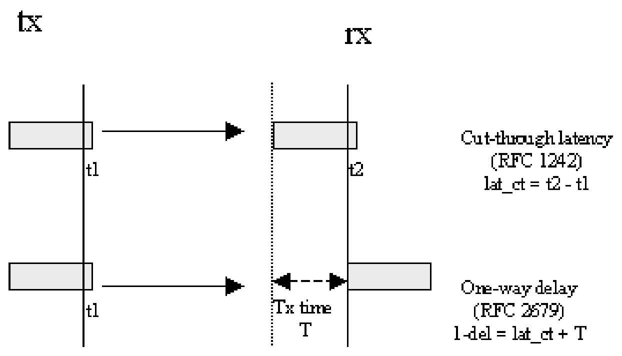


Figure 2: relationship between cut-through latency and one-way delay.

Type-P-One-way-ipdv-jitter [6] is the second metric deployed in this study. It is computed from one-way delay estimations as provided by the methodology described above.

Both one-way delay and ipdv are evaluated in terms of variation over time and average. For the first metric time is represented through the sequence number of received packets[2].
Average is computed over a set of sample means. One sample consists of a single packet.

4.3 Traffic scenarios

The number of streams can vary. Depending on the test a single EF behaviour aggregate or a combination of EF and best-effort (BE) traffic was deployed.
The impact of background traffic on EF streams is analysed for different packet sizes to quantify the difference between the EF ideal behaviour (the traffic profile in case of no resources contention) and the actual behaviour in case of congestion. The EF and BE packet sizes can be a key parameter in one-way delay computation since they affect the interleaving between BE and EF data units in transmission queues.

In each test scenario packet classification and marking are implemented at the edge i.e. at the ingress interface of the first-hop router. Packets are then distributed to different queues in the scheduling module according to the packet DSCP. Diffserv functional blocks (classification, metering, marking, policing and scheduling) where enabled only in the first hop to study the effect of diffserv configuration on EF behaviour within a single diffserv node.

In this study we cover the following range of scheduling algorithms:
- Weighted Fair Queuing,
- Self Clocked Fair Queuing,
- Priority Queuing.

[2] Time is not represented as *absolute time* because in our test set-up the traffic generator does not provide the time at which a given packet is received: for each packet only the corresponding delay measure is plotted. However, one-way delay samples can be distributed over time so that for each interval of known length a single packet measure is available (1 msec is the minimum granularity). In this way the gap between two subsequent measurement points can be estimated.

For each scheduling algorithm EF queue depth and EF queue weight were deployed as test parameters.

For both scenarios described in paragraph 5.1 two unidirectional streams - an Expedited Forwarding one and a best-effort one - were run in parallel.
The SmartBits 200 is deployed both as source and receiver of EF traffic. Data is generated by a FastEthernet interface and received by a second identical card at a constant rate equal to 300 Kbps. The best-effort stream consists of 200 pack/sec with payload size equal to 1000 bytes. Best-effort traffic is generated among test workstations; both are Sun Ultra running Solaris 2.7.
FastEthernet was deployed in *duplex* mode to avoid collision effects on measurements.

5. EF: One-Way Delay and Instantaneous Packet Delay Variation

Ideally in a diffserv node EF traffic should be independent of the amount and type of background traffic. In practise this is not possible for a number of reasons.
In the first place, EF packets are subject to queuing in the scheduler queues and this queuing delay is a function of the service time defined by the scheduling algorithm.
In addition, packets issued by a scheduler are collected into a single transmission queue. This implies that high priority traffic has to wait in the transmission queue until all the lower priority packets placed at the head are transmitted. Paragraph 5.2 and 5.3 compare the ideal treatment against the behaviour observed under congestion and without tuning of buffering and scheduling.

Paragraph 5.4 addresses the problem of EF buffer dimensioning: In principle, EF buffers should always be empty in order to keep queuing delay low. This can be enforced through proper over-estimation of the EF queue service rate. However, the presence of an output rate higher than the input one can produce bursts of EF packets out of a set of well-shaped EF streams, even if they are SLS-compliant. The maximum length of such bursts is equal to the EF buffer size at the egress interface of the edge router[3]. Burstiness poses the problem of shaping at boundaries between diffserv domains.

Another key EF implementation issue is the tuning of the service-to-arrival-ratio as defined in RFC 2598 [12]. The service rate of an EF queue has to be a function of the arrival rate, which can be derived from the network configuration. This issue is relevant for that scheduling algorithms in which the departure rate is a function of the EF queue weight like WFQ. The frequency at which an EF queue is serviced is a fundamental property since it impacts the queuing time of a packet. This issue is analysed in paragraph 5.5.

Different queuing algorithms can be adopted for the support of PHBs. The decision of the right solution is a diffserv implementation issue and paragraph 5.6 compares the performance of an EF stream under two different scheduling algorithms: Weighted Fair Queuing and Priority Queuing.

5.1 Network Layout

Several diffserv platforms are analysed to evaluate a broader spectrum of technical approaches, namely the IBM 2212 router and the CISCO 7200 router. In the former case diffserv is deployed on PPP interfaces, while in the latter on ATM. The goal is to evaluate the implementation of diffserv from different perspectives, not to draw comparisons given the differences in router achitectures and network scenarios deployed in the two cases.

[3] The burst propagates through the diffserv domain and will need shaping at the first boundary. Since shaping requires buffers, the optimum shaper configuration can be expressed as a function of the EF queue size at the edge.

The network layouts deployed in the two cases are illustrated in figure 3 and 4.

5.1.1 Scenario 1

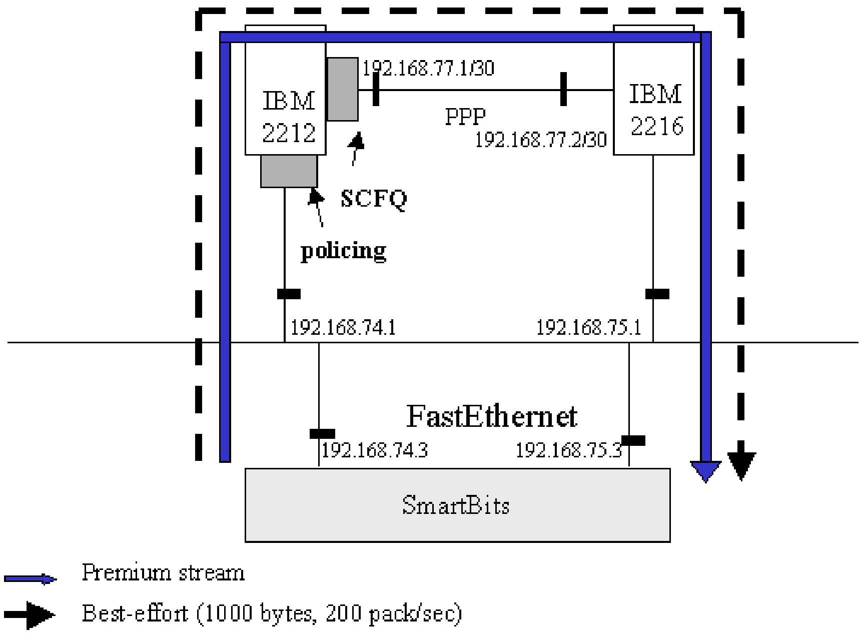


Figure 3: test layout in the local area

In IBM routers bandwidth guarantees are enforced through a memory allocation scheme in which given a traffic class the amount of memory reserved to it is proportional to its link capacity share. Input EF traffic is controlled by a two-parameter token bucket policer, whose burst size is configurable. EF packets are classified and marked. At the egress point of the edge router EF traffic is subject to queuing. The scheduling algorithm deployed is a type of Weighted Fair Queuing called Self-Clocked Fair Queuing. Both the EF queue depth and its weight (its default value is 90%) can be configured. SCFQ is enabled on the egress interface of the IBM 2212 (PPP interface) as illustrated in figure 3.

The IBM 2216 is transparent from the QoS point of view, i.e. no QoS features is enabled on that router.

The operating system version 3.3 was deployed on the IBM 2212 deployed in these tests is the 3.3. The default EF configuration was deployed with the only exception of the premium buffer size parameter. The premium queue weight is also kept constant (90% is the default value).

5.1.2 Scenario 2

In scenario 2 (figure 4) the testbed spans a metropolitan area. The network is based on ATM CBR connections whose bandwidth was set to 1 Mbps or 2 Mbps depending on the test. Diffserv functional blocks (classification, marking, metering and scheduling) are enabled only in the first hop router (a CISCO 7200). This second scenario is completely based on CISCO platforms, but we assume that the two CISCO 7500 are transparent, i.e. they do not modify the EF profile sent out by the edge diffserv node. The deployment of a network layout based on a single router platform is part of our future test programme

Scenario 2 was deployed to compare the EF behaviour under different scheduling algorithms (WFQ and PQ) and to perform tuning of the transmission queue.

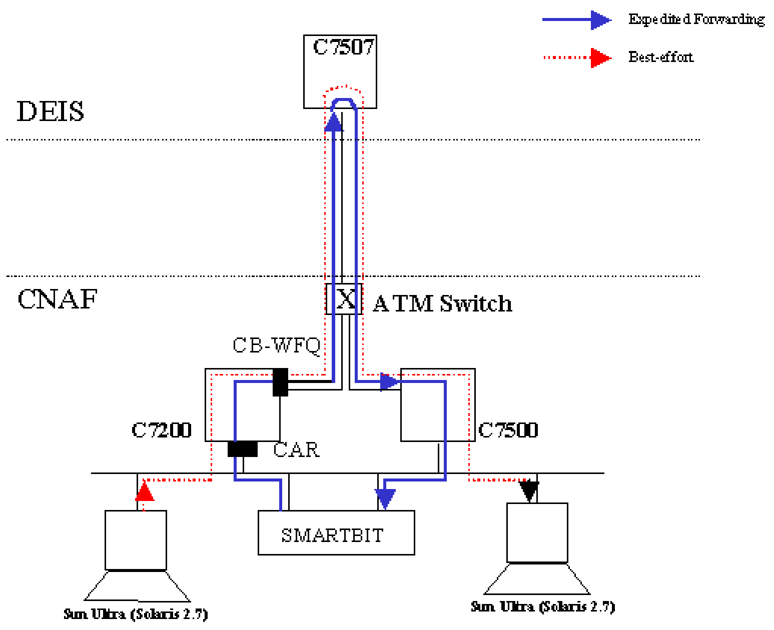


Figure 4: layout of test in the metropolitan area

5.2 EF Traffic without Resource Contention

In this test the EF stream consists of a well-shaped UDP flow at a constant bit rate equal to 300 Kbps. EF packets are processed by an MF classifier based on the source/destination address and on the protocol type. As figure 5 shows, without best-effort traffic average one-way delay is a linear function of the EF packet size, in fact, without resource contention the main delay component is represented by transmission time for any router architecture.

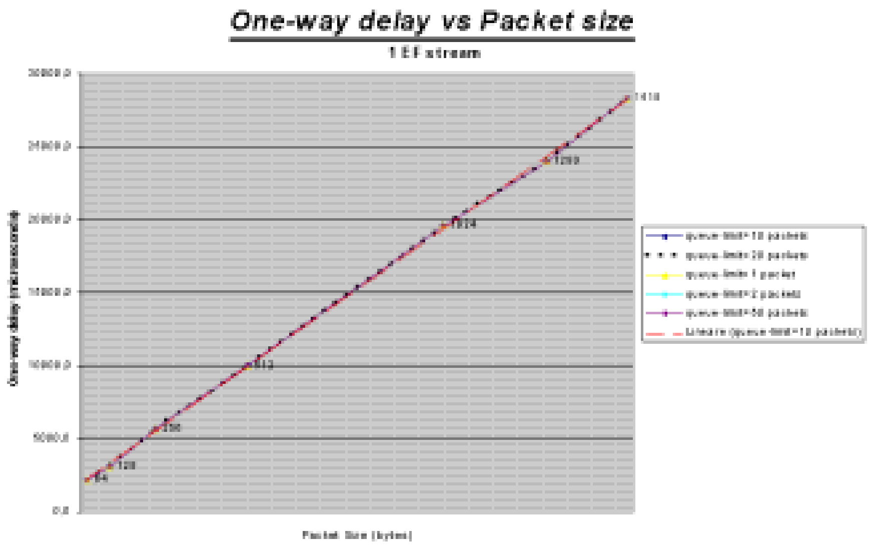


Figure 5: average one-way delay for different EF packet sizes
without best-effort traffic (CISCO 7200). One-way delay is a linear function of the packet size.

One-way delay is almost constant in time for any EF packet size and any EF queue length, as figure 6 shows. The average instantaneous packet delay variation depends on the packet size but is almost negligible, i.e. well below 1.15 msec.

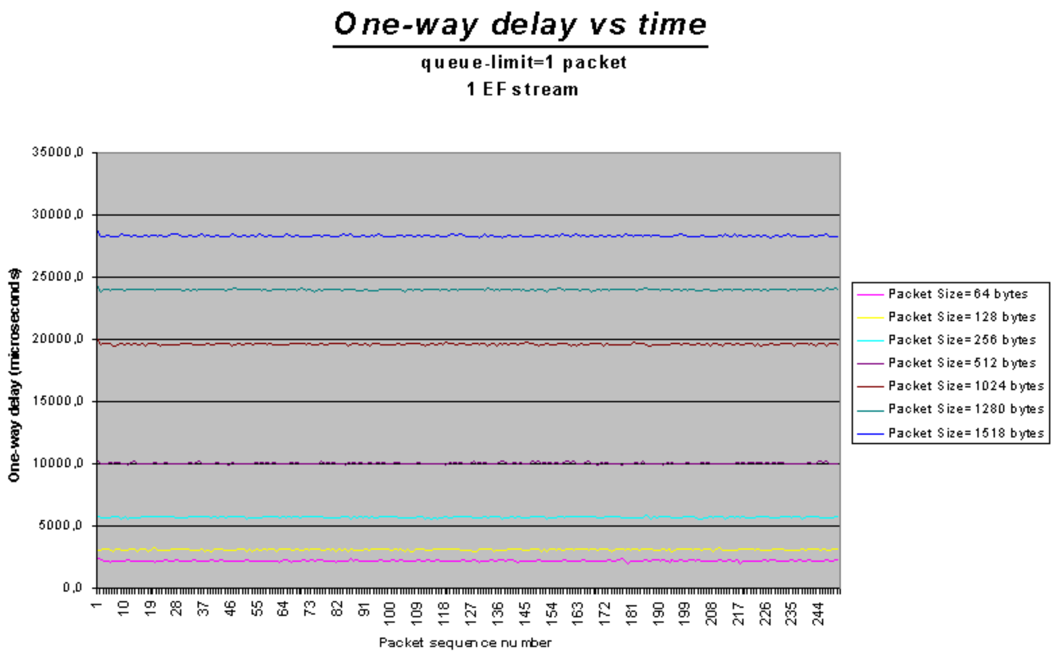


Figure 6: one-way delay over time for different EF packet sizes (scenario 2, CISCO 7200).

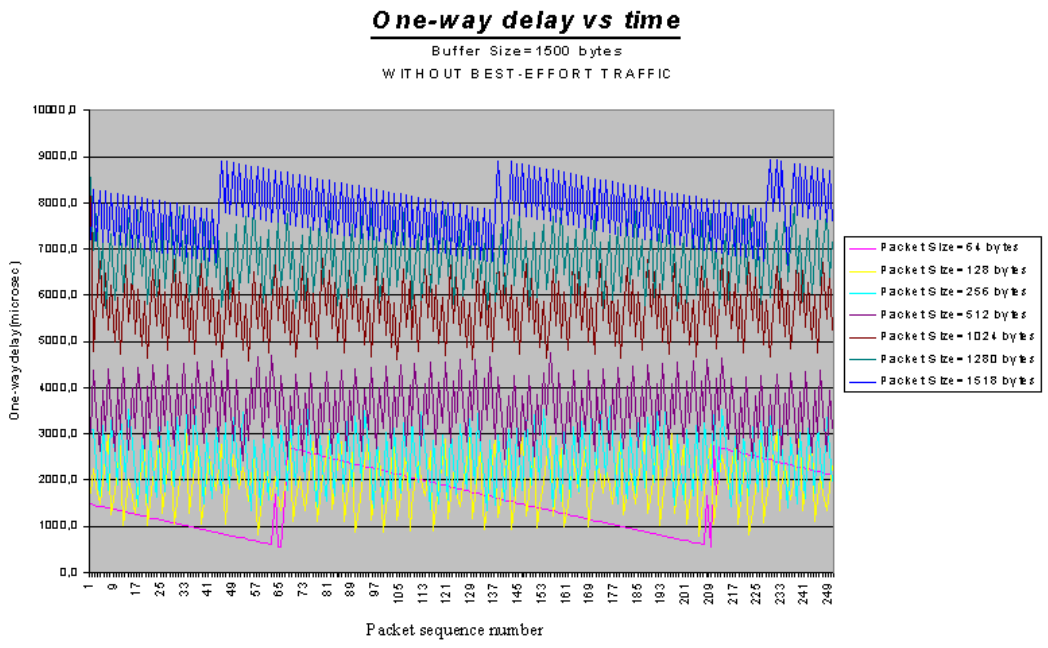


Figure 7: one-way delay over time for different EF packet sizes
(scenario 1, IBM 2212)

However, one-way delay can depend on the scheduling implementation adopted in a diffserv node. By repeating the equivalent of tests above in scenario 1 on the IBM 2212, we discovered that depending on the implementation even in an ideal scenario the variation in one-way delay can be of some milliseconds[4]. Results in figure 7 refer to one EF stream at 300 Kbps and to an EF buffer size of 1500 bytes. The EF service rate is 300 Kbps (departure rate = arrival rate).

Not only one-way delay varies, but fluctuations can be periodic: as figure 8 shows two cycles can be identified.

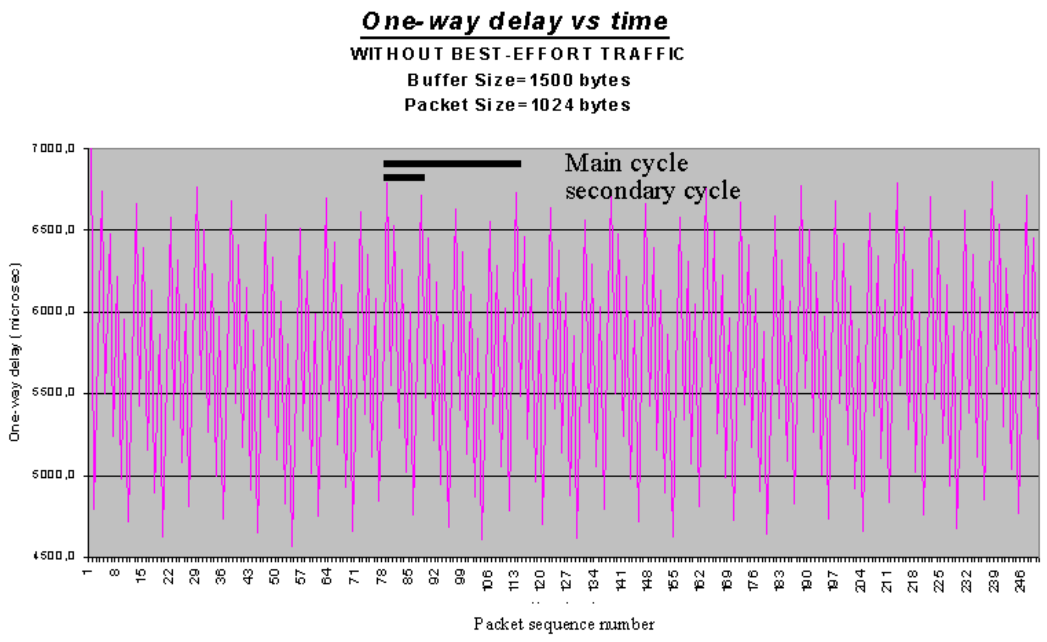


Figure 8: one-way delay over time for a packet size of 1024 bytes (scenario 1, IBM 2212)

[4] The study of the variation of one-way delay on IBM 2212 platforms is a subject of current and future study.

5.3 EF Traffic with Resource Contention

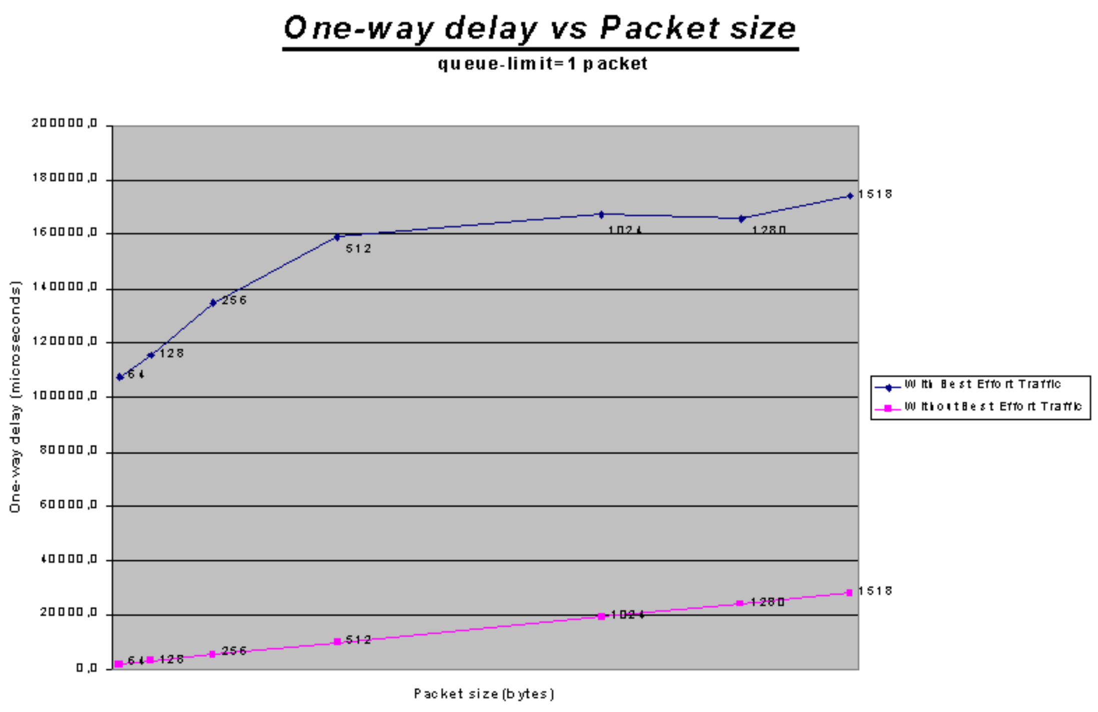


Figure 9: average one-way delay for different packet sizes with and without best-effort traffic. WFQ is the scheduling algorithm and the departure rate is equal to the arrival rate. The length of the transmission queue is 40 units of 512 bytes (scenario 2, CISCO 7200).

In a more realistic scenario the EF traffic profile is necessarily influenced by the presence of other behaviour aggregates. In order to quantify this effect we deployed two classes: an EF class and a best-effort class.
Two constant bit rate streams, one per class, are generated. The EF stream (300 Kbps of IP packets) complies with the EF contract while the best-effort stream congests the line. The best-effort traffic volume is 1.65 Mbps of IP data[5] - 2 Mbps if ATM overhead is included - so that the aggregate exceeds the line capacity (1 Mbps in this test)[6].

According to our tests the effect of background traffic depends on three factors:

- transmission queue depth
- service-to-arrival-ratio
- EF scheduling algorithm.

Figure 9 compares one-way delay with and without best-effort traffic under Weighted Fair Queuing in a worst case scenario. The EF departure rate is equal to the arrival rate (300 Kbps), i.e. no service

[5] 200 pack/sec, where one packet is 1000 bytes long.
[6] The deployment of just 2 CBR streams of packets with constant size was chosen on purposeis to keep our analysis simple. In order to reproduce more realistic test scenarios several classes with a greater variety of application streams have to be deployed. This is part of our future test programme.

rate over-provisioning is applied. In addition, the transmission queue depth at the egress interface is set to its default value (40 units of 512 bytes), while the EF queue (parameter *queue-limit*) is constant and equal to 1 packet. The capacity of the egress ATM connection is 1 Mbps against a input traffic aggregate (EF plus best-effort data) of 1.95 Mbps (2.30 Mbps or more[7] when including ATM overhead). The variation depends on the packet size, in the worst case it can be up to 150 msec.

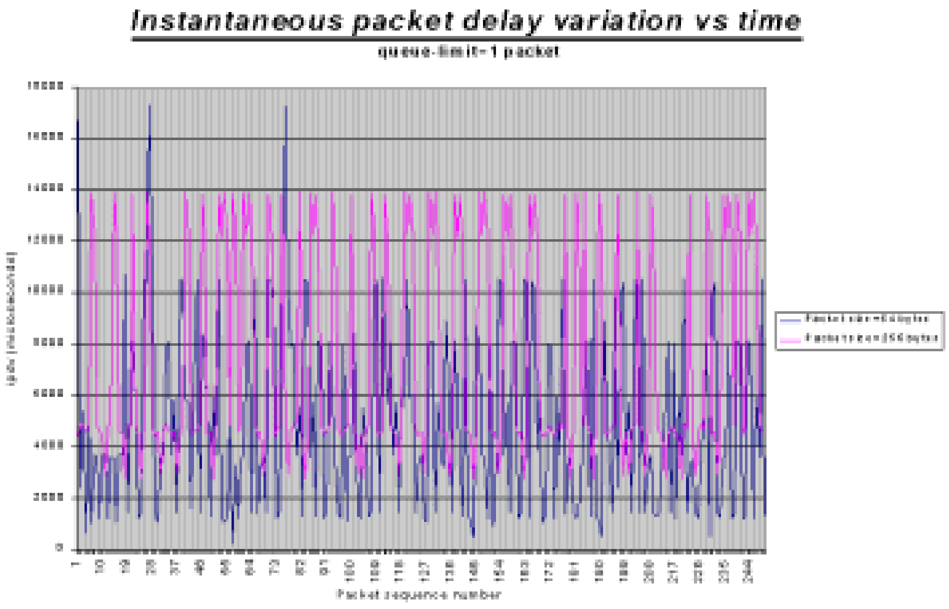


Figure 10: ipdv over time for packet sizes of 64 and 256 bytes

The average ipdv depends on the packet size. It is a function of the height and of the frequency of ipdv peaks, which vary with the packet size as curves in figure 10 and 11 show. For packet sizes of 1024 and 1518 bytes, the peak exactly corresponds to the transmission time of 1 BE packet of 1000 bytes at 1 Mbps (ATM overhead included), which is equal to 9.33 msec.

The average ipdv depends on the packet size. It is a function of the height and of the frequency of ipdv peaks, which vary with the packet size as curves in figure 10 and 11 show. For packet sizes of 1024 and 1518 bytes, the peak exactly corresponds to the transmission time of 1 BE packet of 1000 bytes at 1 Mbps (ATM overhead included), which is equal to 9.33 msec.

[7] 300 Kbps do not include ATM overhead (in scenario 2). So the exact EF rates depends on the EF packet size.

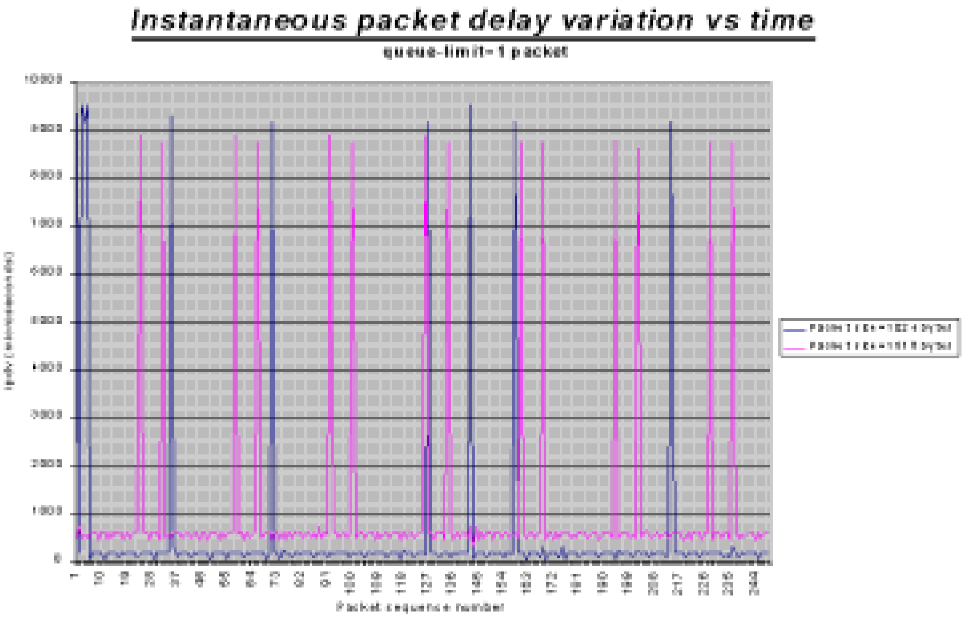


Figure 11: ipdv over time for packet sizes of 1024 and 1518 bytes

5.4 EF Buffer Dimensioning

5.4.1 EF buffer

Even with best-effort traffic the EF buffer depth is irrelevant in terms of one-way delay and ipdv. If WFQ is properly configured, i.e. if the departure rate is equal or bigger than the short-term arrival rate, then the maximum instantaneous EF queue depth is 1 packet. This is confirmed by tests in both scenario 1 (under Self Clocked Fair Queuing, IBM 2212) and scenario 2 (WFQ, CISCO 7200).

5.4.2 Transmission queue

With term *transmission queue* we define the storage unit which collects data from the scheduler and sends packets on the line. While its implementation can vary depending on the diffserv node platform, normally its service policy is FIFO and it serialises the transmission of packets from different classes.
The transmission queue size can have a great impact on EF one-way delay as queuing delay depends on the number and size of lower priority packets located at the head of the queue.
Tests were carried out in scenario 2 (CISCO 7200) by keeping BE and EF traffic volumes constant and by gradually increasing the size of the transmit queue[8]. As figure12 shows, one-way delay increases linearly with the depth of the transmission queue size (expressed in units of 512 bytes). One-way delay can vary greatly from a few tens of milliseconds when the size is equal to 5 to more than 1 sec if the queue size is 500 particles.

[8] According to the queuing architecture of CISCO routers the transmit queue is called *transmit ring* and its size is set by parameter *tx-ring-limit*. The transmission queue length is expressed in *particles*, i.e. in units of 512 bytes.

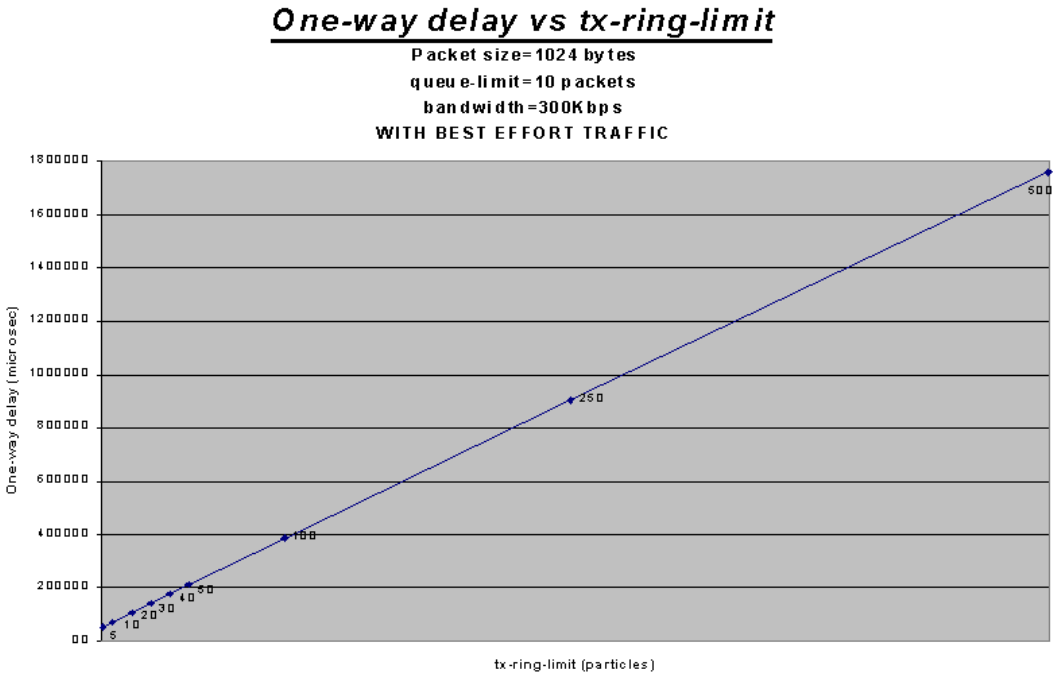


Figure 12: relationship between average one-way delay and transmission queue size (tx-ring-limit) for EF packets of 1024 bytes

On the other hand, ipdv is not influenced at all. As figure 13 shows, variations are always within 9.5 msec, a time interval which corresponds to the transmission time of 1 BE packet of 1000 bytes at 1 Mbps when the ATM overhead is taken into consideration. One-way delay varies periodically, where the period is a function of the ATM PVC bandwidth. In this test periodicity is due to the highly regular pattern of both EF and best-effort traffic, i.e. to the regularity of EF and best-effort data units.
For each transmission queue length EF packets always experience a minimum one-way delay which depends on the ring size. Then, an additional amount of delay can be present: it varies between 0 and two times the transmission time of a BE packet.
For example, let's consider a transmission queue length of 10 particles. During intervals CD and EF the amount of additional delay is a fraction of the transmission time of a BE packet. Delay varies between 0 – when the EF packet arrives right after the end of the transmission of a BE packet – and the transmission time of a whole BE packet – when it arrives at the very beginning of its transmission. On the other hand, during interval GH an additional delay component equal to the transmission of 2 particles can be present due to the fact that if a BE packet is queued but only one free particle is present, then the packet is placed into the transmission queue anyway. Since a BE packet is of 1028 bytes, 3 particles are needed to store a single BE packet. This means that the maximum instantaneous transmission queue length of the ring is X + 2 particles where X is the value set by configuration.
Given two constant bit rate streams within a period delay can increase or decrease depending on the fractional part of ratio *R* between BE packet rate and EF packet rate. If R < 0.5 delay decreases, while if R > 0.5 it increases. For R = 0.5 the delay increment can alternatively be 0 or ½ of the transmission time of a BE packet. In figure 13 the period length is constant (an integer multiple of 30 packets) since ratio R is the same for any transmission queue.

The step size in the saw-tooth curves depends on the transmission time of a BE packet, which in our case is a function of the capacity of the ATM PVC.

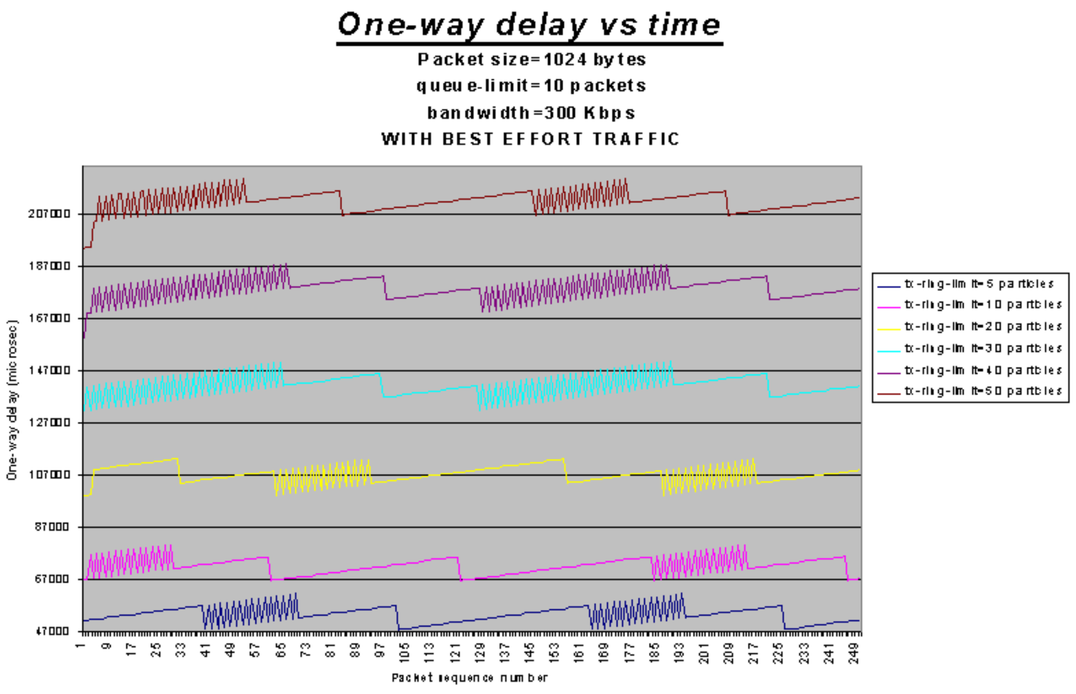


Figure 13: one-way delay variation over time. For each transmit queue length periods are integer multiples of 30 packets

5.5 EF Service-To-Arrival-Ratio with WFQ

Given the EF queue service rate S_r (expressed as percentage of the line rate), the link rate l_r and the EF arrival rate A_r, RF 2598 defines parameter *Service-To-Arrival-Ratio* (STAR) as:

$$S_r * l_r = A_r * STAR$$

STAR has to be tuned so that delay and jitter due to EF queuing are minimised.

In this test we verify the impact of parameter STAR on one-way delay and ipdv for different packet sizes in scenario 2. STAR varies in the range [1, 5]. Value 5 means that the EF service rate is 5 times the EF arrival rate. As usual, two streams are run in parallel: a 300 Kbps EF stream and a 1.65 Mbps (2 Mbps if ATM overhead is included) of best-effort traffic. Transmission queue is 5 particles to minimise delay, according to the results from the previous paragraph.

Service rate can have a great impact for large EF packets, as figure 14 shows: The decrease with a higher service rate can be up to 25 msec. For smaller EF packets the benefit is less evident for the WFQ formula deployed to compute the service time of a packet:

pack_time = time + length * weight

So, the smaller the EF packet size the higher is the chance that an EF packet is scheduled before a BE packet. In addition, since the overall EF rate is constant, if the EF packet size decreases, then the packet rate increases and the number of BE packets interleaving EF packets is smaller. This means that more EF packets can arrive during the transmission time of a BE packet.

STAR values greater or equal to 4 do not improve queuing delay.
If START is equal to 1 then the variation in one-way delay is greater as figure 15 shows. In this case the probability that up to 2 BE [9] packets are sitting in the tx queue in front of an EF packet is higher.
Our tests confirm the simulation results reported on in RFC 2598.

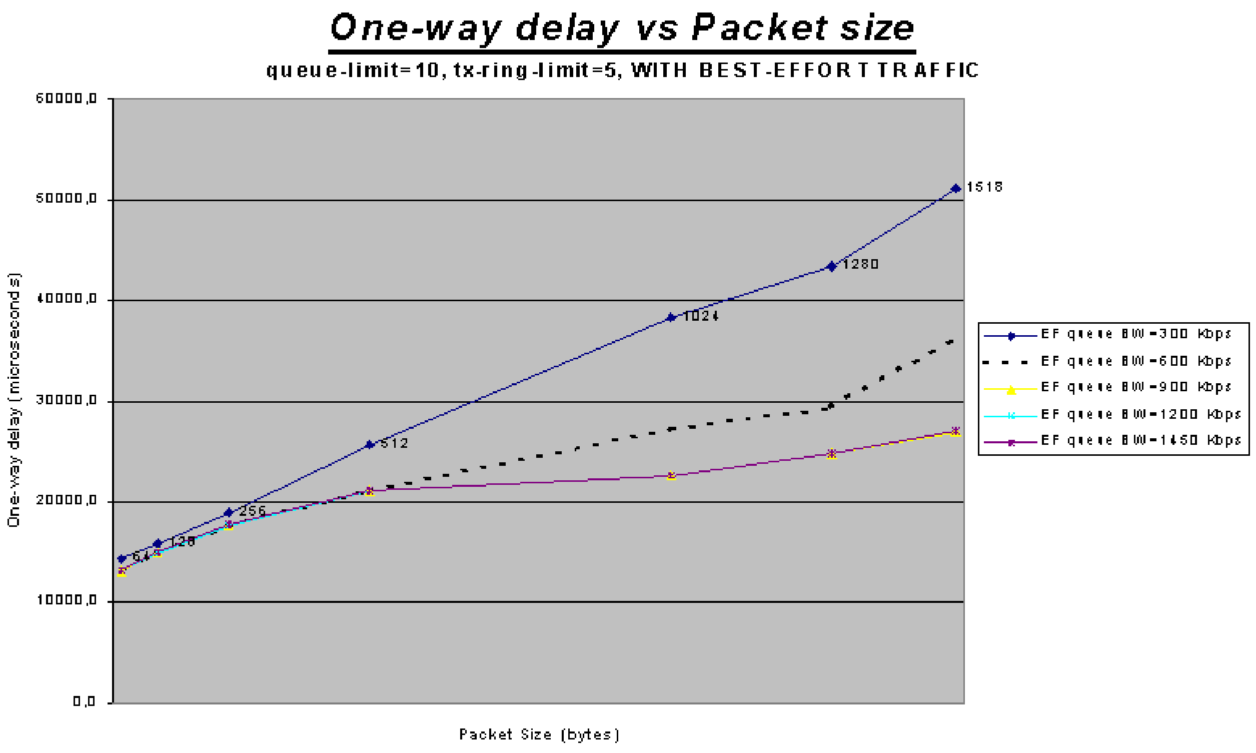


Figure 14: average one-way delay for different service rates and different packet sizes (scenario 2)

[9] The transmission queue lenght is 5 particles so at any time it can store up to 2 BE packets.

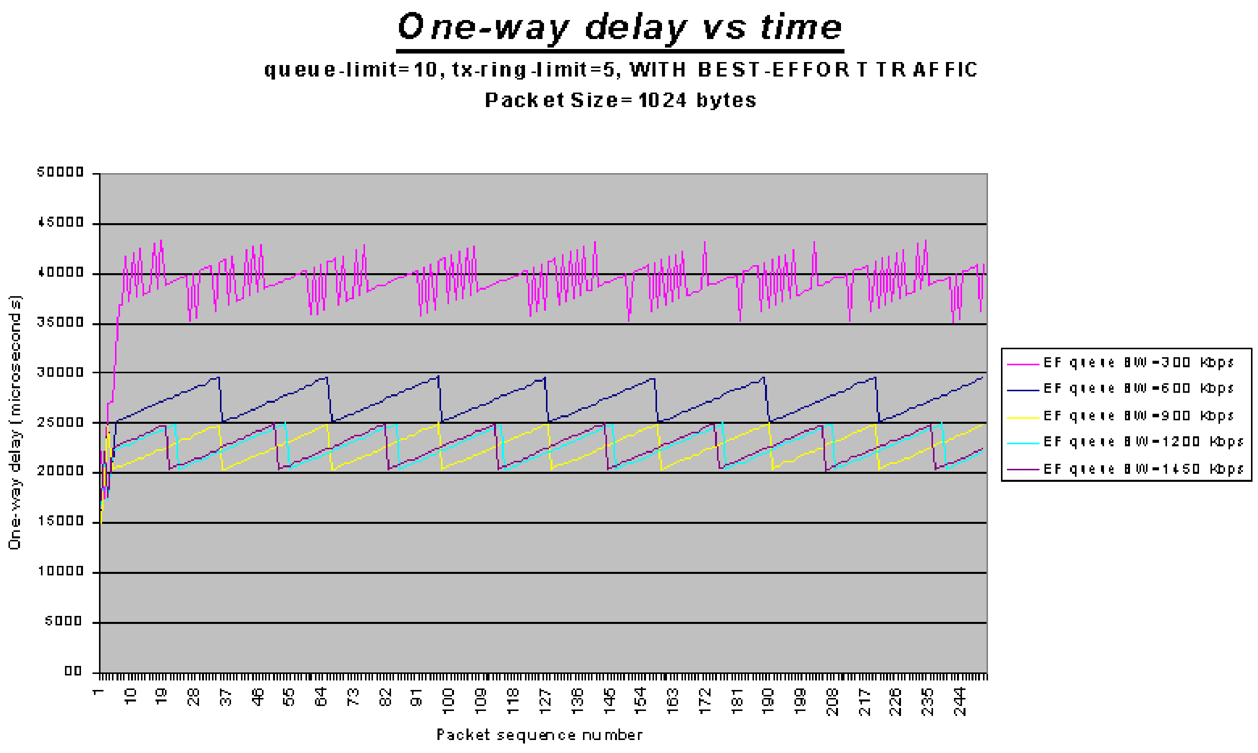


Figure 15: one-way delay over time (scenario 2)

5.6 Weighted Fair Queuing versus Priority Queuing

A lot of different scheduling algorithms have been described in literature [20-27]. The choice of the most suitable one is of great importance given the considerable impact that the algorithm can have on one-way delay.

With Weighted Fair Queuing (WFQ) any class is guaranteed with a minimum rate and the service time of a packet depends on its length and on the weight of the queue it belongs to. On the other hand, with Priority Queuing (PQ) [21,22,23] a timely packet delivery is guaranteed, the priority queue is always serviced before any other queue. This means that at any time if a priority packet arrives, the priority queue will be serviced next. Priority queuing can cause starvation if priority traffic is not policed properly.

According to the general PQ architecture each packet is associated with a priority level. With N priority levels where priority is proportional to the number, the scheduler serves a priority K packet only if priority queues K+1, K+2, …., N are empty. This scheme is also called *multilevel priority with exhaustive service*. A scheduler can have an arbitrary number of priority levels depending on the number of delay classes that the network operator wants to support. In a priority queuing scheme the higher priority traffic volume impacts both delay and available bandwidth of lower priority packets.

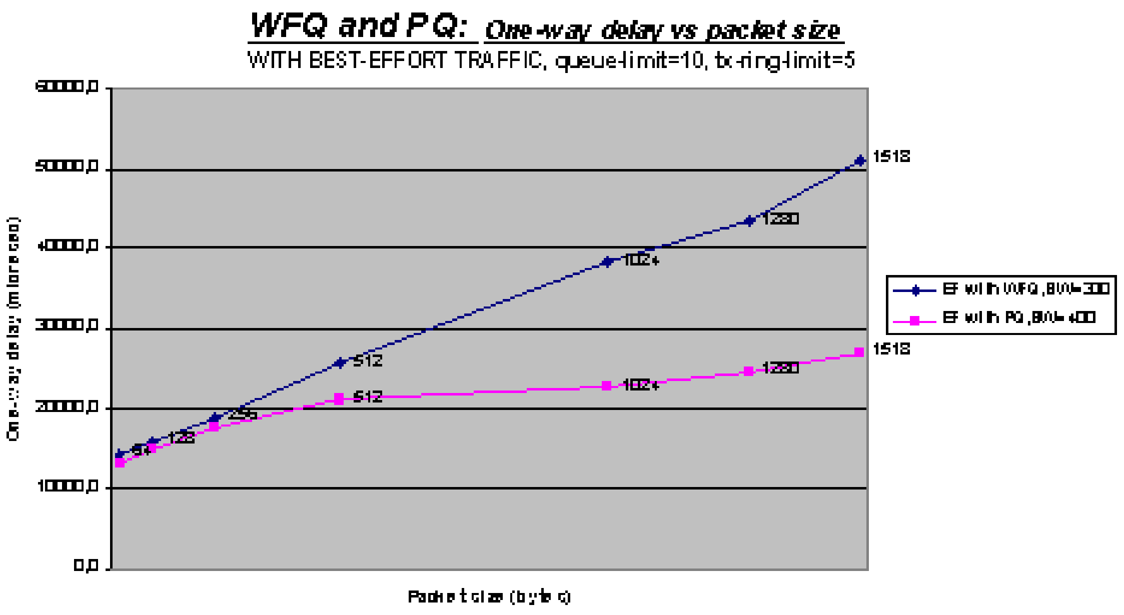


Figure 16: average one-way delay for different EF packet sizes with WFQ and PQ (scenario 2)

Priority queuing applied to EF traffic guarantees lower queuing delay, since the maximum EF queuing time is equal to the transmission of an MTU BE packet. Figure 16 compares delays with WFQ and PQ, where the EF service rate under WFQ is equal to 400 Kbps against 300 Kbps of EF traffic. The benefit is greater for larger EF packet sizes, since a greater number of BE packets can interleave subsequent EF datagrams. Results show that under PQ with MTU EF packets the decrease in one-way delay can be greater than 24 msec.

For large EF packets PQ also minimises EF queuing jitter since PQ reduces the frequency of ipdv peaks. As figure 17 shows, the step is equal to the transmission time of a BE packet (1028 bytes) at 2 Mbps (the rate of the ATM connections).

However, even with PQ EF profile can still be considerably influenced by background traffic. Figure 18 indicates that after adding best-effort traffic even with PQ one-way delay permanently increases and the difference is in tens of milliseconds. In figure 18 this is indicated by packet 204, when best-effort traffic is added. The contribution of best-effort traffic to the increase in EF one-way delay under PQ is an issue still under investigation.

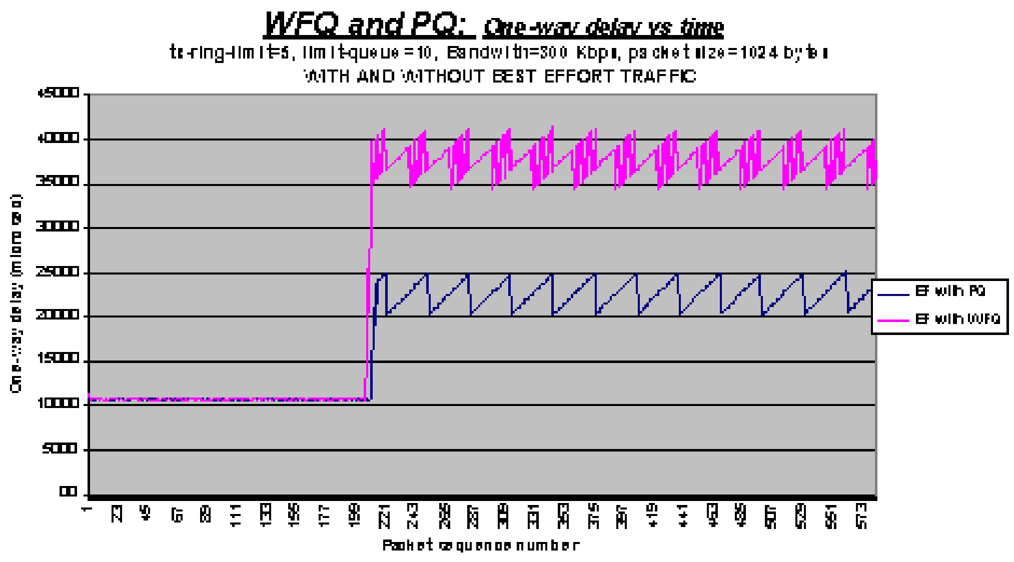


Figure 17: one-way delay over time with WFQ and with PQ for EF packets of 1024 bytes

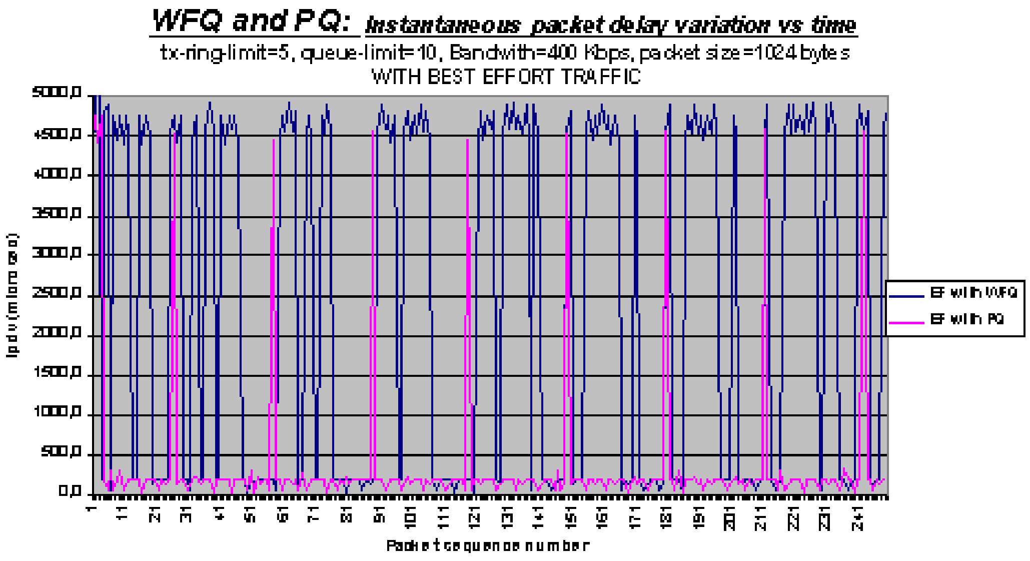


Figure 18: increase in one-way delay with WFQ and PQ when best-effort traffic is added (scenario 2)

6. Conclusions and future work

This paper shows that in order to properly support EF traffic in a diffserv node tuning of both buffering and scheduling is required.

With any scheduling algorithm, the transmission queue size is the fundamental parameter which needs tuning in order to preserve the EF profile as far as possible from the impact of other behaviour aggregates. Several scheduling algorithms can be adopted, but the choice has a great impact on one-way delay. Our study focuses on Weighted Fair Queuing and Priority Queuing: Priority Queuing guarantees lower jitter than Weighted Fair Queuing. Under WFQ a moderate over-estimation of the service rate gives a more timely delivery of EF data, since it decreases on-way delay, while the impact on instantaneous packet delay variation is irrelevant.

EF PHB analysis has to be extended to test it under more realistic traffic scenarios, e.g. with a larger number of sources and with a range of different BE and EF packet sizes and traffic profiles. The effect of EF traffic aggregation on the time properties of a single microflow will be investigated for the support of end-to-end EF-based services. Finally, the study of EF traffic when it is deployed in the wide area along a chain of different diffserv-capable routers is another topic of great interest.

7. Acknowledgements

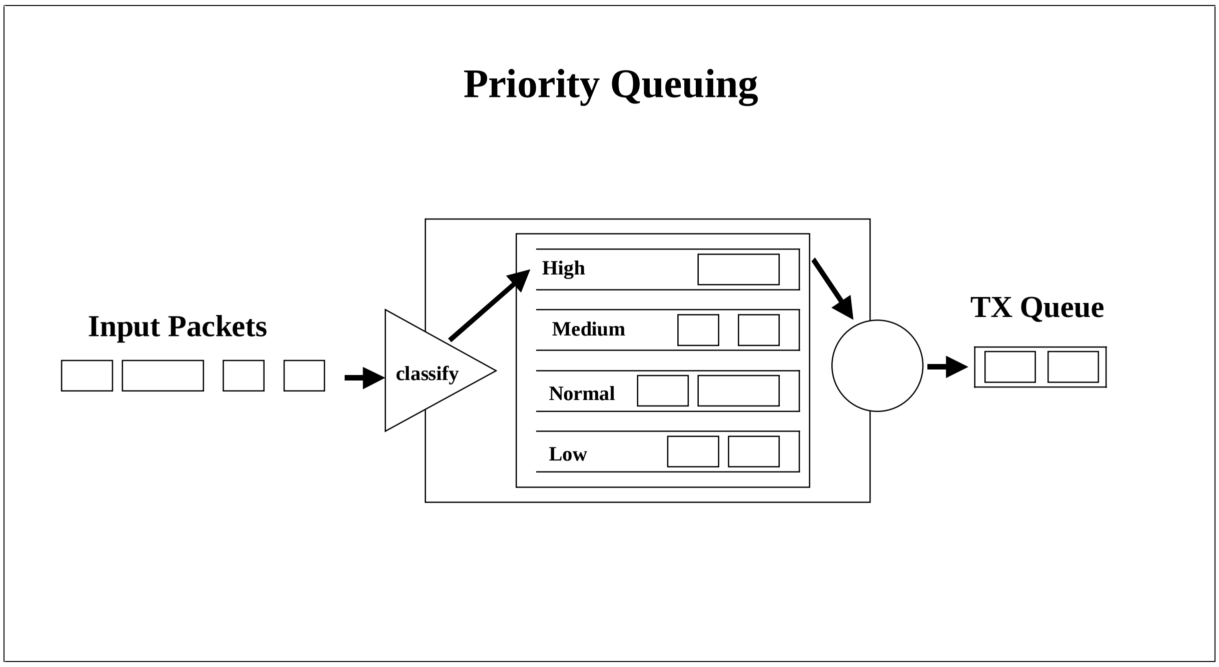
This paper is a joint work of the TF-TANT task force, which we acknowledge for the valuable collaboration provided. We also thank CISCO Systems, IBM and Netcom Systems for the equipment supplied and for the technical support provided during the course of the experiments. We also thank Francis McSwiggan for his work on NTP.

8. Appendix

QoS mechanisms

Priority Queuing

In the priority scheduling scheme, each packet is associated with a priority level. If there are N priority levels, and higher numbered priority corresponds to a packets with higher priority, the scheduler serves a packet from priority level K only if there are no packets awaiting service levels K+1, K+2, …., N. This scheme is also called *multilevel priority with exhaustive service*. A scheduler can have an arbitrary number of priority levels depending on the number of delay classes that the network operator wants to support. In a priority queuing scheme the higher priority traffic increases the delay and decreases the available bandwidth of all lower priority connections. In extreme situations the lower priority traffic may starve because there is always something to send from higher priority levels. In the PQ schema when a queue fills up, additional packets arriving are dropped [1,2]. In order to avoid starvation of lower priority traffic, a policer or shaper may be used to limit the rate of higher priority traffic.
During transmission Priority Queue gives absolute preferential treatment to high priority traffic. In the CISCO implementation packets are classified per protocol (IP, IPX, SNA, …), per incoming interface, per packet size or using access lists. The scheduling mechanism is implemented per output interface (in CISCO routers) [3]. Cisco implements a policing /shaping functionality called CAR (Committed Access Rate), that used with priority queuing may avoid starvation of lower priority traffic.



Priority queuing scheme

Weighted Fair Queuing

WFQ is an approximation of Generalised Processor Scheduling (GPS) that do not make GPS's infinitesimal packet size assumption, and, with variable size packets do not need to know the mean packet size in advance (as WRR). GPS is an ideal (and unimplementable) work-conserving scheduling discipline. GPS serves traffic classes as if they were in separate logical queues, visiting nonempty queue in turn and serving an infinitesimally small amount of data from each queue. Connections can be associated with service weights and they receive service in proportion to this weight. Let us consider a router where N connections with equal weights send data to the scheduler at an infinite rate. The scheduler should allocate each of them 1/Nth share of the bandwidth (because it serves an infinitesimal part of data it achieve this goal). If one source, say source A, sends data slower than its share (1/N) , its queue at the scheduler is occasionally empty and is skipped by the scheduler. Because GPS is Round Robin the time saved when this occurs, is equally distributed to the other connections [1,2,3]

WFQ compute the time (called *finish number* in the WFQ terminology) a packet would complete service had it been served according to GPS, , then serves packets in finishing time order. In practice WFQ simulates GPS and uses the result to determine service order.

To calculate the finish number, the round number, a real value that starts from 0 and increasing at a rate inversely proportional to the number of active connections, is used. A connection is called *active* if the largest finish number of a packet either in its queue or served last from its queue is largest than the current *round number*. If the round number is known the finish number may be calculated as follows. The finish number of packets arriving at inactive connections is equal to the sum of the current round number and the packet size in bit. If packet arrives to an active connection the finish number is the sum of the largest finish number in the queue (or last served) and packet size in bit. The main problem in computing a packet's finish number is determining the current round number of simulated GPS server. Computing the exact round number is hard due the iterated deletion problem. An example best explain the problem. Suppose the round number at time 0 is 0 and has a slope of 1/5. Then we might suppose that at time 5 the round number is 1, our supposition might be wrong because one of the 5 connections active at time 0 might have become inactive in the meantime and the slope increased to 1/4. Deleting inactive connections from the list of active connection require fairly complex computation once every few microseconds. This is the major problem in implementing WFQ in high speed networks [4,5].

9. References

[1] ATM forum; http://www.atmforum.org/

[2] RFC 2205: Resource ReSerVation Protocol (RSVP) – Version 1, Functional Specification

[3] *Integrated Services*; http://www.ietf.org/html.charters/intserv-charter.html

[4] *Differentiated Service*, http://www.ietf.org/html.charters/diffserv-charter.html

[5] *IP Performance Metrics*; http://www.ietf.org/html.charters/ippm-charter.html

[6] RFC 2679: *One-way Delay Metric for IPPM*, G.Almes, S.Kalidindi, M.Zekauskas.

[7] *Instantaneous Packet Delay Variation Metric for IPPM*, C. Demichelis, P.Chimento; ippm draft, work under progress.

[8] RFC 1242: *Benchmarking Terminology for Network Interconnection Devices*; S.Bradner.

[9] *A Conceptual Model for Diffserv Routers*, Y. Bernet, A.Smith, S.Blake; difserv draft, work under progress

[10] RFC 2474 *: Definition of the Differentiated Services Field (DS Field) in the Ipv4 and Ipv6 Headers*;

[11] RFC 2475: *An Architecture for Differentiated Services*;

[12] RFC 2598: *An Expedited Forwarding PHB*, V.Jacobson, K.Nichols, K.Poduri.

[13] *The Joint DANTE/TERENA Task Force TF-TANT*; http://www.dante.net/tf-tant/

[14] *Quantum Project (the Quality network Technology for User-Oriented Multi-Media)*; http://www.dante.net/quantum/

[15] *Differentiated Service Experiment*: Report; T.Ferrari (editor), TF-TANT interim report Jun 99 – Sep 99. http://www.cnaf.infn.it/~ferrari/tfng/ds/del-rep1.doc

[16] RFC 1305 : *Network Time Protocol (Version 3), Specification, Implementation and Analysis*; David L. Mills

[17] Advanced Networks and Services; http://www.advanced.org/

[18] *Internet Delay Measurements using Test Traffic*; H.Uijterwaal, O.Kolkman – RIPE Network Coordination Centre

[19] *The Cost of QoS Support in Edge Devices: An Experimental Study;* R.Guerin, L.Li, S.Nadas, P.Pan, V.Peris.

[20] RFC 1046: A Queuing Algorithm to Provide Type-of-Service for IP Links; W.Prue, J.Postel.

[21] *Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks*; H.Zhang.

[22] *Why WFQ Is Not Good Enough For Integrated Services Networks*; J.C.R.Bennett, H.Zhang.

[23] *The bandwidth guaranteed prioritized queuing and its implementations law*, K.L.E. Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE Volume: 3 , 1997, Page(s) 1445-1449 vol3.

[24] *A comparative study of parallel and sequential priority queue algorithms;* Robert Rönngren and Rassaul Ayani; ACM trans. Model. Comput. Simul. 7,2 (Apr. 1997), pages 157 – 209.

[25] Cisco IOS 12.0 Quality of Service Solution configuration Guide, Cisco Press, S.Jose Ca. 1999, pp. QC41-QC54

[26] *Link-sharing and Resource Management Models for Packet Networks*; S. Floyd, V.Jacobson, ACM Transactins on Networking, Vol 3 No. 4, Aug 1995,

[27] *Implementing Real Time Packet Forwarding Policies using Streams*; I.Wakeman, A.Ghosh, J.Crowcroft.

[28] The bandwidth guaranteed prioritized queuing and its implementations Law, K.L.E. Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE Volume: 3 , 1997, Page(s): 1445 -1449 vol.3

[29] A comparative study of parallel and sequential priority queue algorithms; Robert Rönngren and Rassul Ayani; ACM Trans. Model. Comput. Simul. 7, 2 (Apr. 1997), Pages 157 – 209

[30] A. Banerjea and S. Keshav. Queuing delays in Rate Controlled Networks. Proceedings of IEEE INFOCOM '93, San Francisco, 1993.

[31] D. Bertesekas and R. Gallager. Data Networks Second Edition. Prentice Hall, 1992.

[32] C.R. Bennet and H. Zang. WF2Q: Wrost-Case Fair Weighted Fair Queuing. Proceedings of IEEE INFOCOM '96, S. Francisco, 1996.

[33] J. Davin and A. Heybey. A simulation study of fair queuing and policy enforcement. Computer Communication Review, Vol. 20, N. 5, October 1990, pp.23-29.

[34] A. Demers, S. Keshav, and S. Shenker. Design and Analysis of Fair Queuing Algorithm. Proceedings of ACM SIGCOMc'89, Austin September 1989.