Testing IP Differentiated Services Implementations

Carlos Parada, Jorge Carapinha, Francisco Fontes, Solange Lima and Paulo Carvalho
{carlos-f-parada, jorgec, fontes}@ptinovacao.pt, Portugal Telecom INovação, S.A., 3810-106 Aveiro, Portugal
{solange, paulo}@uminho.pt, Departamento de Informática,Universidade do Minho, 4710-059 Braga, Portugal

Abstract: Diffserv architecture is pointed out as a promising solution for the provision of QoS in the Internet in a scalable manner. The main objective of this work is to test, evaluate and discuss, from a practical point of view, two platforms for implementing Diffserv services: one developed at ICA/EPFL for the Linux OS and the other based on Cisco Systems routers. After comparing the configuration strategy in each platform, QoS related functionalities (e.g. classification, marking, policing, shaping) are tested and assessed. In particular, the implementation of EF and AF PHBs is analysed. The capacity of providing bandwidth guarantees and performing adequate traffic conditioning is evaluated as well as the impact background traffic has on delaying high priority traffic. Moreover, the computational effort Diffserv puts on routers is also measured in terms of CPU utilisation.

Keywords: *QoS, Diffserv, PHB, Expedited Forwarding (EF), Assured Forwarding (AF), Best Effort (BE), Traffic Conditioning, Performance Testing.*

1. INTRODUCTION

The Differentiated Services (Diffserv) architecture is considered a promising solution to implement Quality of Service (QoS) in the Internet in a scalable manner [1]. Bringing the network complexity to edge routers, it aims to keep the core network simple. In opposition to Integrated Services (*IntServ*) architecture [2], Diffserv does not require per-flow state and signalling at every hop [3]. Instead of dealing with individual flows, traffic is aggregated in a limited number of classes of service (CoS) according to its QoS requirements. Traffic belonging to each class is forwarded in each node according to a defined Per Hop Behaviour (PHB). The Internet Engineering

Task Force has defined two PHB. The Expedited Forwarding (EF) PHB is intended to offer a low delay, low jitter, low loss and assured bandwidth service [4]. The Assured Forwarding (AF) PHB is intended to offer a group of four service classes with a certain assigned level of forwarding resources (buffers and bandwidth), and three drop precedence (DP) per class, assuring to each class a minimum bandwidth [5]. In a Diffserv network, services are provided using traffic classification and traffic conditioning (TC) at network edges coupled with the concatenation of PHBs along the transit path [1].

Diffserv routers are expected to perform different functions (not mutually exclusive) depending on their network or domain location herein called edge and core components. Edge components are located at DS domain boundaries and their main functions are classification and TC based on a set of pre-defined rules. The enforcement of TC is fundamental so that a Service Level Agreement may be established either between a customer and a DS domain or DS domains. Core components are located inside a DS domain and their main functions are classification of traffic aggregates and queuing.

The main objective of this work is to test, evaluate and discuss, from a practical point of view, two platforms for implementing Diffserv services: one developed at ICA/EPFL (*The Institute for Computer Communications and Applications / École Polytechnique Fédérale de Lausanne*) for the Linux OS [6] and the other based on Cisco Systems routers running IOS versions *12.1(1)T* and *12.1(3)T*. Although other Diffserv implementations exist [7, 8], the two chosen are widely used. While ICA/EPFL implementation is publicly available and open, Cisco equipment was already part of our network test infrastructure. The Linux testbed, involving edge and core routers, interconnects two networks geographically located at Portugal *Telecom Inovação* (PTIN) and *Instituto de Telecomunicações* (IT) in Aveiro city. The Cisco testbed is located at PTIN.

For a wide variety of test scenarios, this study analyses how differentiated services may be configured in both platforms, discussing their underlying functionalities. Edge and core components are assessed attending to their expected behaviour. The implementation of PHBs and DPs is appraised regarding bandwidth allocation and achieved throughput. On Cisco testbed, the impact of different background traffic on high priority classes is also measured in terms of delay. The impact on router performance is also measured in terms of CPU utilisation.

Recent studies [9,10] focused on particular aspects of this study. However, in our opinion, the wide variety of tests carried out identifying the main strengths, weaknesses and problems of the two platforms are an additional contribution to the field.

After a brief introduction, a summary of Diffserv functionalities under evaluation is presented in section 2. How these functions are supported in

Linux and Cisco IOS is described in sections 3 and 4, respectively. The complete set of experiments and results are presented and discussed in section 5 and 6 for Linux and Cisco IOS testbeds, respectively. The main conclusions are reported in section 7.

2. EVALUATED DIFFSERV FUNCTIONALITIES

Diffserv functionalities can be divided in edge and core components. A brief description of their objectives is highlighted below before discussing their implementation on ICA/EPFL and Cisco IOS platforms.

2.1 Edge Component

An edge component may encompass the following functions: *Classification* selects packets based either on one or more header fields (Multi-Field/MF Classification) or on the DS CodePoint (DSCP) (Behaviour Aggregate/BA Classification). While the former is usually applied to individual flows, the latter is applied to traffic aggregates already marked. *Marking* sets the DS field to a particular DSCP [3] so that a particular PHB can be applied to them. *Metering* measures the temporal properties of traffic previously classified. This information, verified against the traffic profile specified in the Traffic Conditioning Agreement, is used by other TC functions to trigger according actions to packets in or out-of-profile. *Policing* enforces a specified traffic profile preventing non-conformant traffic from entering the network by dropping or remarking it with lower priority. *Shaping* regulates traffic submitted to the network by adjusting traffic characteristics to a defined profile. The Token Bucket (TB), Leaky Bucket (LB) or combinations thereof are common algorithms used for shaping.

2.2 Core Component

A core component may encompass the following functions: *Classification* selects traffic based on a single header field - the DSCP- (BA Classification). *Queuing* is essential to implement differentiation. Apart from FIFO, which *per si* is not oriented to packet differentiation, Priority Queuing (PQ) and Fair Queuing (FQ) are possible approaches to implement QoS-aware queuing. PQ implements a strict priority among existing queues. This means that while a high priority queue has traffic to be scheduled, low priority traffic will not be served. Although this mechanism may be particularly convenient to implement EF PHB, it may lead to traffic starvation. FQ aims

to solve PQ limitations by controlling the bandwidth allocated to each class. Variants of this algorithm have been used to implement at the same time the EF and the AF PHBs. *Congestion Avoidance* prevents queue congestion by discarding packets according to a particular criterion. Initially proposed by Jacobson and Floyd [11], Random Early Detection (RED) algorithm is a reference in this context. RED parameters include a minimum and a maximum threshold (*Min, Max*), and a drop probability (DP) for *Max* (see Figure 3). RED variants have been developed such as WRED, GRED or RIO to deal with multiple AF DPs.

3. LINUX PLATFORM

The Linux implementation under study was developed at ICA/EPFL [6]. In order to establish a Diffserv network, the configuration strategy lays in the definition of a certain number of nested functions. This is achieved using mainly the *tc* tool. *ip*, *route* or *ipchains* can also be used. Three entities can be configured with *tc*: *nodes* (have a queuing discipline - *qdisc* - associated with them), *filters* (classifiers), and *classes*.

3.1 Edge Component

At the edge level, the main implemented functionalities are as follows: *Classification MF/BA* and *Marking* are achieved using *qdisc* DSMARK, being created as many classes as the number of existing CoS. *Policing* is also implemented using *qdisc* DSMARK. It is possible to define *Dropping* or *Remarking* of out-of-profile packets. *Shaping* is implemented using *qdisc* Token Bucket Filter (TBF), which follows a Token Bucket algorithm.

3.2 Core Component

At the core level, the main implemented functionalities are as follows: *Classification BA* is configured at the nodes level, using DSCP *filters*, based on the *tcindex* classifier. *Queuing* is configured mainly using *qdisc* Class-Based Queuing (CBQ). Although there are other *qdiscs* available (apart from FIFO), common policies may coexist in the same CBQ, for instance strict priority, proportional bandwidth sharing, guaranteed bandwidth, or traffic isolation. *Congestion Avoidance* is implemented resorting to *qdisc* RED and Generalised RED (GRED). The first one is an implementation of a RED-based algorithm, while the second one allows defining *n* simple REDs.

4. CISCO PLATFORM

The Cisco platform is configured through the usual Command Line Interface (CLI). Setting up DS services in this system is far simpler than in Linux, however, it is less flexible than the latter since nesting functionalities are not allowed. Individual mechanisms can be configured separately.

4.1 Edge Component

At the edge level, the main implemented functionalities are as follows: *Classification MF/BA* and *Marking*, such as in Linux, are not carried out separately. Cisco implementation uses Policy-Based Routing (PBR), which allows the classification of packets based on multiple fields. *Policing* is based on Committed Access Rate (CAR). Traffic exceeding CAR can be either dropped or remarked. *Shaping* is a functionality implemented through Generic Traffic Shaping (GTS). GTS follows the TB algorithm.

4.2 Core Component

At the core level, the main implemented functionalities are as follows: *Classification BA* is a function embedded in the queuing functions. *Queuing* includes three algorithms in addition to FIFO: PQ implementing a strict priority scheme; Custom Queuing (CQ) allowing a percentage bandwidth allocation scheme; Class-Based Weighted Fair Queuing (CB-WFQ) where a minimum guaranteed bandwidth can be defined for each class (useful to implement AF classes), being the remaining bandwidth divided proportionally to the reserved bandwidth. A high priority class can also be defined, having strict priority. *Congestion Avoidance* mechanisms are implemented through Weighted RED (WRED), which uses multiple drop probability profiles.

5. TESTING LINUX PLATFORM

In this section, the Linux differentiated services implementation is configured, tested and evaluated.

5.1 Traffic Related Tools and Equipment

Two traffic generation tools are used in the experiments: *mgen* for UDP traffic and *netperf* for TCP. While in *mgen* the volume of generated traffic

needs to be specified, in *netperf* traffic is generated according to TCP ability to adapt to network load conditions. Traffic analysis is carried out resorting to the *tcpdump* tool and *PrismLite / RADCOM* equipment. While *tcpdump* is used for simple bandwidth measurements and debugging, and *PrismLite / RADCOM* allows more elaborated numerical and statistical analysis.

The main testbed equipment consists of routers and end systems running *Linux RedHat 6.X* and *Mandrake 7.X*. The *tc* version used in these tests is the one included in the *iproute2-000503* with the patch *ds-8* for *iproute2*, using the *kernel 2.3.40*.

5.2 Linux Testbed

The test platform running Diffserv over Linux is illustrated in Figure 1. The clouds represent the DS networks located at PTIN and at IT, respectively. The network layout consists of two Linux systems, a traffic sender and a receiver, and three Linux routers. These routers (two edge, one core) form a single DS domain.

All nodes are connected using Ethernet at 10Mbps, except the core router at PTIN and the edge router at IT, which use an ATM CBR connection, running classical IP over ATM. The ATM connection allows limiting the peak rate to a pre-defined value. This option is taken in most of the experiments in order to force a backbone bottleneck. In this way, a certain congestion level can be easily forced and managed. *PrismLite*/*RADCOM* measurements were carried out at core router outgoing ATM link.

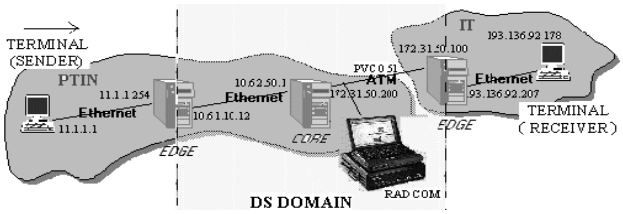

Figure 1. Testbed used in Linux tests**.**

5.2.1 Testing Edge Functionalities

The objective of this set of experiments is to assess the effectiveness of TC actions. *mgen* and *tcpdump* are used as traffic generator and traffic analyser respectively. Test scenarios for different traffic rates, packet sizes, source/destination address and ports are considered. Table 1 summarises the results obtained.

Table 1. Summary of the Linux test results (edge).

TEST	OBJECTIVE	RESULTS
Marking (DSMARK)	Check if all packets are correctly marked based on the traffic characteristics	The Marking is correct for all different traffic scenarios
Policing with Dropping (DSMARK drop)	Check if packets are policed and dropped (if needed) correctly	Inconclusive results (discussed below)
Policing with Remarking (DSMARK remark)	Check if packets are policed and remarked (if needed) correctly	Inconsistent behaviour (discussed below)
Shaping (TBF)	Check Traffic Shaping	Behaviour generally correct. The shaped rate was correct but the inter-packet time varied from about 1 to 15ms (discussed below)

In *Policing with Dropping*, after a large number of measurements, no packet drops were registered. The reason for this behaviour may be explained by the DS software version in use.

In *Policing with Remarking*, after exhaustive tests it was verified that *Policing* was accomplished correctly for rates up to 800Kbps. For upper rates, the behaviour is equal as if 800Kbps was defined. The reason for this is the low precision of the Linux system timer, which is unable to handle small time intervals. This also explains the large inter-packet time variation that occurs in *Shaping*. This behaviour has also been noticed and conveniently explained in [12].

5.2.2 Testing PHBs

This section covers an important set of experiments whose objective is to evaluate the traffic forwarding behaviour in the core router, assessing whether traffic is being differentiated as expected or not. The measurements focus on EF, AF and BE PHBs in terms of allocated and used bandwidth, in the presence of UDP and TCP traffic. CBQ is the Linux *qdisc* in use. Bandwidth in the ATM link was limited to 4Mbps and 6Mbps (approximately 3.7 and 5.5Mbps at IP level) depending of the test. Other test parameters are as follows: EF peak rate of 1Mbps; AF1x and AF4x minimum guaranteed rate of 1Mbps and 1.5Mbps, respectively.

PHBs testing results are similar for UDP (Figure 2-a) and 2-b)) and TCP traffic (2-c) and d)). Figure 2-a) and c) represent EF and BE achieved throughput, while b) and d) represent EF, AF11, AF41 and BE throughput, over a time period of one minute. The different traffic is launched along this period. In the TCP test, one connection is used for each PHB.

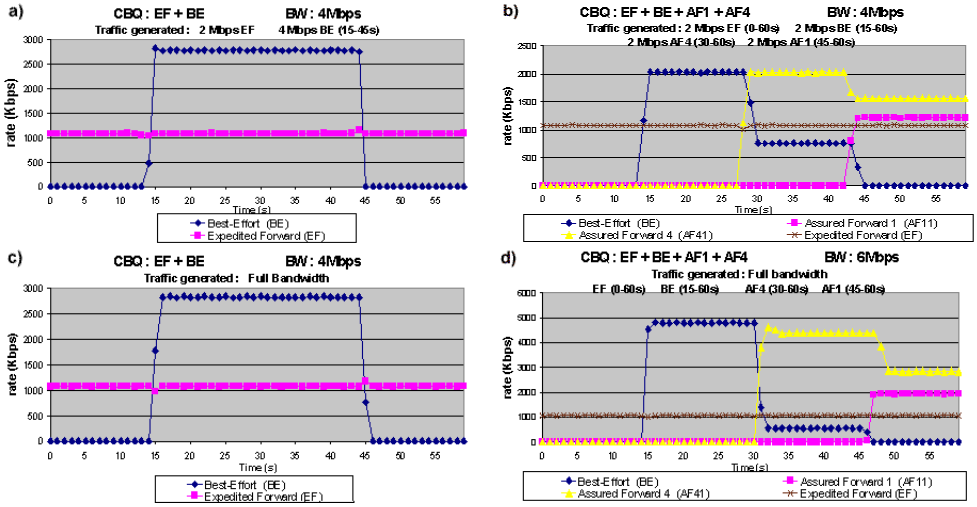

Figure 2. Throughput per PHB for UDP and TCP traffic (UDP – a) and b); TCP – c) and d)).

Figure 2 shows that EF traffic is clearly isolated from other traffic, i.e. both BE and AF packets do not interfere with EF (although a slight variation may occur when other traffic is injected in the network). Thus, EF throughput is independent of the remaining traffic and the allocated rate is guaranteed for traffic in-profile, i.e. above 1Mbps packets are dropped. For AF traffic, the minimum allocated bandwidth is also guaranteed. The remaining bandwidth (from 1Mbps to 1.5Mbps) is shared (not evenly) among the existing classes. As expected, BE traffic only gets through when high priority traffic does not take the maximum available bandwidth.

5.2.3 Testing Drop Precedence

This section focuses on how a class with different drop precedence is affected by congestion and by a particular congestion control mechanism – GRED. A single AF1x class with three DP is considered (both for UDP and TCP traffic). GRED parameters in use are illustrated in Figure 3.

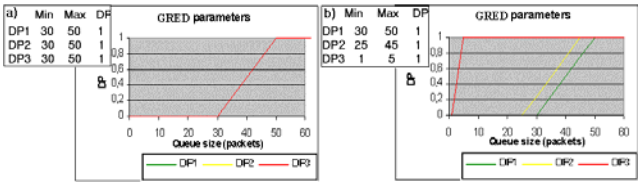

Figure 3. GRED parameters.

The ATM connection bandwidth was set to 2Mbps (about 1.8Mbps at IP level). The results for UDP and TCP traffic are shown below.

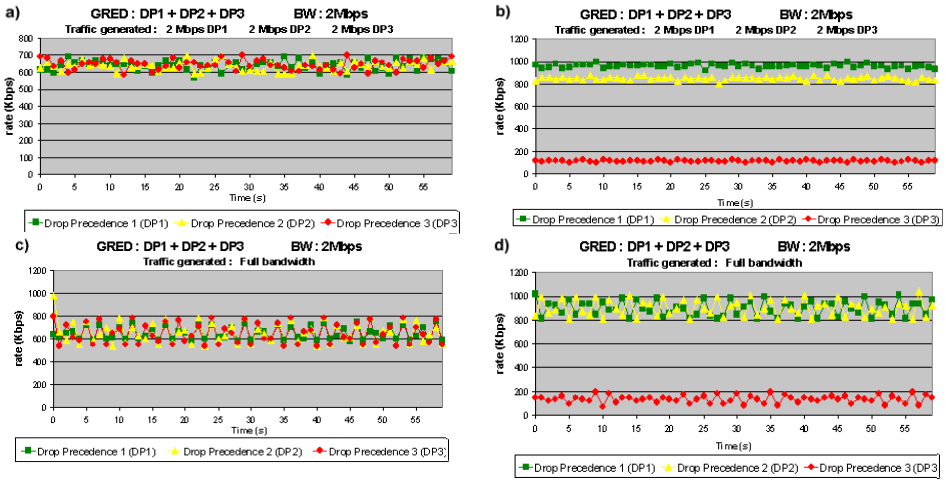

Figure 4. Throughput for the AF1x class (UDP – a) and b); TCP – c) and d)).

Figure 4 shows that, for equal GRED parameters, each AF1x receives identical treatment regarding throughput and packet loss. For different GRED parameters (*Min* and *Max*), AF1x behaviour evolves consistently, for instance, traffic with higher DP is drastically dropped and AF13 gets the smaller bandwidth share. Finally, AF1x throughput for TCP traffic shows more variability than the UDP traffic as consequence of TCP slow-start.

5.3 Linux Tests Summary

Positive aspects:

- great flexibility in the configuration of Diffserv mechanisms due to the capacity of nesting functionalities;
- number of Diffserv functionalities implemented;
- Marking and Shaping work correctly;
- easy to implement standard PHBs with the existing Queuing and Congestion Avoidance mechanisms;
- good performance both for UDP and TCP traffic.

Negative aspects:

- complexity and size of the configuration scripts;
- lack of documentation;
- problems with Policing (Dropping/Remarking) and Shaping (jitter).

6. TESTING CISCO IOS PLATFORM

In this section, the Cisco IOS differentiated services implementation is configured, tested and evaluated.

6.1 Traffic Related Tools and Equipment

The traffic generation tools used in the experiments are *mgen* for UDP traffic and *netperf* for TCP. Traffic analysis is carried out using *tcpdump* and *PrismLite / RADCOM*. *SmartBits* equipment is used for delay analysis (it solves clock synchronisation problem). The main testbed equipment consists of one *Cisco 7200* with *IOS 12.1(3)T*, one *Cisco 7500* with *IOS 12.1(1)T*, routers and end-systems running *Linux RedHat 6.X* and *Mandrake 7.X*.

6.2 Cisco Testbed

The test platform using Cisco routers is illustrated in Figure 5.

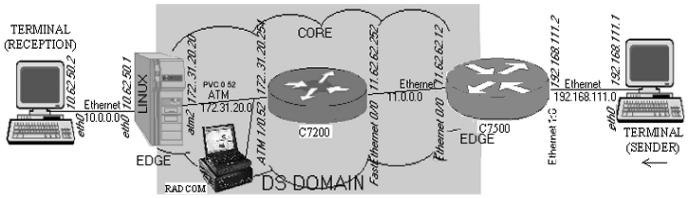

Figure 5. Testbed used in Cisco tests.

All nodes (located at PTIN headquarters) are connected using Ethernet at 10Mbps, except the core and Linux router, which use an ATM nrt-VBR connection with classical IP over ATM. Such as in Linux, this connection allows forcing a bandwidth bottleneck in the routers. *PrismLite* / *RADCOM* measurements were carried out at core router ATM outgoing link. Due to the lack of support for the DSCP field in the used IOS versions we have used the *precedence* field (this problem was solved in IOS 12.1(5)T).

6.2.1 Testing Edge Functionalities

Following the sort of experiments reported for Linux, this set of tests assesses the effectiveness of TC actions such as Marking, Policing and Shaping. As in Linux, *mgen* and *tcpdump* are used as traffic generator and traffic analyser, respectively. Table 2 summarises the results obtained.

Table 2. Summary of the Cisco test results (edge).

TEST	OBJECTIVE	RESULTS
Marking (PBR)	Check if all packets are correctly marked based on the traffic characteristics	Marking correct for different source rates, traffic types and packet lengths
Policing with Dropping (CAR drop)	Check if packets are policed and dropped (if needed) correctly	Policing / Dropping correct for different source rates, traffic types and packet lengths
Policing with remarking (CAR remark)	Check if packets are policed and remarked (if needed) correctly	Policing / Remarking correct for different source rates, traffic types and packet lengths
Shaping (GTS)	Check Traffic Shaping	Shaping correct for different traffic types and bucket sizes. Both rate and inter-packet time are shaped correctly

6.2.2 Testing PHBs

Our objective is to assess how EF, AF and BE PHBs perform on Cisco equipment regarding the allocated and used bandwidth, with either UDP or TCP traffic. The ATM connection bandwidth was set to 5Mbps, EF peak rate to 1Mbps, AF1 and AF4 minimum guaranteed rates to 1Mbps and 1.5Mbps, respectively, and CB-WFQ (see results in Figure 6). Traffic corresponding to a given class is transmitted along pre-defined time periods.

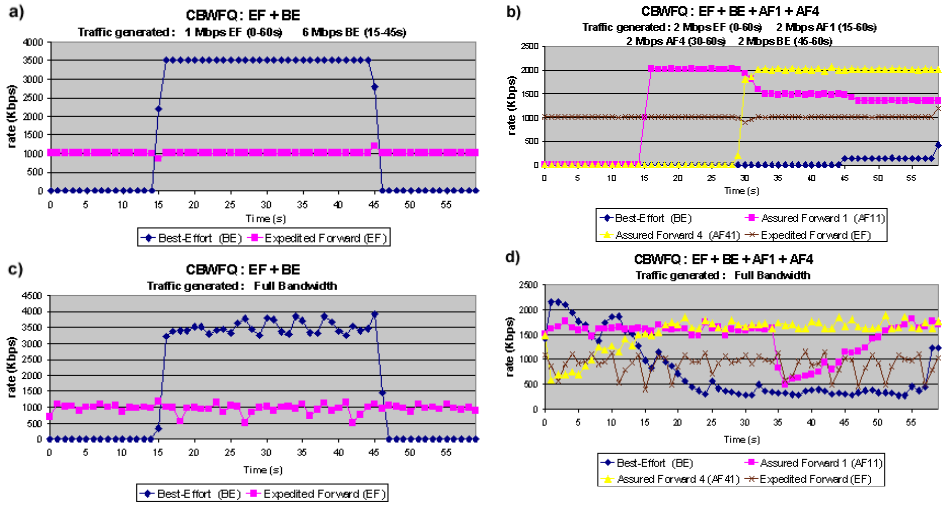

Figure 6. Throughput per PHB for UDP and TCP traffic (UDP - a) and b); TCP c) and d)).

The results obtained for UDP and TCP traffic differ considerably. In fact, throughput measurements for UDP traffic follow our expectations. The rate achieved by EF is constant and limited by its peak rate. AF classes receive their minimum allocated rate, being the remaining bandwidth divided by them proportionally to the reserved bandwidth. As [1, 3] do not state how the remaining bandwidth is distributed, BE traffic also shares (by option) this bandwidth in the same way as other classes (see Figure 6-b)). For TCP traffic the instability around reserved bandwidth value is notorious. Figure 6-d) shows that AF11 throughput is kept below the minimum rate during more than 10s. In our opinion, this behaviour is a result of heavy packet dropping at the router and consequent TCP slow-start adaptation.

6.2.3 Testing Drop Precedence

In the following experiment AF1x behaviour is tested for WRED. Traffic is generated as AF11, AF12 and AF13 over an ATM connection limited to 2Mbps. The tests were carried out for UDP and TCP traffic and the results are shown in Figures 7-a) and 7-b), respectively.

Unexpectedly, throughput measurements showed invariance for the set of WRED parameters shown in Figure 3. AF1x throughput results were expected to be close to the ones presented in Figure 4. However, for UDP traffic, either AF11 or AF12 or AF13 can take over (almost) the maximum allocated bandwidth randomly (Figure 7-a). For TCP traffic, the available link capacity is always used in a balanced way among the existing AF1x traffic (Figure 7-b). Once again, the variability of TCP rates is clear.

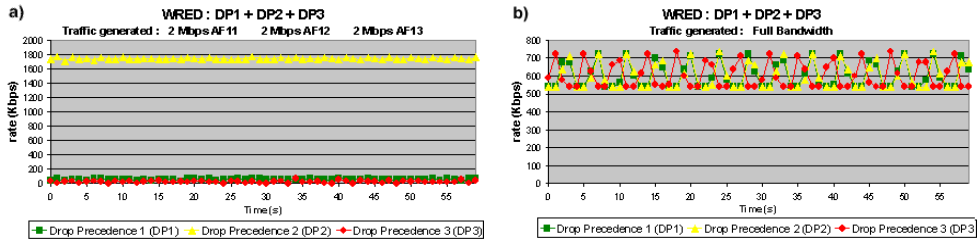

Figure 7. Throughput for the AF1x class (UDP – a); TCP – b)).

This unexpected and incorrect behaviour may be due to the use of ATM PVC interfaces where WRED cannot be configured at the interface level. In order to solve this, CB-WFQ was defined at ATM sub-interface level, where WRED was configured. Although this allowed proceeding with the tests, the WRED parameterisation appeared not to be effective.

6.2.4 CPU Utilisation

In this section, the impact of supporting Diffserv functionalities is evaluated in terms of additional CPU utilisation. An UDP traffic load of 8Mbps is equally distributed among EF, AF1, AF4 and BE (2Mbps each). Policing and Shaping is done at 1Mbps for each traffic class.

Graphs a) and b) in Figure 8 illustrates the CPU utilisation at the ingress router for packet sizes of 500 and 50 bytes, respectively. Graphs c) and d) correspond to equivalent measurements carried out at the core router.

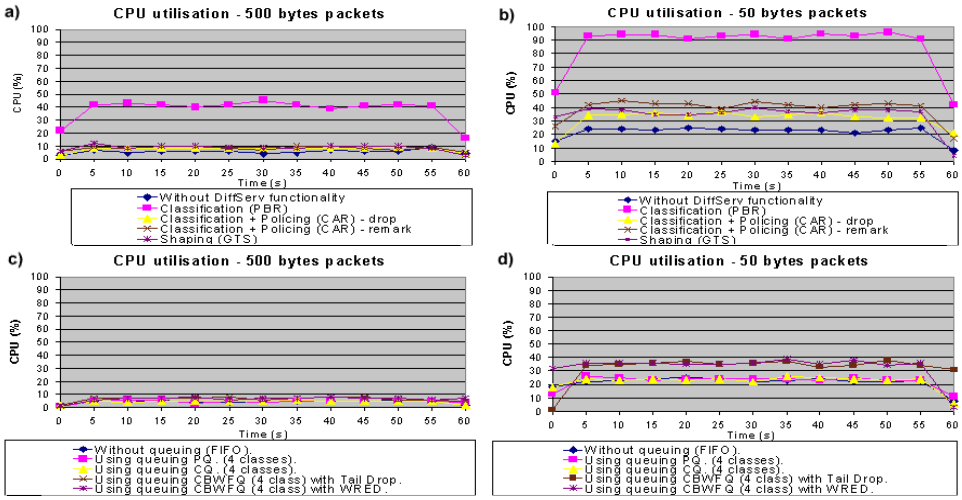

Figure 8. CPU utilisation.

The results presented show that edge functionalities require higher computational effort when comparing with the *without Diffserv* case, especially, for small packet sizes. Supporting *MF Classification* (PBR) is a CPU-consuming task. This behaviour is unexpected as *Classification and Policing* using CAR adds *Policing* to *Classification* and remains a lighter task. These results illustrate that PBR is significantly more inefficient than CAR. Figure 8-a) shows an increase from below 10% to over 40% in CPU utilisation, while for small packets (50 bytes) this value goes above 90%. CPU utilisation almost doubles when running *Classification and Policing (CAR) with remarking;* this increase is smaller when *dropping* is carried out instead of *remarking*.

Most of the tested core functionalities require a negligible increase on CPU utilisation, either for packets of 50 or 500 bytes. Only CB-WFQ implementation causes an additional utilisation of around 10%, for the smaller packet sizes. Generically, the results show that reducing packet size

(the total number of packets increases as the rate is kept constant) requires additional processing from DS routers, especially at the network edge. For higher traffic rates the CPU requirements may increase substantially.

6.2.5 Delay Measurements

The main objective of delay measurements was to assess the ability to offer low delays to high priority traffic. The initial network layout presented in Figure 5 was slightly modified so that traffic is sent and received at the same equipment (*SmartBits*) (Figure 9). This solves clock synchronisation problem when measuring absolute delays. To increase accuracy, the delay results presented below are taken from the average of 100 measurements.

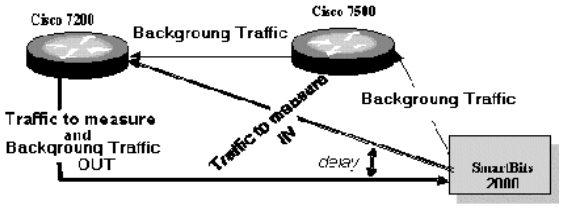

Figure 9. Testbed used for delay measurements.

These tests focused on EF PHB in order to assess if a low delay can be guaranteed in the presence of low priority background traffic. The ATM connection was set to 5Mbps. The traffic is generated so that the network operates without packet loss. The test scenario includes different types of background traffic competing with a single EF traffic flow. Background traffic can be either BE (CBR or on-off (bursty)) or AF (CBR), for different packet size and for two link loads (see results in Figure 10).

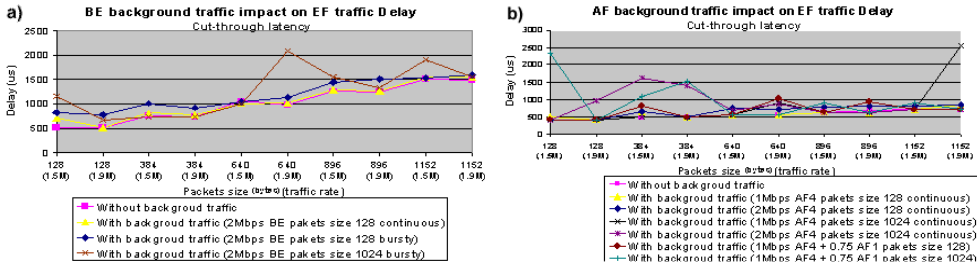

Figure 10. Impact of background traffic on EF PHB.

Note that, EF traffic is slightly delayed by BE background traffic essentially in the presence of bursty (ON-OFF) traffic. EF delay is also

influenced by BE packet sizes, as serialisation delay increases (as peaks in Figure 10-a demonstrate). When a background packet is being transmitted, even if an EF packet arrives it has to wait until the BE packet transmission is completed (recall that the implemented CB-WFQ defines a low-latency queue for EF traffic). Thus, the higher BE packet size is, the longer EF traffic waits for being scheduled. In the presence of bursty BE traffic this effect is stressed as the BE queue has periods of high activity. The difference of having AF or BE as background traffic is not significant.

6.3 Cisco Tests Summary

Positive aspects:

- configuration of services using simple and few commands;
- number of Diffserv functionalities;
- TC mechanisms and Queuing generally work as expected;
- amount and quality of available documentation;
- platform used worldwide.

Negative aspects:

- lack of DSCP support (included from Cisco IOS version 12.1(5)T on);
- mapping to layer two with some limitations (e.g. ATM);
- problems in running WRED in ATM interfaces;
- CB-WFQ queuing discipline for TCP traffic showed problems;
- less flexible than the Linux implementation, but much simpler.

7. CONCLUSIONS

The implementation of IP differentiated services on two platforms – a Linux and a Cisco IOS based testbed – has been established, configured, tested and evaluated regarding both Diffserv functionalities and algorithms supported. This work also assessed the impact edge and core functionalities have on Cisco routers and the impact background traffic has on EF traffic.

The Diffserv implementation developed at ICA/EPFL for Linux allows a flexible set up of services through a configuration strategy based on nested functionalities. This flexibility may however lead to intricate and long configurations. The Cisco testbed, including a smaller number of functionalities oriented to traffic differentiation, is able to provide Diffserv services while keeping the overall configuration simple. Concerning Diffserv functionalities, ICA/EPFL implementation allows the correct definition of rules and mechanisms to provide efficient traffic differentiation. EF traffic rate is not impaired by other competing traffic and AF traffic can achieve

minimum bandwidth guarantees. The drop precedence for AF PHB group worked as expected for UDP and TCP traffic, under GRED. Policing showed problems in remarking, for rates above a certain (machine-dependent) value.

Cisco platform was very effective on edge component implementation, where traffic policing and shaping were strictly enforced. In our testbed, the core component showed poor performance and unexpected behaviour on TCP traffic differentiation. Drop precedence tests did not lead to meaningful results due to limitations in configuring WRED effectively at the ATM level.

Measuring CPU utilisation, it was found that while the increase of CPU demand at core routers is negligible, at edge routers it may be significant depending on how classification and policing/remarking are accomplished. The increase on CPU utilisation in edge and core routers may be notorious as packet size decreases. Regarding the impact background traffic has on EF traffic, the measurements showed a slight increase on delay essentially in the presence of bursty traffic and for large packets.

Future work will include test tuning in both platforms using upgraded software versions. PHB concatenation and PDB testing will also be covered.

ACKNOWLEDGMENTS

Work included in the EURESCOM P1006 / DISCMAN Project framework.

REFERENCES

[1] S. Blake et al., *RFC 2475 - An Architecture for Differentiated Services*, 1998.
[2] R. Braden et al., *RFC 1633 – Integrated Services in the Internet Architecture: an Overview*, June 1994.
[3] K. Nichols et al., *RFC 2474 - Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, December 1998.
[4] V. Jacobson et al., *RFC 2598 - An Expedited Forwarding PHB*, June 1999.
[5] J. Heinanen et al., *RFC 2597 - Assured Forwarding PHB Group*, June 1999.
[6] W. Almesberger, http://icawww.epfl.ch/linux-Diffserv/, November 1999.
[7] NEC, http://www.nec-cx.com/products/, September 2001.
[8] IBM, http://www.networking.ibm.com/netprod.html, September 2001.
[9] T. Ferrari, P. Chimento, *A Measurement-based Analysis of EF PHB Mechanisms*, 2000.
[10] S. Radhakrishan, *Internet Protocol QoS Page*, http://qos.ittc.ukans.edu.
[11] S. Floyd, V. Jacobson, *Random Early Detection gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, pp 397-413.
[12] K.Wehrle, H. Ritter, *Problems of Traffic Shaping in ATM and IP Networks Using Standard End Systems*, ATM2000 Conference, http://atm2000.ccrle.nec.de/.