A Queue Management System for Differentiated-Services IP Routers

Goncalo Quadros, Antonio Alves, Joao Silva, Henrique Matos, Edmundo Monteiro, Fernando Boavida

CISUC – Center for Informatics and Systems of the University of Coimbra
Communications and Telematic Services Group
- Pólo II, 3030 COIMBRA - PORTUGAL
Tel.: +351-39-790000, Fax: +351-39-701266,
E-mail: {quadros, aalves, jonas, kikas, edmundo, boavida}@dei.uc.pt

Abstract. Packet scheduling and queue management strategies are key issues of DiffServ per-hop behaviours. This paper proposes a queue management system that, in conjunction with scheduling mechanisms, is able to support class differentiation. The general principles and the architecture of the queue management system are presented. The proposal is supported by a prototype that was subject to several tests, in terms of packet drops and burst tolerance. The test results are presented and analysed, allowing an assessment of the usefulness and effectiveness of the underlying ideas.

1 Introduction and Framework

One of the most challenging demands for the new generation of network elements able to provide quality of service (QoS) is to provide better ways to manage packet queues lengths, as it is well known that some form of "active queue management" is needed to obtain better performance levels of the communication system - for instance, less transit delay, less packet loss level, better use of the available bandwidth, etc.. Important research is being conducted by many teams studying and discussing models and approaches for supporting such systems [2], [7], [8].

This paper contributes to that discussion by presenting a strategy for queue management (QM) specifically in QoS-capable IP networks following the *differentiated services* (DS) framework[1] [3]. The work presented here was conducted to fulfil the requirements of a broader project, whose main goal is to develop a new approach for the support of traffic classes in IP networks, while following the same framework.

This broader, on-going project has three main goals: (1) to develop mechanisms to provide effective QoS capabilities in Network Elements (NE); (2) to conceive ways to select adequate, QoS-aware paths for packet forwarding along communication

[1] This framework is being promoted by the *Differentiated Services Working Group* of the IETF [5].

systems; (3) to implement effective ways for system management, including a strategy for traffic admission control.

This paper results from the work done in relation to goal (1), which drove to the proposal of a new PHB (per-hop behaviour) and to the construction of a router prototype that implements it [13]. Two main contributions of this prototype are a new IP packet scheduling strategy, presented in [14], and a new IP queue management strategy, presented here.

In Section 2 the general principles which rule the design of the proposed QM system are presented. Section 3 discusses its architecture, focusing on a prototype developed to test the underlying ideas. Section 4 presents the tests carried out on developed prototype and discusses the corresponding results.

2 General Operational Principles of a DS Queue Management System

Two important drawbacks characterise the traditional approach in use in the Internet for queue management, known as the *tail drop* (TD) approach [2]. The first drawback is called the *lockout* phenomenon, which happens when a single or few flows monopolise the queue space, preventing other flows from getting the space that normally they would use. The second drawback is known as the *full queue* problem and occurs because the TD approach allows full, or almost full, queues, during long periods of time. This increases the delay seen by packets and reduces the NE tolerance to packet bursts. In turn, lower tolerance to packet bursts results in a higher percentage of dropped packets and lower link utilisation levels, because packets are dropped in sequence and flows that are responsive to congestion (like, for instance, TCP flows) back-off synchronously. As a rule of thumb, a QM system should always provide room for a packet arriving at a network element.

The tail drop approach is, therefore, inadequate for the modern Internet. In addition, and considering the DS framework, a network element should have an effective way to share its resources among different traffic classes, according to their QoS needs. As buffer space is an important resource, QM disciplines have an important responsibility in terms of resource sharing, side by side with packet scheduling disciplines.

When executing drops, queue management systems should also consider the nature of the involved traffic. They should avoid dropping consecutive UDP packets belonging to the same flow, because the impact of loss will be dependent on the distance between dropped packets [9] for most applications that use this transport protocol. Consecutive dropping should also be avoided for TCP packets, in order to minimize the probability of eliminating packets belonging to the same TCP flow.

Still related to the diverse nature of traffic, QM systems should deal with the growing volume of flows that are unresponsive to congestion (for the purpose of this discussion, UDP flows are considered to be unresponsive to congestion; nevertheless, UDP flows can also be responsive to congestion, depending on the upper layer protocols that are using this transport protocol). For this, they should use mechanisms to protect responsive flows from unresponsive flows – currently, the resources freed by the former are immediately, and unfairly, used by the latter, when congestion

happens. Additionally, QM systems should implement effective means to manage unresponsive flows, to avoid congestion.

The queue management system developed at LCT-CISUC was motivated by the previous considerations. In broad terms, its design addressed two, closely related fields. In the first one, the idea was to design enqueuing and dropping processes in a way that avoids lockout, promotes burst tolerance and minimises the impact of packet loss on applications. In the second field, the idea was to conceive an effective way to manage queue lengths in order to control the drop level seen by the flows of each class[2]. Accordingly, the prototype constructed to test the proposed ideas was implemented in two phases, which resulted in two different modules: the *packet drop management* module and the *queue length management* module. These modules are presented in the next section.

3 Architecture of the Queue Management system

Figure 2 of [13], in the present QofIS'2000 Proceedings, depicts the QoS-capable router prototype implemented in LCT-CISUC, highlighting the central object of the queue management system operation – the IP output queues.

At IP level, after the IP processing activity, packets are classified and stored accordingly in an output queue. It is there that the mechanisms which differentiate the QoS provided to classes act. The classifier/marker is responsible for determining each packet class and/or for setting the information related to it in the packet's header, following the strategy defined by IETF-DSWG[3] [11].

The monitor continuously measures the QoS provided to classes, applying a QoS metric developed for this purpose. Its operation is discussed with more detail below.

The scheduler is responsible for choosing the next packet to process. The dropper is responsible for choosing the packets to drop and also, in close relation to the monitor, for adjusting some operational parameters such as queue lengths.

The queue management system, seen as whole, involves the process of storing packets in IP output queues and the action of the dropper. It is presented in the next two subsections, through the discussion of its two main modules.

3.1 The Packet Drop Management Module

There are two important parameters related to the dropper operation, that characterise each output queue: **`q_physical_limit`**, the maximum possible length for each queue (which is arbitrarily large); and **`q_virtual_limit`**, the desired maximum length for each queue.

Each time a packet is to be enqueued and the physical limit is reached it will be immediately dropped. As this limit is made extremely large, this will be uncommon. Thus, every packet or burst of packets will normally be enqueued.

---

2 Each class has its own and exclusive queue in an NE.

3 *Internet Engineering Task Force - Differentiated Services Working Group*

From time to time the dropper is activated. This time is adjustable and should reflect the length of packet bursts to be accommodated (the default value is 15 packets). When the dropper is activated, it verifies whether or not there are more packets than the amount imposed by the virtual_limit for each queue. If this is the case, the dropper discards the excess packets in order for that limit to be respected. TCP packets will only be dropped if they exceed a given percentage of the total number of packets; moreover, TCP and UDP packets are randomly dropped.

The logical operation of the Packet Drop Management Module is presented in figure B.1 of [13]. In short, the module always provides room for an incoming packet, packet bursts are naturally tolerated, the drop of UDP packets is scattered and it is possible to protect TCP traffic from UDP traffic.

3.2 Queue Length Management Module

The *Queue Length Management Module* is responsible for the dynamic allocation of buffers to classes. The allocation of buffer space to queues is performed in conjunction with the scheduler action (which distributes the transmitter capacity among classes). The former controls the level of packet drops, and the latter controls the delay seen by packets [13].

The strategy used to share buffer resources follows the one used by the scheduler to share transmitter capacity [14]. The LCT-CISUC QoS metric [13][15] is nuclear to that strategy. According to this QoS metric, the quality of service is quantified through a variable named *congestion index* (CI). There is a CI related to delay and a CI related to loss. The queue length management module uses the latter.

Each class is characterised by a *degradation slope* (DSLOPE), which determines the classes' sensitivity to the degradation of the loss level. As can be seen in figure 1, a traffic class highly sensitive to loss degradation will have a high degradation DSLOPE.

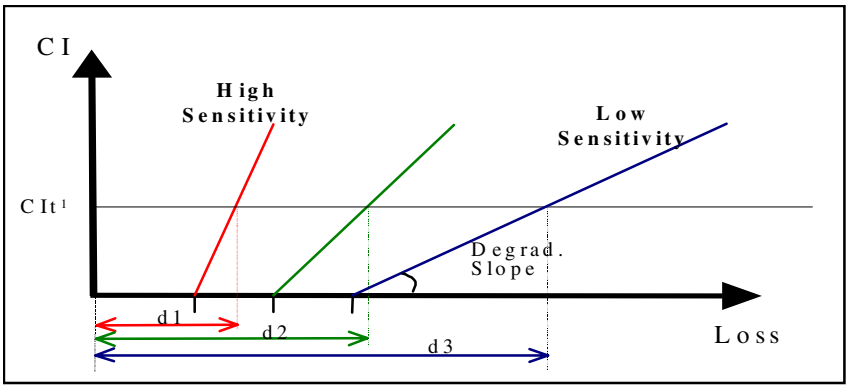

Fig. 1. CIs for three traffic classes w/ different sensitivities to loss degradation (d1, d2 and d3 represent the loss level experienced by each class when the congestion index has the value CIt)

It is easy to understand that, with growing loads – that is, as the loss CI grows – the drop level seen by most sensitive classes is lower than the drop level seen by less

sensitive classes. Using other words, more sensitive classes are protected by less sensitive classes, which absorb the major part of the degradation.

Figure 2 presents the logical operation of the Queue Length Management Module.

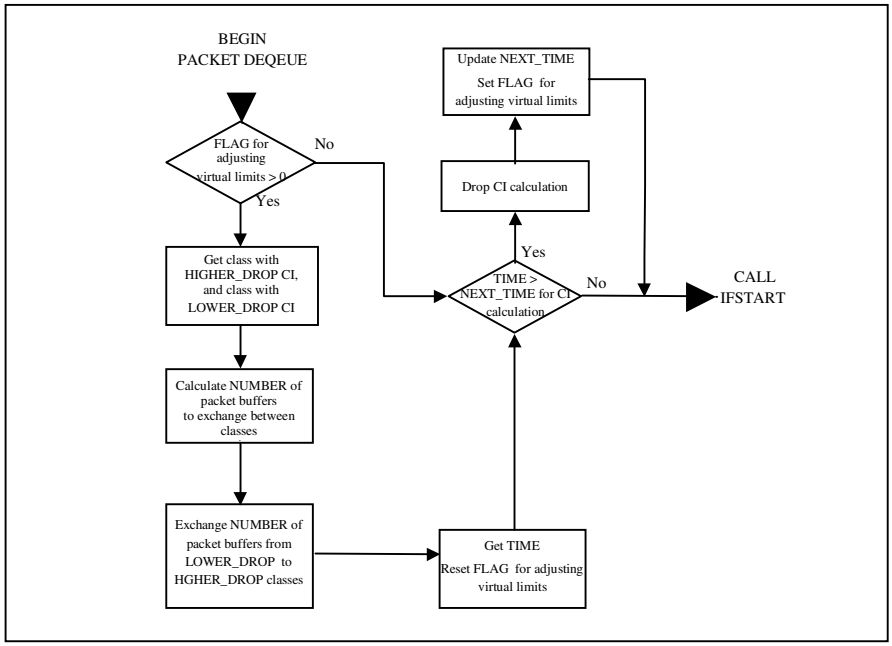

Fig. 2. Queue length management module logical operation

4 Tests

The initial testbed used to carry out the tests consisted of a small isolated network with 4 Intel Pentium PC machines configured with a Celeron 333Mhz CPU, 32 MB RAM and Intel EtherExpress Pro100B network cards. The prototype router (*Router*) ran FreeBSD 2.2.6 [6], patched with ALTQ version 1.0.1 [4], with 64MB RAM. Two hosts, *Source1* and *Source2*, generated traffic directed towards a destination host (*Sink*) through *Router*. Each host only generated traffic of a given class, in order to guarantee the independence of the generated packet streams. To perform additional tests involving one more class, another host – *Source3* – was installed in a subsequent phase.

The tests were performed with the aid of three basic tools: *Nttcp* [12], *QoStat* [1] and a modified version of Mgen [10] - Mgen_m.

Nttcp and *QoStat* were used to test the Queue Length Management Module. The former was used to generate data flows at the maximum possible rate according to a uniform load distribution, which competed for all the resources available in the communication system. The latter was used to monitor the kernel – namely the

operation of the QM system and the QoS it provided to classes. QoStat was also used to dynamically change the operational parameters related to queue management.

Mgen_m was used to test the Packet Drop Management Module. It was constructed using some parts of the MGEN code, with performance and reliability (in what respects to packet logging) in mind. The MGEN tool was also deeply modified in what respects packet generation, in order to allow the generation of IPRAW packet streams and bursty packet streams according to a strategy followed in [8], as described in 4.2 under 'General Test Conditions'.

4.1 Test of the Queue Length Management Module

The first set of tests involved only two classes and high loads, with packets of 1400-byte length. The maximum number of packets in the queuing system (MAX_PCKS_ONQ) was configured to 60.

Moreover, the two classes were configured with the same sensitivity to delay degradation: sensitivity slope equal to 45 degrees. On the other hand, the loss degradation sensitivity of the traffic classes changed with time, starting with a slope degradation of 45 degrees for both classes and, after each 25s time interval, taking up the following values (class1-class2): 40-50; 30-60; 25-65; and finally 20-70.

The results of the tests are presented in figure 3. The capacity of the prototype to differentiate classes is obvious, namely in terms of the ability to provide different loss levels to classes. When they have the same sensitivity to loss degradation, both classes suffer nearly the same loss level. Changing their sensitivity to degradation results in a coherent change of the rate of dropped packets. When a class becomes more sensitive to loss degradation (see class 2 in figure 3) the rate at which its packets are dropped decreases.

Moreover, the sharp transition shown in the graphs reveals that the system has a good capacity to react to changes. Thus, in this test, the prototype reveals a good capacity to effectively control the classes' loss level.

Notice that, as the classes are configured to have the same sensitivity to delay degradation, the average packet transit delay experienced by packets of both queues is grossly the same in all the tests - as can be verified in figure 3. It is possible to see that the delay increases as the classes sensitivities diverge. This is natural given that the asymmetry of classes sensitivity induces asymmetry of queue lengths. This asymmetry changes the delay reference, which is obviously determined by the bigger queue.

Table I shows some average values got by the QoStat tool during our experiments.

The transfer of buffer space from the class with less sensitivity to the other class is quantified. It is possible to understand that the prototype reacts to changes in the classes' degradation slope (which induces asymmetry on the correspondent loss congestion indexes), transferring buffer space from one class to the other. When both classes have the same sensitivity, the prototype gives nearly the same number of buffers (29 and 31) to each class.

It is also possible to see that, in fact, the prototype reacts to changes in the classes' DSLOPE giving the same loss congestion index to both classes. As expected, the congestion index is lower when the difference between degradation slopes is bigger.

DSlope Class1/ Class2	Class 1	Class2	Class 1	Class2
	Q_virtual_limit	*Q_virtual_limit*	*CI*	*CI*
45/45	29	31	50	49
40/50	22	38	49	48
30/60	11	49	43	43
25/65	6	54	38	38
20/70	2	58	32	32

Table I. Prototype operational values for the first set of tests

Figure 4 shows the results of one additional test, carried out using the same methodology as in the test just presented, but now having configured different sensitivities to delay degradation of classes, instead the same delay sensitivity. In the test presented in Figure 4 the delay DSLOPE of class 1 was fixed at 40 degrees and the one of class 2 fixed at 50 degrees. As expected, the prototype still reveals the capacity to differentiate traffic classes, but now, coherently, providing different average transit delay to packets belonging to different classes.

In order to evaluate the real influence of the queue management system on the router behaviour, some tests were carried out activating and deactivating it. To deactivate the QM system means to use the traditional tail-drop scheme, with maximum queue lengths of 50 packets for the queues of both classes.

The tests follow the approach mentioned before. The sensitivity to delay degradation was fixed - DSLOPE was made equal to 45 degrees during all the test.

In the first 25-second time interval, a loss degradation slope equal to 45 was used in both classes, and the QM system was activated. In the second 25s interval the QM system was deactivated. In the third interval, loss degradation slopes equal to 30 and 60 degrees were used respectively in class1 and class2, and the QM system was again activated. In the fourth time interval it was, once more, deactivated.

Figure 5 presents the results of this test. During the first time interval, as the DSLOPEs related to delay and loss were equal to 45 degrees for both classes, the classes received the same treatment in terms of average transit delay seen by packets, as well as in terms of number of packets forwarded by second. When the QM system was deactivated the average transit delay raised substantially. This corresponds to the growth of the maximum output queues length, and reveals an important fact about the efficiency of the QM system under test. In fact, with the QM system running, the same loss level can be achieved with a much lower average transit delay. That is, for a given loss level, the QM system leads to much shorter average queue lengths.

The third and forth time intervals clearly show the importance of the QM system. When the classes' sensitivities to loss degradation are different, the prototype is only able to differentiate them accordingly when the QM system is activated.

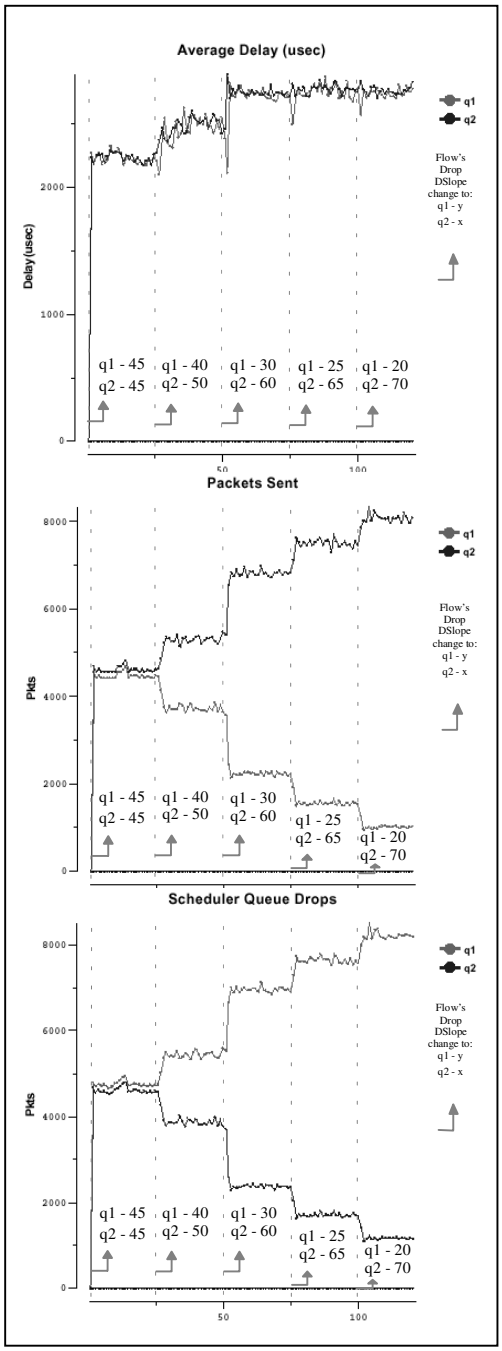

Fig. 3. Variation of average transit delay, number of packets sent, and number of dropped packets with classes' sensitiveness to loss degradation - delay dslope fixed to 45/45 (class1/class2)

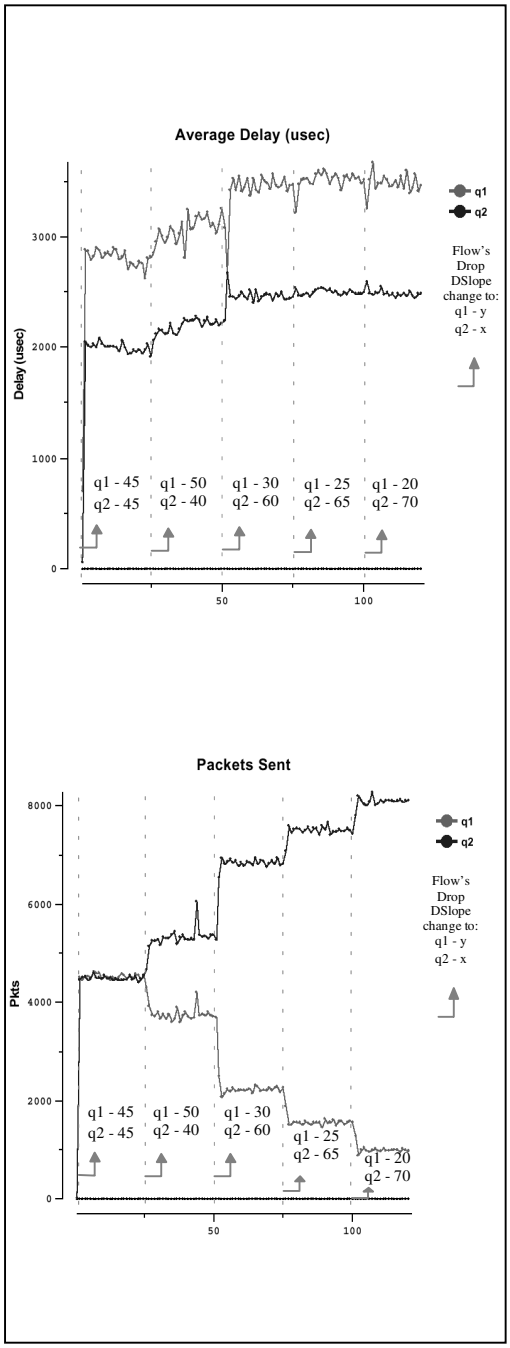

Fig. 4. Variation of average packet delay and number of packets sent with classes' sensitivity to loss degradation - delay dslope fixed to 40/50 (class1/class2)

4.2 Test of the Packet Drop Management Module

The main goals of the tests presented in this section are to evaluate the capacity of the prototype to tolerate bursts and to evaluate the effectiveness of the drop strategy. The idea was to evaluate the improvements that can be expected from the prototype behaviour, when compared to a *normal router* behaviour.

The general test strategy consisted of generating two packet streams (PS) – one *probe* PS and one *load* PS – and to measure the impact of the queuing and dropping process on the probe stream.

The first set of tests were executed using a normal FreeBSD computer as *Router*, and generating packets streams in *Source1 and Source2* hosts. As normal router, a FreeBSD v2.2.6 system running on a PC was used. The Mgen_m tool was used for packet generation purposes, which was motivated by the reasons referred above.

The second set of tests was executed using the router prototype, and generating the two packet streams on the same class. This was done in order for the tests to be comparable. As there is only one IP output queue when using a normal router, packet streams of the same class were generated to guarantee also that only one queue was used in the tests with the router prototype.

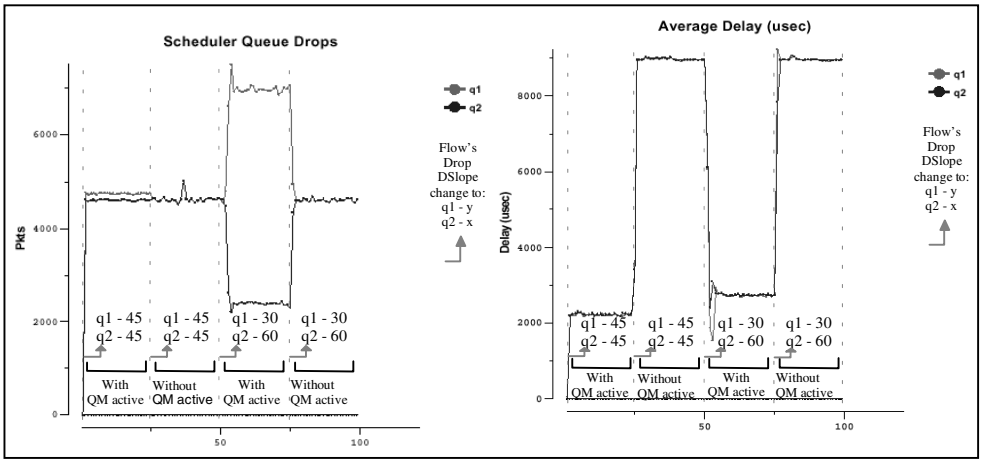

Fig. 5. Variation of average packet delay and number of packets sent with classes' sensitiveness to loss degradation with and without QM

The tolerance to packet bursts was evaluated measuring the number of packets dropped in sequence – a higher tolerance to bursts means less packets dropped in sequence.

The effectiveness of the dropper strategy was evaluated using a metric developed by Koodli and Ravikanth [9]. In general terms, the intention was to measure whether or not the prototype was able to spread out the drop of UDP packets using the referred loss-distance-stream metric. This metric determines a *spread factor* associated with the packet loss rate.

Consider a stream of packets characterised by a sequence number that starts at zero and increases monotonically by one. The difference between the sequence number of a lost packet and the one of the previously lost packet is named the *loss distance*.

According to the used metric, a lost packet is noticeable if its loss distance is no greater than delta - where delta is a positive integer named *loss constraint* .

The following subsections present (1) the general test conditions, (2) the tests carried out for evaluating the burst tolerance and (3) the test performed to evaluate the effectiveness of the drop strategy.

General Test Conditions

The tests were carried out using two different load settings – LD1 and LD2 – and two settings for traffic pattern: smooth traffic, SMT, and bursty traffic, BST. In short, the tests were performed in the following 4 different scenarios: SMT-LD1, SMT-LD2, BST-LD1, and BST-LD2.

The size of generated packets was fixed to 1000 bytes. The LD1 scenario was constructed generating each of the two packet streams at a rate of 52 Mbps. In the LD2 scenario, each packet stream was generated at 60 Mbps. SMT corresponds to traffic that follows a uniform distribution whereas BST corresponds to traffic that follows a Poisson distribution – four back-to-back packets generated at exponential time intervals. The tests result was the log file of the probe PS on *Sink*. Through it, the drop distribution was analysed.

For each of the referred scenarios ten tests were executed using the normal router, and another ten tests using the prototype router. Each test involved the generation of approximately 200.000 packets. The values presented below correspond to the average of the values obtained in each test.

Additionally, one of the scenarios was chosen (the LD2-BST scenario) and again two sets of ten tests were carried out, now using as probe flow a stream of IP Raw packets. The idea was to emulate a TCP packet stream without using any congestion control mechanism (and thus, to evaluate the differences of the prototype drop behaviour when dealing with TCP traffic instead of UDP traffic). The results of the tests are presented in the following sub-sections.

Evaluation of the Tolerance to Packet Bursts

Figures 6 through 9 show the percentage of total dropped packets in bursts of 1, 2, 3… n packets.

It is possible to see that, when the router prototype is used, the histogram shifts to left. Using other words, packets are dropped in sequences of only a few packets – in most cases only one packet. This is evident in all the scenarios used in tests.

Thus, the prototype reveals good characteristics in what respects its capacity to accommodate bursts of packets, leading to much better behaviour than in the case of a normal router, using the traditional tail-drop approach for queue management. This is also apparent even in the BST scenarios, where there are bursts of packets, and where the use of the prototype does not result in long sequences of dropped packets.

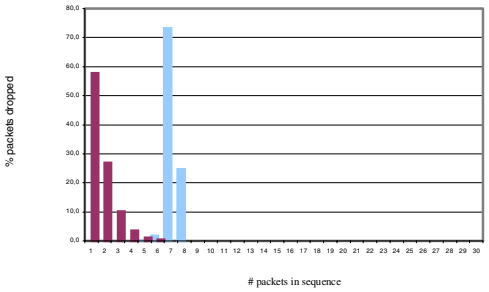

Fig. 6. Burst Tolerance (scenario LD1-SMT)

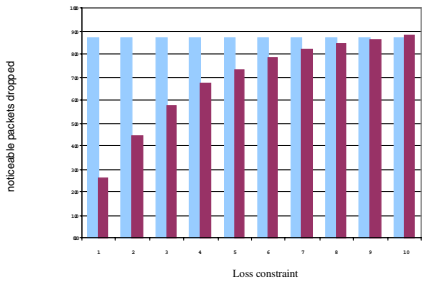

Fig. 10. Noticeable Loss (scenario LD1-SMT)

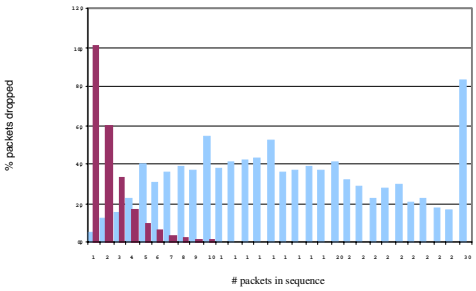

Fig. 7. Burst Tolerance (scenario LD1-BST)

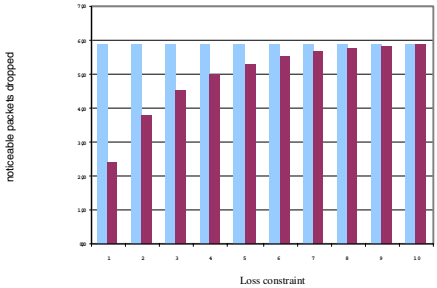

Fig. 11. Noticeable Loss (scenario LD1-BST)

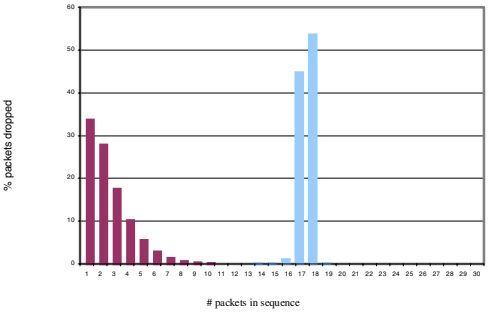

Fig. 8. Burst Tolerance (scenario LD2-SMT)

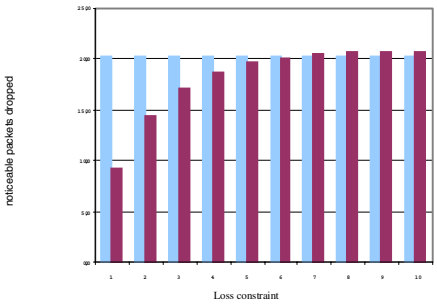

Fig. 12. Noticeable Loss (scenario LD2-SMT)

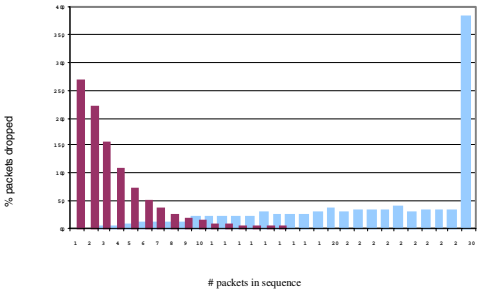

Fig. 9. Burst Tolerance (scenario LD2-BST)

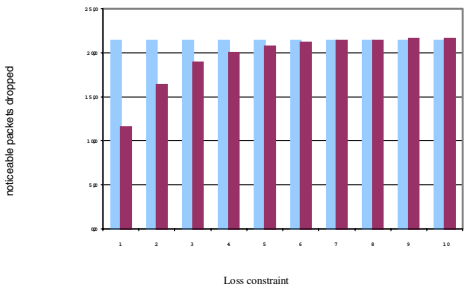

Fig. 13. Noticeable Loss (scenario LD2-BST)

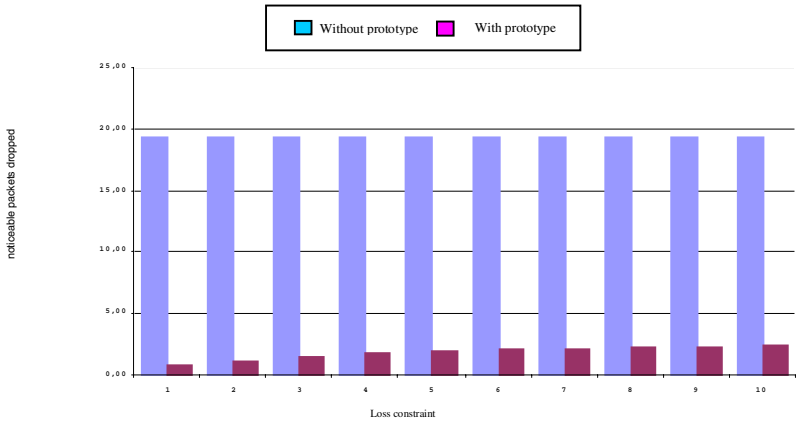

Fig. 14. Noticeable Loss (scenario LD2-BST)

Evaluation of the drop effectiveness

Figures 10 through 13 show the evolution of the *noticeable loss* with *loss constraint*, for the different load scenarios. The figures show that the prototype tends to spread out the drop of packets. In fact, it can be seen that, when using a normal router, almost all the packet drops are "noticeable" when the loss constraint is 1 (using other words, noticeable losses immediately reach a value near the maximum when the loss constraint is only 1). When using the router prototype, the percentage of noticeable losses when the loss constraint is 1 is much lower, and it grows gradually until its maximum value.

This happens in all the scenarios. It is, nevertheless, more evident with lower loads (LD1) than with higher loads (LD2); in turn, it is more evident with smooth traffic than with bursty traffic. In fact, in such conditions, the noticeable loss for low loss constraint falls more deeply, taking as reference the maximum noticeable loss. This was to be expected: higher loads and bursty traffic will tend to increase the drop level and, thus, drops cannot be as "spread out" as in the others scenarios.

Figure 14 presents the results of the same type of tests as the ones presented before (scenario LD2-BST), but now using a TCP packet stream as probe traffic. The IP raw capability of FreeBSD was used for this purpose, generating datagrams in such a way that they are processed by *router* as if they correspond to TCP traffic.

One of the most evident conclusions (see figure 14) is that the prototype effectively protects TCP traffic from UDP traffic. Noticeable losses are much lower when the prototype is in use.

In short, despite the low level of drops, the prototype shows the expected behaviour.

5 Conclusions and Future Work

Network elements are essential for the provision of predictable and controlled quality of service. Inside network elements, scheduling and queue management are important building blocks of per-hop behaviours that can lead to the desired quality of service.

This paper presented a queue management system developed for the support of differentiated services. After the presentation of the queue management system principles and general architecture, tests to its main blocks were presented. The tests revealed a good capacity for class differentiation, with good performance both in terms of packet drops and burst tolerance, showing that it is possible to overcome the drawbacks that characterise the traditional approach in use in the Internet for queue management.

The presented queue management system is part of QoS-capable router prototype being developed by the authors. Further work will be carried in the context of this prototype, namely the execution of further tests (scalability tests and tests with real load patterns) and the refining of dropping and queue-length management algorithms. Additionally, work addressing issues such as scheduling, QoS-aware routing and QoS management in under way, in more complex scenarios.

References

1. Alves, A., Quadros, G., Monteiro, E., Boavida, F.: *QoStat – A Tool for the Evaluation of QoS Capable FreeBSD Routers*, Technical Report, CISUC, July 99. http://lct.dei.uc.pt/papers/QoStat_TR.PDF
2. Braden, B., Clark,, D., Crowcroft, J., Davie, B., Deering, S., Estrin D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., Zhang, L.: Re-commendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 98.
3. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: "An Architecture for Differentiated Services", RFC 2475, December 1998. ftp://ftp.isi.edu/in-notes/rfc2475.txt
4. Cho, K: A Framework for Alternate Queueing: Towards Traffic Management by PC Based Routers, in Proceedings of USENIX 1998 Annual Technical Conference, New Orleans LA, June 1998. http://www.csl.sony.co.jp/person/kjc/kjc/papers/usenix98
5. http://www.ietf.org/html.charters/diffserv-charter.html.
6. http://www.freebsd.org
7. Feng, W., Kandlur, D., Saha, D., Shin, K.: "Blue: A New Class of Active Queue Management Algorithms" U. Michigan CSE-TR-387-99, April 1999. http://www.eecs.umich.edu/~wuchang/work/CSE-TR-387-99.pdf
8. Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance, *ACM/IEEE Transactions on Networking*, Volume 1, Number 4, August 93.
9. Koodli, R., Ravikanth, R.: One-way Loss Pattern Sample Metrics, Internet Draft, October 1999. http://www.ietf.org/internet-drafts/draft-ietf-ippm-loss-pattern-02.txt
10. The MGEN Toolset. http://manimac.itd.nrl.navy.mil/MGEN/
11. Nichols, K., et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
12. New TTCP Program, http://www.leo.org/~bartel/nttcp/
13. Quadros, G., Alves, A., Monteiro, E., Boavida, F.: *An Approach to Support Traffic Classes in IP Networks*, Proceedings of QofIS'2000. http://lct.dei.uc.pt/papers/SupportTC_TR.PDF

14. Quadros, G., Alves, A., Monteiro, E., Boavida, F.: An Effective Scheduler for IP Routers, to be published in the proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), Antibes, France, 4-6 July 2000. http://lct.dei.uc.pt/papers/NewScheduler_TR.PDF
15. Quadros, G., Monteiro, E., Boavida, F.: "A QoS Metric for Packet Networks", Proceedings of SPIE International Symposium on Voice, Video, and Data Communications, Conference 3529A, Hynes Convention Center, Boston, Massachusetts, USA, 1-5 November 1998.