An Approach to Support Traffic Classes in IP Networks

Goncalo Quadros, Antonio Alves, Edmundo Monteiro, Fernando Boavida

Laboratory of Communication and Telematics
Center for Informatics and Systems of the University of Coimbra
Pólo II, Pinhal de Marrocos, 3030 Coimbra, Portugal
`{quadros, aalves, edmundo, boavida}@dei.uc.pt`

Abstract. Per-hop behaviours capable of supporting different traffic classes are essential for the provision of quality of service (QoS) on the Internet according to the Differentiated Services model. This paper presents an approach for supporting different traffic classes in IP networks, proposing a new per-hop behaviour called D3 - *Dynamic Degradation Distribution*. The approach allows for the dynamic distribution of network resources among classes, based on the measured quality of service and on the sensitivity of classes to performance degradation, without implying any substantial change to current IP technologies. A router prototype developed according to the proposed approach is presented, along with the results of experimental tests that were performed. The test results demonstrate the feasibility and the effectiveness of the underlying ideas.

1 Introduction and Framework

Differentiated Services (Diffserv) is an architectural framework currently being studied and proposed for the Internet, by which different levels of network quality-of-service (QoS) can be provided to different traffic streams [5]. The basic principle of Diffserv networks is that routers will handle packets of different traffic streams by applying different per-hop behaviours (PHBs) to packet forwarding.

The Internet has been growing at an extremely fast pace and not always in a predictable way. The large number of every-day new users induces more and more load on the network. It also promotes the emergence of new services which, again, result in more traffic being generated. In this context, it is not surprising that one often finds very poor quality levels when using the network.

One possible approach to QoS provision on the Internet is to extend the current single-class-best-effort Internet paradigm to a multiple-class-best-effort paradigm. This is the approach that is explored in the work presented in the current paper.

The proposed concept of multiple-class-best-effort provides a way to deal with traffic classes considering their relative QoS needs but still treating them *as well as possible*. This can be done by protecting more important classes when the load grows and letting less important classes absorb the major part of the performance degradation. Such a change – simple and totally compatible with the principles of the

IP technology – can have a strong impact on IP networks behaviour, contributing in a positive way to the global satisfaction of IP service consumers and providers.

The multiple-class-best-effort proposal requires an integrated approach to scheduling and queue management. Although significant work exists in each of these fields (for instance, WFQ, W2FQ or Stochastic Fair Queuing [4], [19], in the area of packet scheduling, and RED and BLUE [12], [11] for queue management) these approaches do not work in an integrated way, in addition to having several known limitations and problems [18], [6], [27].

At the Laboratory of Communications and Telematics of UC (LCT-UC) the authors are working on a project which main goal is to study an alternative approach to support traffic classes in IP networks following the Diffserv framework, having in mind the multiple-class-best-effort paradigm. This effort can be divided into three main parts. The first one is to develop new mechanisms in Network Elements (NE), such as routers and switches, in order to support the multiple-class-best effort model referred before. The second is to conceive a way to select adequate paths for the forwarding of packets of different classes along the communication system. The third part is to implement an effective way to manage the system, which may include some form of traffic admission control at network edges as a way of controlling the communication system load. Naturally, the idea is to do all this in an integrated and coherent way.

This paper refers to the NE part of the referred work. It presents the foundations for a proposal of a new PHB named D3 (which stands for *Dynamic Degradation Distribution*). Specifically, section 2 presents the main characteristics and basic ideas that support D3, focusing the discussion on a router prototype developed to evaluate the feasibility and effectiveness of such a PHB. Section 3 presents the tests carried out on the prototype and analyses their results. Section 4 summarises the contributions and presents guidelines for further work.

2 A Qos-Capable Router Prototype

The purpose of D3 is to continuously re-distribute router resources in such a way that some classes will be protected from performance degradation at the expense of others. Classes have different sensitivity to QoS degradation - a given relative degradation on transit delay, for instance, can be unacceptable for some applications but well tolerated by others. D3 guarantees that the increase in transit delay and loss suffered by packets when the load grows reflects the sensitivity of classes to delay and loss degradation. In short, more sensitive classes will suffer less degradation with load increase, at the expense of a greater degradation suffered by less sensitive classes. The *degradation* is *dynamically distributed* among classes, and the control of this distribution is made independently for transit delay and loss.

For the purpose of quantifying the quality of service provided to classes a QoS metric presented in [28] was used. This metric is described in the following subsection.

2.1 QOS Metric for Packet Networks

The IP Performance Metrics Working Group (IPPM) of IETF [IPPM] is conducting a major effort for the definition of a set of standard metrics that can be applied to evaluate data delivery services. IPPM considers QoS characteristics (such as delay or loss) independently [24], [1], [2], [16]. In spite of IPPM and other work, there isn't, as far as the authors know, a comprehensive way to measure QoS considering different characteristics altogether. The metric developed at LCT-UC intends to address this challenge: to provide a broad and coherent view of the QoS given to applications, despite the heterogeneous nature of the characteristics being measured (for example delay or loss).

According to this metric, quality of service is quantified through a variable named *congestion index* - CI. For each class, there will be a CI related to transit delay and a CI related to loss.

The concept of *degradation slope* (DSlope) is used by the metric for the definition of classes' sensitivity to delay and loss degradation. A traffic class highly sensitive to QoS degradation for a given QoS characteristic will have a high DSlope associated to that characteristic. Figure 1 refers to three classes with different sensitivities to delay degradation (it would be the same if we were talking of loss). Classes with lower DSlope (measured in degrees) will be less sensitive to degradation, so their CI will grow slowly[1].

Using this metric one can say that resources are being shared in a fair way when the different classes have the same CI value related to delay and the same CI value related to loss. In fact, in that case, one can say that the impact of the degradation on applications is barely the same for all of them, despite the different absolute values of transit delay and losses experienced by their packets. Thus, when a router is dealing with n classes, in a given time interval $[t_i , t_{i+1}]$ formula (1) should stand.

$$CI_{avg}{}^{Class\_1}_{Delay}[t_i , t_{i+1}]= CI_{avg}{}^{Class\_2}_{Delay}[t_i , t_{i+1}]=\ldots= CI_{avg}{}^{Class\_n}_{Delay}[t_i , t_{i+1}] = CI_{avg}{}_{Delay}[t_i , t_{i+1}], \text{ and} \qquad \mathbf{(1)}$$

$$CI_{avg}{}^{Class\_1}_{Loss}[t_i , t_{i+1}]= CI_{avg}{}^{Class\_2}_{Loss}[t_i , t_{i+1}]=\ldots= CI_{avg}{}^{Class\_n}_{Loss}[t_i , t_{i+1}] = CI_{avg}{}_{Loss}[t_i , t_{i+1}]$$

The time interval $[t_i , t_{i+1}]$ should be relatively small - one can take as reference the time it takes to send a few tens of output link MTU-sized packets[2].

Figure 1 (left part), represents the status of a NE at a given instant in time (t_1) in what respects to the provision of QoS to classes considering only transit delay. CIs are equal for all the classes (CIt_1), corresponding to an average transit delay of d1, d2 and d3 for packets of class1, class2 and class3, respectively. At a given subsequent instant in time (t_2) the NE is experiencing a higher load. In this case, the NE degrades the performance given to all classes. The CIs are still equal for all classes but now they have a higher value (CIt_2) - Figure 1 (right part).

[1] For simplicity reasons, a linear variation has been considered. It is possible to use different variation types, for instance a logarithmic one - which is, perhaps, better suited to represent the sensitivity of human beings to QoS degradation [20].

[2] Experiments with a time interval corresponding to 8 MTU-sized packets were carried out and the observed overhead was low.

The degradation of absolute values of transit delay between t_1 and t_2 is $(d'_1 - d_1)$, $(d'_2 - d_2)$, and $(d'_3 - d_3)$, respectively for class 1, class 2, and class 3. It is clear that $(d'_3 - d_3)$ is much larger than $(d'_2 - d_2)$ which is, in turn, larger than $(d'_1 - d_1)$. So, class 1 – the one that is more sensitive to delay degradation – was protected when the load grew, at the expense of class 3 that absorbed the major part of the degradation.

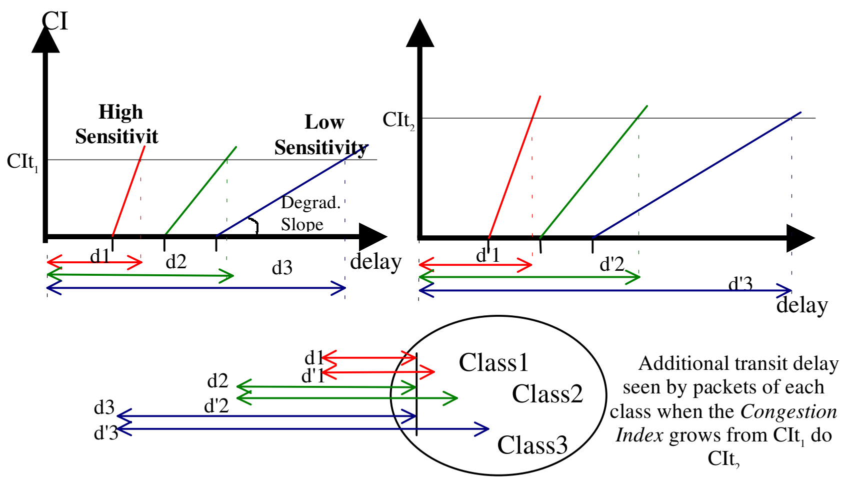

Fig. 1. Congestion indexes for three traffic classes with different sensitivities to delay degradation

2.2 Prototype Main Components and Architecture

In order to evaluate the proposed ideas and to prove their feasibility a QoS-capable router prototype, supporting the D3 per-hop behaviour, was implemented. Figure 2 presents its basic components: classifier; packet monitor; packet scheduler and packet dropper. The prototype allocates one independent queue to each class. The classifier/marker is responsible for determining each packet class and/or setting the information related to it in its header, following the strategy for IP packet classification/marking defined by the DSWG[3] [21].

The monitor is responsible for determining the average package transit delay and loss for each class, and for calculating the correspondent congestion indexes. By default, it is configured to perform the calculations each 10 ms, but it may be configured to use different values. The frequency of CI calculation should be defined as a trade-off between processing overhead and system responsiveness.

The core resources of a router are its transmitter capacity and its memory. The role of the packet scheduler and the packet dropper is to dynamically distribute those resources among traffic classes. The dynamic nature of the currently proposed

[3] DSWG – *Differentiated Services Working Group* of IETF [DSERV]

distribution contrasts with the much more common static allocation of resources to classes. According to the delay Congestion Indexes, the scheduler slows down the forwarding of packets of some classes and speeds up others. According to the loss Congestion Indexes, the dropper provides more memory to some queues removing it from others. The criterion to rule the dynamic distribution of resources is, as mentioned before, the equalisation of CIs.

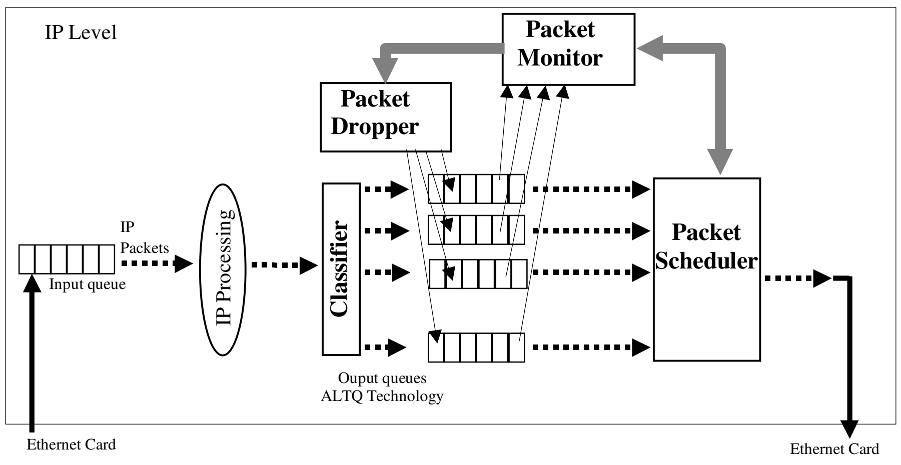

Fig. 2. Router prototype architecture

The basic scheduler operational principle is the one formally expressed in formula (1). This principle determined the scheduler design which was, nevertheless, influenced by some lessons learned with the study and test of other disciplines [27] as, for instance, the *weighted fair queueing* discipline [15], [8]. Early on, it was realised that it was not possible to use a pure *work-conserving* packet scheduler discipline in the platforms available at LCT-UC. In fact, the dynamics of the systems in use tends to serialise the appearance of packets in queues [27], [29], inhibiting any capacity of such disciplines to effectively differentiate traffic. One advocate, in agreement with other research work [17], that a scheduler should be able to pick the best part of the work-conserving and non-work-conserving worlds, namely i) the simplicity and good level of resource utilisation and ii) the capacity to maintain some part of the resources available for high priority traffic. The basic algorithm that supports the proposed scheduler is presented in Appendix A and detailed in [25].

The dropper developed at LCT-UC [26] includes also some important lessons learnt from experiments and from the work of other researchers [6], [18], [7], [16]. It avoids dropping TCP packets in order to protect TCP traffic from UDP traffic – TCP has its own mechanisms for congestion avoidance triggered by packet drops whereas UDP has no such mechanisms. In addition, UDP and TCP packets are randomly dropped. Most important, the dropper removes memory from some queues and allocates it to others, according to the load and QoS class needs. The criterion that rules the dynamic reallocation of memory is the equalisation of the classes' drop

Congestion Indexes. The basic algorithm that supports the proposed dropper is presented in Appendix B.

The FreeBSD operating system [10], patched with ALTQ technology [8], was used to implement the router prototype. Its protocol stack was modified in order to include the components referred above. The scheduler and dropper were developed separately, and submitted to specific tests [25], [26]. They were integrated in the prototype only at a subsequent stage of the project. In the next section the first test suite carried out on the router prototype is presented.

3 Prototype Tests

The main goal of the tests presented in this section is to prove the feasibility and effectiveness of the concepts proposed in this paper.

The testbed used to carry out the tests consisted of a small isolated network with 5 Intel Pentium PC machines configured with a Celeron 333Mhz CPU, 32 MB RAM and Intel EtherExpress Pro100B network cards. The prototype router ran FreeBSD 2.2.6, patched with ALTQ version 1.0.1, with 64MB RAM. Three hosts generated traffic directed towards a destination host through the prototype router. Each host only generated traffic of a given class, in order to guarantee the independence of the generated packet streams.
The tests were performed with the aid of two basic tools:

- *Nttcp* [22] - to generate UDP packets with a given length and at a given rate.
- *QoStat* [3] - to monitor the kernel, namely the operation of the installed prototype. This tool was developed at LCT-UC. With it, it is possible to obtain data related to QoS provision in real time and also to change the most important operational parameters of the prototype.

3.1 Tests - Fixing the Classes' Sensitivity to Loss Degradation

In the first set of tests, the classes' sensitivity to loss degradation was set to a fixed value, corresponding to a degradation slope of 45 degrees for all classes. Conversely, class 1 and class 3 sensitivity to delay degradation varied with time: class 1 became less sensitive, class 3 became more sensitive, and class 2 maintained its sensitiveness. Table 1 presents the progression of each class degradation slope with time.

Time (s)	Class 1		Class 2		Class 3	
	Delay DSlope	Drop DSlope	Delay DSlope	Drop DSlope	Delay DSlope	Drop DSlope
0	45	45	45	45	45	45
25	40	45	45	45	50	45
50	35	45	45	45	55	45
75	30	45	45	45	60	45
100	25	45	45	45	65	45

Table 1. Evolution of classes' DSlopes with time during the test

The *QoStat* tool was used in order to dynamically change each class' *delay DSlope* after each 25-seconds time interval - the changes were always applied simultaneously to all classes. As the objective was to evaluate the global prototype behaviour specifically under high load conditions, *nttcp* was used to generate fixed length

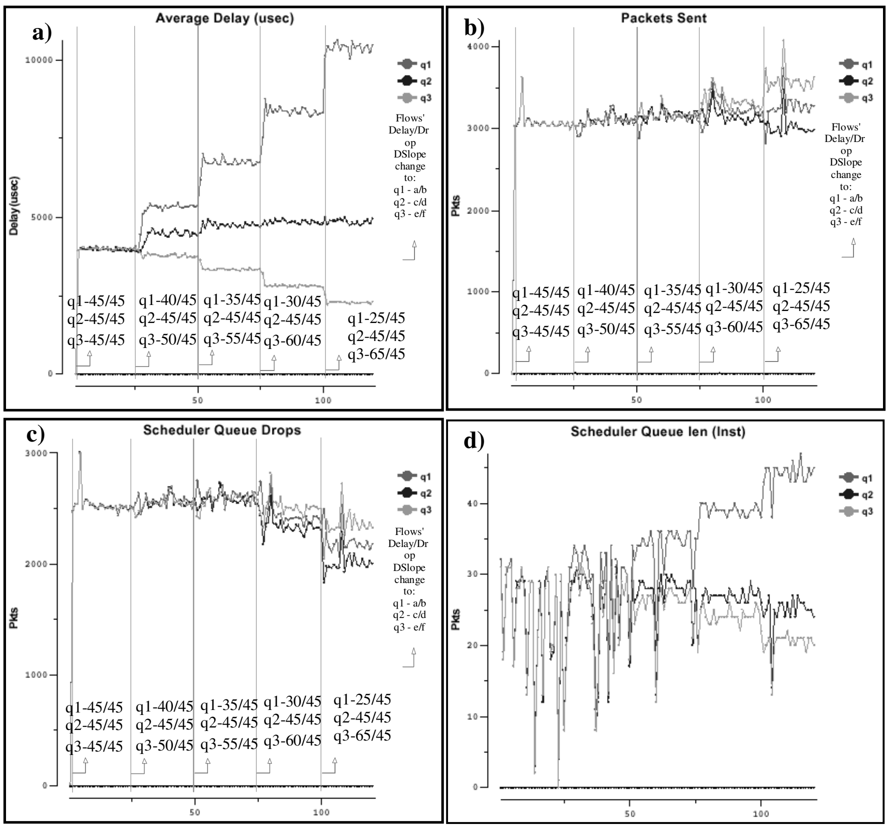

Fig.3. Variation of the following values with classes sensitiveness to degradation: a) Average IP transit delay (over 1 second intervals); b) Number of packets sent per second; c) Number of dropped packets per second; d) Instantaneous IP output queue length (sampled every second)

packets of 1400 bytes, at about 60 Mbps (for each class). The results of this test are presented in figure 3.

When the sensitivity to delay degradation changes, the scheduler reacts re-distributing the processing capacity among classes - the sharp transition shown in the graph of Figure 3a) reveals a good responsiveness of the system to the change. As class 3 becomes more sensitive to loss degradation, its packets experience smaller delays. The opposite happens with class 1 as it becomes less sensitive to degradation.

Conversely, because the classes sensitivity to drop degradation has the same value, the number of processed packets per unit of time is roughly the same for all of them (Figure 3b). Nevertheless, we can see that as the difference between the best and the worst *delay DSlope* increases the system starts to diverge a little. This is due to a lesser effectiveness of the algorithm that maps the difference between the measured CI for each class to the scheduler operational adjustment. The algorithms that convert CIs differences on scheduler and dropper operational adjustments are very simple and constitute one of the subjects for further study.

We repeated the tests presented before, now changing the fixed values configured for the drop degradation slope. Instead of configuring the value 45 for all classes, we configured the values 30, 45 and 60, for class 1, class 2 and class 3, respectively. Those values were still maintained fixed during the experience. The average packet transit delay and number of sent packets measured under those conditions are shown in figure 4. One can see that the router is still able to react coherently when the delay degradation sensitivity is changed, but now differentiating classes in terms of their packets average transit delay, according to the configured *delay DSlope*.

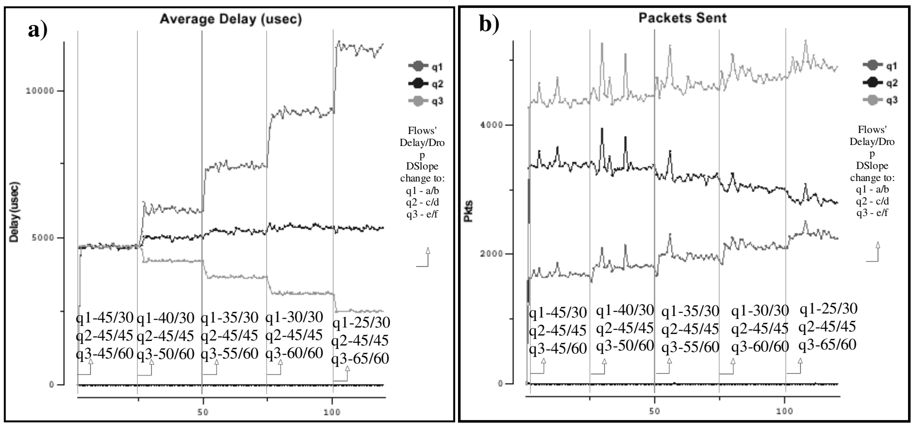

Fig.4. Variation of the following values with classes sensitiveness to degradation: a) Average IP transit delay (over 1 second intervals); b) Number of packets sent per second;

3.2 Tests - Fixing the Classes' Sensitivity to Delay Degradation

In the following step, a complementary set of tests was made. This time the sensitivity of the classes to delay was fixed (with a degradation slope of 45 degrees for all classes), and their sensitivity to loss degradation was made to change during the course of the tests. Table II presents the progression of each class degradation slope with time.

Time (s)	Class 1		Class 2		Class 3	
	Delay DSlope	Drop DSlope	Delay DSlope	Drop Dslope	Delay DSlope	Drop DSlope
0	45	45	45	45	45	45
25	45	40	45	45	45	50
50	45	35	45	45	45	55
75	45	30	45	45	45	60
100	45	25	45	45	45	65

Table 2. Evolution of classes' DSlopes with time during the test

The results of this test are presented in figure 5. When the sensitivity to losses

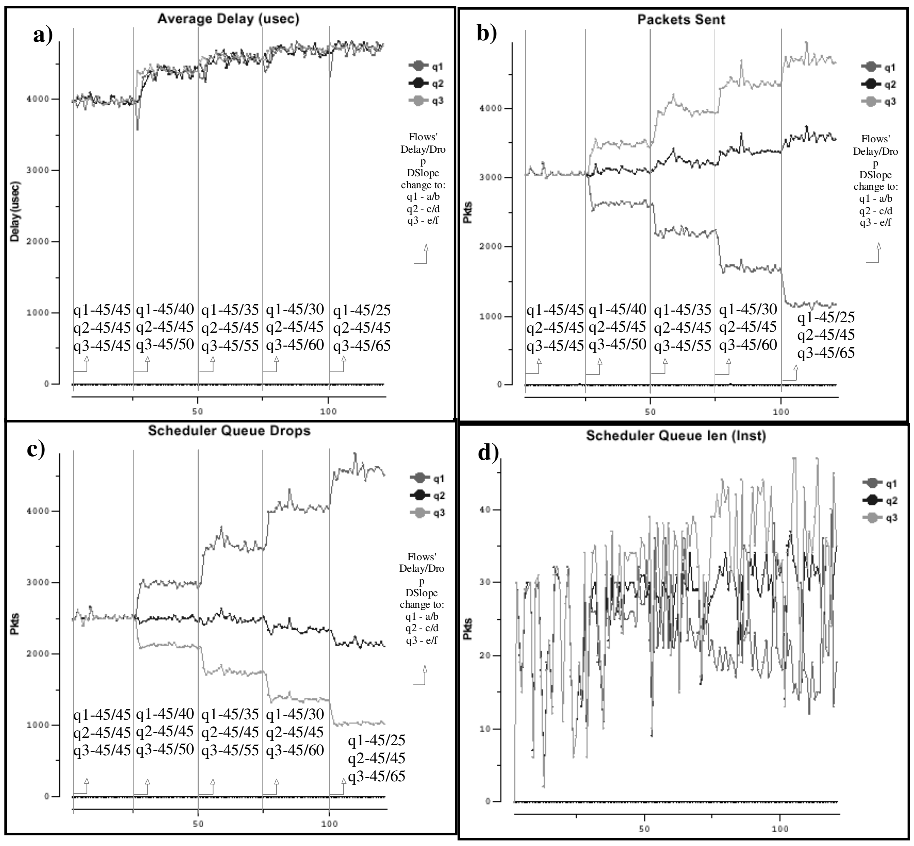

Fig.5. Variation of the following values with classes sensitiveness to degradation: a) Average IP transit delay (over 1 second intervals); b) Number of packets sent per second; c) Number of dropped packets per second; d) Instantaneous IP output queue length (sampled every second)

changes, the drop effort is re-distributed among classes. As class 3 becomes more sensitive to loss degradation, less of its packets are dropped, and so more of its packets are processed. The opposite happens with the class becoming less sensitive to loss degradation. This is a consequence of the re-distribution of the maximum queues' length among classes. When class 3 becomes more sensitive to loss it receives a greater share of the global queue space (Figure 5d).

4 Main Contributions and Future Work

This paper presented an approach to support traffic classes in IP networks, based on the Differentiated Services framework. The basic idea is to dynamically share the resources available in the network among existing classes, according to the measured quality of service and to the classes' sensitivity to QoS degradation. When applied to routers, the proposal leads to a new per-hop behaviour, moving from the current single-class-best-effort paradigm in use on the Internet to a multiple-class-best-effort paradigm. The approach is supported by a QoS metric developed at LCT-UC, which was also presented in this paper.

One of the main advantages of the presented proposal is that it provides ways for an efficient use of communication resources, considering the classes needs related to performance. In addition, the approach does not imply any substantial changes to current IP technologies, which is another important advantage, if not the most important one. All the classes are treated as well as possible, considering that they have different sensitivity to performance degradation.

To evaluate the presented ideas a router prototype was implemented. Its main components are a packet classifier, a monitor, a scheduler and a dropper. The tests that were carried out on the prototype proved the feasibility of the underlying ideas and showed that, despite the use of simple algorithms, the prototype revealed a stable and effective behaviour.

Currently, as on-going work, the authors are studying ways to extend the presented ideas beyond the network element. More specifically, mechanisms for QoS-aware dynamic routing of traffic classes and for admission control based on the values of congestion indexes are being implemented.

References

1. G. Almes, S. Kalidindi, M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999. ftp://ftp.isi.edu/in-notes/rfc2679.txt
2. G. Almes, S. Kalidindi, M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, September 1999. ftp://ftp.isi.edu/in-notes/rfc2680.txt
3. Antonio Alves, Goncalo Quadros, Edmundo Monteiro, Fernando Boavida, 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
4. J.C.R. Bennett and H. Zhang, ``WF2Q: Worst-case Fair Weighted Fair Queueing", INFOCOM'96, Mar, 1996.
5. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998. ftp://ftp.isi.edu/in-notes/rfc2475.txt
6. Thomas Bonald, Martin May, Drop Behavior of RED Bursty and Smooth Traffic, in Proceedings of IWQoS'99, London, May 31-June 4, 1999.

7. B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, Re-commendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 98. ftp://ftp.isi.edu/in-notes/rfc2309.txt
8. Kenjiro Cho, 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
9. http://www.ietf.org/html.charters/diffserv-charter.html.
10. http://www.fbsd.org
11. W. Feng, D. Kandlur, D. Saha, K. Shin, "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
12. S. Floyd et al., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993. http://www-nrg.ee.lbl.gov/floyd/papers.html.
13. J. Gollick, “Catching the Multimedia Wave”, *Data Communication*s, March 1999.
14. http://www.ietf.org/html.charters/ippm-charter.html.
15. Srinivasan Keshav, On the efficient implementation of fair queuing, Internetworking: Research and Experience, September 1991.
16. R. Koodli, R. Ravikanth, One-way Loss Pattern Sample Metrics, Internet Draft, October 1999. http://www.ietf.org/internet-drafts/draft-ietf-ippm-loss-pattern-02.txt
17. Jorg Liebeherr, Erhan Yilmaz, Workconserving vs. Non-Workconserving Packet Scheduling: An Issue Revisited, in Proceedings of IWQoS'99, London, May 31-June 4, 1999.
18. Martin May, Jean Bolot, Christophe Diot, Brian Lyles, Reasons not to deploy RED, in Proceedings of IWQoS'99, London, May 31-June 4, 1999.
19. P. McKenney, Stochastic Fairness Queueing, in Proceedings of IEEE INFOCOM, San Francisco, California, June 1990.
20. Edmundo Monteiro, Fernando Boavida, Gonçalo Quadros, and Vasco de Freitas, "Specification, Quantification and Provision of Quality of Service and Congestion Control for New Communication Services", Proceedings of the 16th AFCEA Europe Brussels Symposium & Exposition , AFCEA/IEEE ComSoc, Les Pyramides, Brussels, Belgium, 18-20 October 95.
21. Nichols, K., et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
22. New TTCP Program, http://www.leo.org/~bartel/nttcp/
23. Quality of Service Routing in the Differentiated Services Framework, Marília Oliveira, Bruno Melo, Gonçalo Quadros, Edmundo Monteiro, submitted to IWQoS 2000.
24. V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998. ftp://ftp.isi.edu/in-notes/rfc2330.txt
25. Goncalo Quadros, António Alves, Edmundo Monteiro, Fernando Boavida, 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
26. Goncalo Quadros, António Alves, Edmundo Monteiro, Fernando Boavida, A New Dropping Scheme for IP Routers, Technical Report, CISUC, February 2000. http://lct.dei.uc.pt/papers/NewDropper_TR.PDF
27. Goncalo Quadros, António Alves, Edmundo Monteiro, Fernando Boavida, How Unfair can Weighted Fair Queuing be?, in proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), Antibes, France, 4-6 July 2000. http://lct.dei.uc.pt/papers/HUnfairWFQ_TR.PDF
28. Gonçalo Quadros, Edmundo Monteiro, Fernando Boavida, “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.

29. Fulvio Risso, Panos Gevros, "Operational and Performance Issues of a CBQ router", *Computer Communication Review*, Volume 29, Number 5, October 1999.

APPENDIX A - SCHEDULER GENERAL ARCHITECTURE

There are two nuclear parameters related to the scheduler operation, which characterize each queue[4]: **`dequeue_time`**, which determines the instant of time after which the next packet of the class can be processed; and **`x_delay`**, which is used to update **`dequeue_time`**. **`x_delay`** determines the time interval (in µs) that must elapse between the processing of consecutive packets of a given class.

The following points concisely present the scheduler operation:

1. the scheduler visits each queue in a round robin fashion;
2. in each visit, it compares the current time with the queue's **`dequeue_time`**; if the former is greater than the latter, a packet is processed;
3. for the active class[5] that is most sensitive to delay degradation (which is named *class_1*), **`x_delay`** is always zero; so the scheduler behaves as a work conserving scheduler for this class;
4. for the remaining classes (*classes_r*), **`x_delay`** will be greater than zero; the scheduler behaves as a non-work-conserving scheduler.

The scheduler operation is logically presented in Figure A.1.

The **`x_delay`** of the remaining classes (classes_r) are dynamically adjusted so that the respective congestion indexes equalize the class_1's congestion index. CIs are calculated and **`x_delay`**s are adjusted every *n* ms (*n* is configurable and experiments were made for *n* equal to 10 and 100 ms). All the packets processed by the scheduler during that time period are considered for the average transit delay calculation and for the respective CI calculation (the short term CI). Some tests using higher frequencies for CI calculation were also carried out, showing that the used strategy does not imply a significant overhead.

APPENDIX B - DROPPER GENERAL ARCHITECTURE

There are two important parameters related to the dropper operation, which characterize each 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.

[4] Each class has its own exclusive queue.

[5] A class with packets recently processed by the scheduler. The frequency of evaluation of active classes is configurable, and should be closely related to the $[t_i, t_{i+1}]$ time interval referred in formula (1).

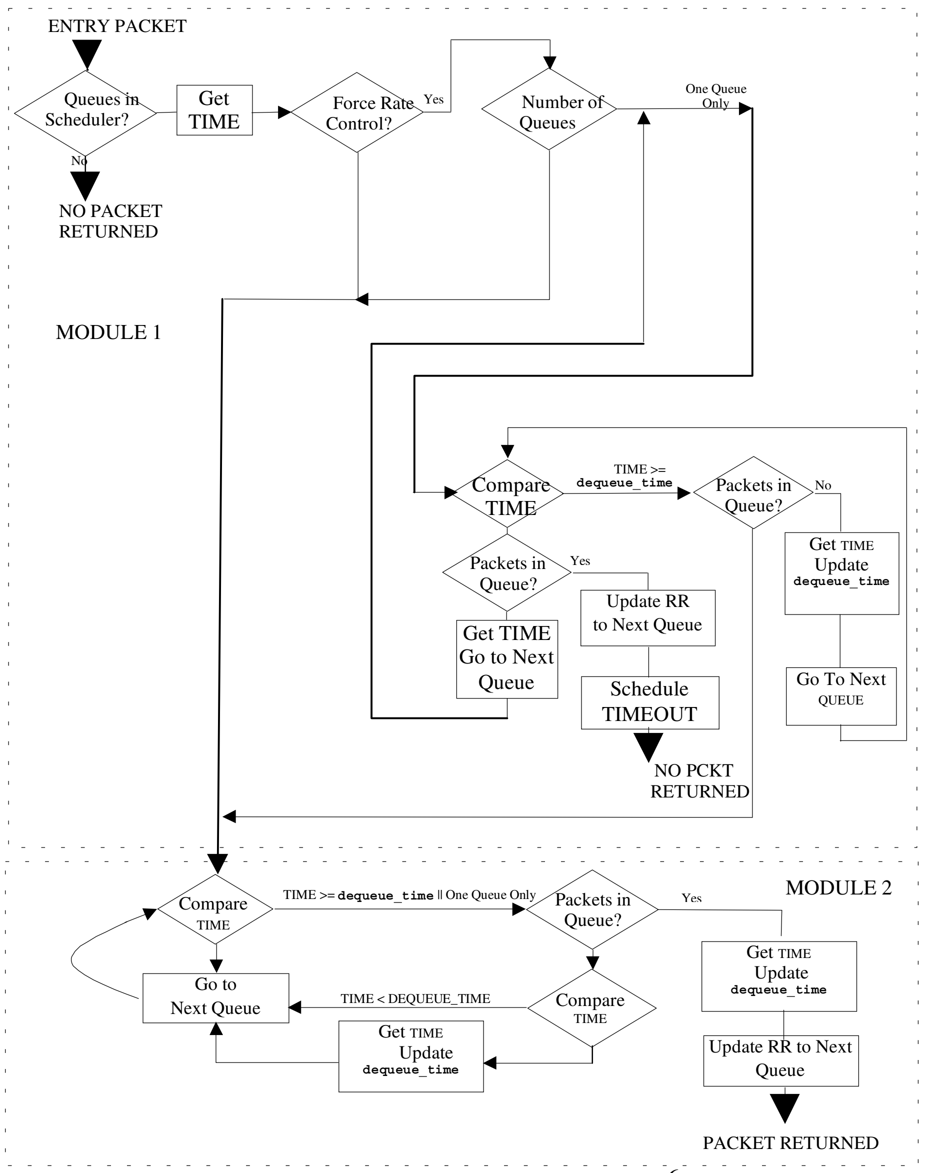

Fig. A1. Scheduler logical presentation[6]

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

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 used is 8

[6] This diagram reflects the two scheduler-development phases and, thus, it is not optimized.

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 the limit to be respected.

For each queue, TCP packets will only be dropped if they exceed a given percentage of the total number of packets. Both TCP and UDP packets are randomly dropped. The Dropper is logically presented in Figure B.1.

The virtual limits for each class are dynamically adjusted, according to the loss CI measured for them (the goal is to maintain the same value of CI for all classes). The sum of the classes' `q_virtual_limit` is a constant (MAX_PCKS_ONQ, which corresponds to the maximum number of packets that can be stored in the queue system at any given time). The queue with the worst CI receives some amount of buffer space taken from the queue with the best CI. The buffer amount transferred from one queue to the other depends on the CIs difference. The timings for loss CI calculation follow the ones for delay CI calculation.

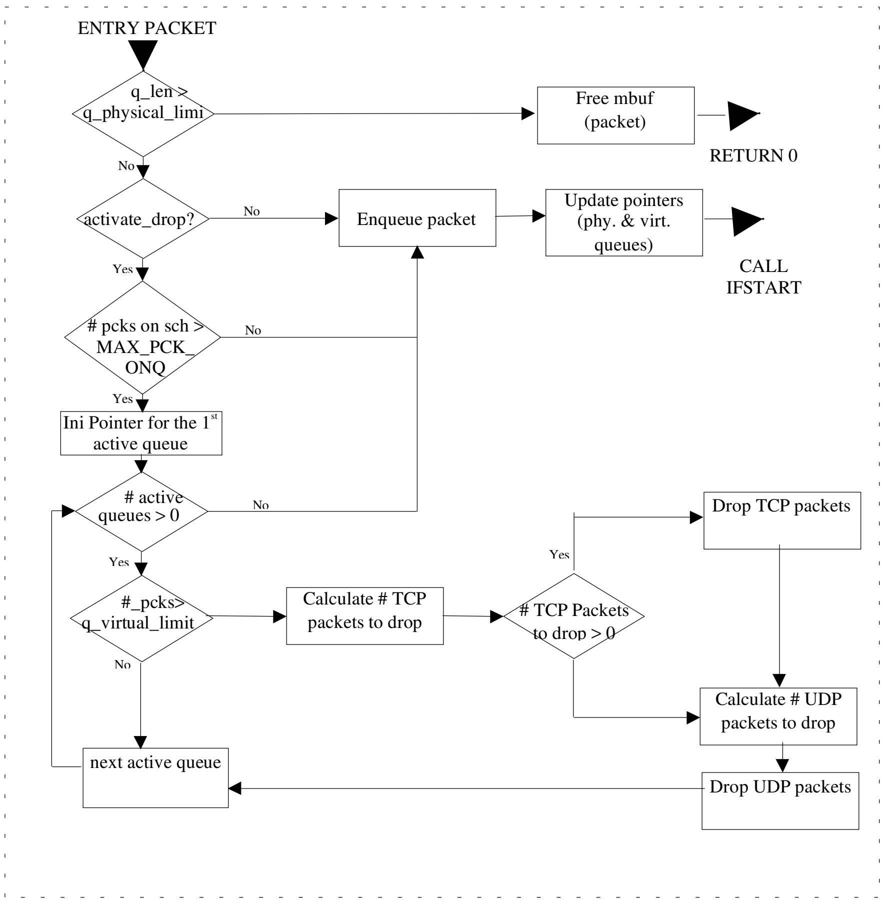

Fig. B.1. Dropper logical presentation