How Unfair can *Weighted Fair Queuing* be?

Goncalo Quadros, Antonio Alves, Edmundo Monteiro, Fernando Boavida
CISUC – Centro de Informática e Sistemas da Universidade de Coimbra
Departamento de Engenharia Informática
{quadros, aalves, edmundo, boavida}@dei.uc.pt

Abstract

This paper presents a study carried out on a Weighted Fair Queuing implementation for Unix routers - the WFQ implementation of the ALTQ project. It shows the WFQ/ALTQ weaknesses and explains why we cannot expect an interesting behavior from a system using such a scheduler.

The conclusions here presented are supported by a set of tests using UDP traffic only. With a tool developed in our laboratory, we were able to show that changing the classes' weights does not necessarily result on a different Quality of Service for each of the existing classes. To achieve this differentiation, the lengths of the queues which serve the scheduler (one for each class) must be increased beyond reasonable values.

We found that the low-level dynamics of FreeBSD systems practically turns WFQ schedulers useless. The same is applicable to any other work-conserving discipline. Thus, an important conclusion of this paper, is that one must design very carefully the platforms that support work conserving disciplines in order to expect adequate behaviors from those systems, in terms of QoS provision.

1. Introduction

At the Communications and Telematics Laboratory of the University of Coimbra we have been working in a new service model applicable to the Internet environment. The idea is to conceive a quite simple model able to differentiate the quality of service given to different traffic classes. By simplicity we mean a model easy to implement, supported to a great extent by the technologies and concepts currently in use on the Internet.

The packet scheduler is a very important component in our model. From the beginning we knew that the service model usefulness and effectiveness would mainly depend on the behavior of this component. In fact, the capacity to differentiate the performance given to different traffic classes, in a controllable way, strongly depends on the scheduler operation.

Our lab is equipped with Intel/FreeBSD hosts. Our foremost approach was to look for schedulers developed for this platform, able to substitute the traditional FIFO mechanism and to give systems the characteristics we wanted. We admitted that the *weighted fair queuing* implementation developed in the ALTQ project – WFQ/ALTQ [Cho98, Cho99], would be an interesting starting platform. Thus, we submitted this scheduler to a diverse set of tests. The purpose was to determine the real WFQ/ALTQ capacity to differentiate and control the performance provided to different traffic classes.

This document presents and analyses the tests made to WFQ/ALTQ using UDP traffic flows only. Section 2 presents the used testbed and the general approach to the tests. Section 3 presents the test results, which are discussed in section 4. Section 5 concludes this paper and presents some directions for future work.

2. General test environment

The testbed used to perform the tests is presented in figure 1. It consists 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 PC Router (named HOST_R in the figure) runs FreeBSD 2.2.6, patched with ALTQ version 1.0.1, with 64MB RAM. HOST_S1 and HOST_S2 are used to generate traffic directed towards HOST_D, through HOST_R.

The main motivation of the tests that were carried out was to determine the effectiveness of the WFQ/ALTQ capacity to provide different qualities of service to different classes of traffic. We wanted to know if – and exactly how – the scheduler could be used to control the performance given to different types of traffic.

As in our testbed we had two hosts available for traffic generation, we used them to create two independent data flows, simulating two independent classes of traffic. In the tests presented here we exclusively used UDP traffic, as this is the most favorable situation to obtain a good behavior related with QoS provision with this type of schedulers. The flows were constituted by long packets (1400 bytes), to enable loads as high as possible.

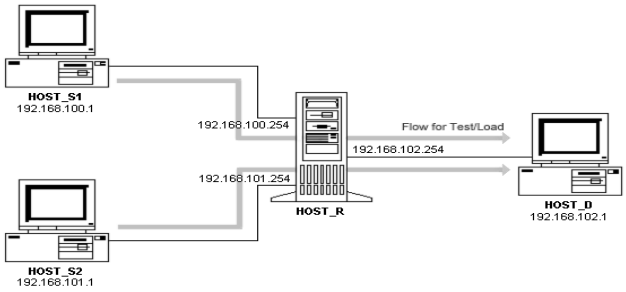

Figure 1 – The testbed

We used two tools to perform our tests: *Nttcp* [NTTCP] for generating data flows – each flow competing for all the resources available in the communication system; *QoStat* for setting scheduler parameters and monitoring the system, namely its parameters related with QoS.

QoStat was implemented in our laboratory and it has been fundamental for our work. Through it, we can dynamically change the most important operational parameters of the scheduler, and continuously get the most important values related to QoS provision. Thus, *QoStat* turns the evaluation of a scheduler easier. All the graphics presented in this paper were produced using this tool. A deeper presentation of its capacities can be found in [Alves99].

Figure 2 presents a logical scheme of the tested system. We will return to this figure below, in order to better understand the meaning of some measurements. For now we just want to refer that we have modified the drop mechanism used by WFQ/ALTQ. In fact, we found that the mechanism distributed with the WFQ/ALTQ biases the operation of the scheduler [Quadros99a], so we decided to substitute it by a simple tail-drop one.

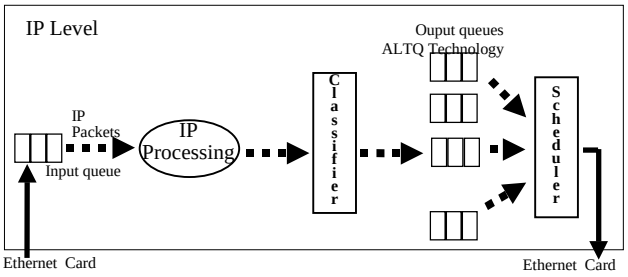

Figure 2 – Prototype logical architecture

3. WFQ/ALTQ testing with UDP exclusive traffic

3.1 The tests and their results

The general test strategy can be presented in three points:

1. to use two different classes, composed of traffic generated by two independent hosts;
2. to fix the weight associated with one of the classes (we chose class 1[1] and the weight 100), and
3. to vary the weight associated with the other class (we chose class 3 and successively the weights 30, 70, 100, 500 and 1000)[2] after each time interval of 25s.

In the first set of tests we exclusively used UDP traffic. We did not change the default WFQ/ALTQ configuration, which means, maximum queue length equal to 50 packets and a base quota[3] of 512 bytes. The results are presented in figure 3.

The figure illustrates the following results:

- Packets average transit delay at IP level, measured over 1 second time intervals. A packet transit delay is the time that lasts between its enqueuing at IP input queue, and its dequeuing from the IP output queue (see figure 2);
- The number of IP packets sent by the output interface per second;
- The number of IP dropped packets per second;
- The instantaneous IP output queue length, sampled every second.

As it is clearly seen, under the present test conditions the scheduler does not show any capacity to differentiate traffic. The variation of the class weight does not have any impact on the way the correspondent traffic is treated.

Given the complete absence of traffic differentiation capacity, and the very high level of dropped packets, we decided to repeat the tests augmenting the maximum IP output queue lengths. Figure 4 presents the results. As can be seen, the increase of the maximum length of the referred queues to 200 (instead of the previous 50) has a very positive impact on the capacity of traffic differentiation of the scheduler. Naturally, it also results in much higher transit delay.

In order to confirm, and better evaluate, the influence of queue length in the WFQ/ALTQ behavior, we executed the following test: using the same testbed, we assigned two independently generated data flows to two different classes with fixed weights of 100 and 1000; given this, we changed the maximum IP output queue lengths (for both queues) from 50, successively to 100, 150 and 200 packets, respectively after 25, 50 and 75 seconds. The results are shown in figure 5 and clearly reveal a relation between output queue length and WFQ/ALTQ capacity to differentiate traffic.

[1] Whose queue name in figures is q1.

[2] We didn't use class 2 in the tests. Moreover, class 0 is always shown – because it is the default class. As we are only using traffic of classes 1 and 3 the measured values connected to class 0 are always null.

[3] Bytes processed in each scheduler visit to a queue with weight equal to 100.

3.2 Analysis of the tests results

To understand why the WFQ/ALTQ scheduler is unable to differentiate traffic it is important to make some comments about its operational characteristics. It is, for instance, important, to realize that this is a work-conserving scheduler, which means that whenever, in a given iteration, the scheduler finds packets in only one queue, it will process them at grossly the maximum possible rate. Thus, whenever the packets belonging to different classes reach the IP level alternately, or not simultaneously, they will be treated the same way, independently of the weight assigned to classes.

We decided to verify if this was the case. In order to do this, we modified the *QoStat* tool to monitor also the percentage of time the scheduler found packets in one, more than one or none of the queues. Table 1 presents the results when using IP output queues with maximum lengths of 50, 100 and 200 packets.

Max IP output queue length / Situation	50 packets		100 packets		200 packets	
	# times	%	# times	%	# times	%
Queues without packets	2016	18	0	0	0	0
One queue with packets	5290	47	1368	15	0	0
More than one queue with pcks	3900	35	7848	85	9100	100

Table 1 – Number of times (and percentage) the scheduler finds packets in none, one or more than one queue, for 1 second time intervals

We can see that when the maximum length of IP output queues is 50 packets, only in a minority of occasions the scheduler finds packets to send in more than one queue. This happens even when using very high loads composed of UDP traffic (which does not adapt to the available capacity in the communication system). Given this, the incapacity of the scheduler to differentiate traffic is not a surprise.

Conversely, we can see that when the maximum IP output queues are 200 packets long, the scheduler finds packets in more than one queue at all times. As we allow the storage of more packets on the queues, the probability of finding packets simultaneously on those queues is higher. Therefore, the effective capacity of the scheduler to differentiate traffic in these conditions is also not a surprise.

What is important to highlight is that, as seen above, with maximum IP output queue of 200 packets the WFQ is, in fact, able to effectively differentiate traffic.

By effective capacity to differentiate traffic we mean a good capacity to control the QoS provided to traffic classes. This capacity is observable in figure 4. In fact, it is possible to see that we can easily change the QoS given to classes varying their weights. It is also possible to see that there is proportionality between the weight assigned to classes and the performance they get.

At this time it is interesting to deeper discuss the results shown in figure 5. This figure shows that there are no significant changes on the number of processed and dropped packets per second when we evolve the maximum length of IP output queues from 150 to 200 packets. Conversely, those values vary when we change the referred length from 50 to 100 packets and, after, to 150 packets.

This happens because when the maximum IP output queue length is 150 packets, like when it is 200 packets, the scheduler always finds packets in both queues. So, in either case, the ratio between packets sent (or dropped) in each class is equal to the ratio between the classes' weight (1:10). The only noticeable difference is a natural one: with longer queues the average transit delay is higher. Thus, it is evident that in the conditions of the tests it can only be negative to have IP maximum output queue lengths greater than 150 packets.

When the maximum IP output queue length is lower than 150, the scheduler doesn't always find packets in both queues. Thus, it cannot always process 10 times more packets from one queue than from another. So, the average ratio between processed packets from the two classes stands behind 1:10, and, as discussed before, it is 1:1 if the maximum IP output queue length is 50 packets.

This behavior is unacceptable from a fair scheduling point of view, taking into account that even 50 packets is a too big maximum for the queue lengths. In fact, we are talking about only one out of several output queues and in current routers the total capacity of the output queuing systems is typically 40-50 packets. For instance, in a router using 4 classes with 50 packets maximum output queue lengths, we would have potentially 4 times more packets in the output queuing system than what we would have in current routers. As a consequence the transit delay would be much higher.

It is evident that the lack of simultaneous presence of packets in both queues is, in fact, responsible for the scheduler inability to differentiate traffic. Figure 5, in conjunction with table 1, makes this clear. But what can be the cause of this? This is a difficult question to answer, even more because of the conditions used to make the tests (very high loads and UDP traffic). We can only say that the dynamics of the operating system and the protocol stack, including NIC hardware and software, is causing some kind of packet serialization in queues. In our tests, with normal queues lengths, they tend to appear

on different queues at different times. Only deeper (and harder) tests could shed some light on the exact cause of this, which, much probably, will vary from system to system (in [Bennett99] the influence of NIC buffers capacity on this kind of issues is discussed). It is, nevertheless, important to highlight that there is no reason to suppose that this kind of problems couldn't happen in WFQ implementations other than the ALTQ one.

To finish this section we would like to say that we have made some other tests: using a different traffic mix (TCP traffic only and a TCP-UDP traffic), different packet sizes and different loads. Because of the lack of space we will not present their results here (they will be published soon). Nevertheless, we can say that all of them corroborate the analysis presented here.

Concluding this analysis, we decided not to use the WFQ/ALTQ scheduler in our model, as it proved highly unsatisfactory.

4. Conclusion and Future Work

At LCT-UC laboratory we are working-on a new IP service model and we needed to choose a scheduler to be used on it. Because of its simplicity, availability, and the characteristics we thought it had, we decided to test the WFQ/ALTQ scheduler, to support its eventual choice. In this paper the results of these tests are presented.

The tests show that we can not expect adequate QoS capabilities from routers using the WFQ/ALTQ scheduler. This happens because, normally, the scheduler doesn't find packets in more than one queue simultaneously, even when the most favorable conditions hold. As this is a work-conserving scheduler, when that doesn't happen the scheduler is unable to differentiate traffic.

Nevertheless, we show that for huge maximum output queue lengths the scheduler is, conversely, able to differentiate traffic classes. In this case, the probability of finding more than one queue with packets is higher, which is determinant to the scheduler better QoS behavior. The problem is that the transit delay is also higher, too much higher.

Summing up, we have shown that at least with the ALTQ implementation we can not expect good results of the *weighted fair queuing* discipline. Its work-conserving nature is a killing characteristic when the system dynamics results in difficulties to guarantee the simultaneous presence of packets in more than one queue. We noticed that the same type of problems can possibly appear in other types of platforms, such as dedicated router platforms.

Given the results of the tests presented here we decided to implement a new scheduler. Our idea is to pick the best part of both worlds: to use a work-conserving scheduling when it makes sense and to use a non-work-conserving scheduling when work-conserving scheduling destroys the desired QoS behavior. This strategy was also considered by other researchers [Liebeherr99], and work already under way in our laboratory has proved – with promising results – that this is possible. In [Quadros00b] these results are presented and discussed.

Acknowledgements

This work was partially supported by the Portuguese ministry of science and technology (MCT), under the program praxis XXI - PRAXIS/P/EEI/10168/1998.

References

[Alves99] Antonio Alves, Goncalo Quadros, Edmundo Monteiro, Fernando Boavida, QoStat – A Tool for the Evaluation of QoS Capable FreeBSD Routers, Technical Report, CISUC, July 99.

[Bennett99] Jon Bennett and Christopher Stein, 2-Bit Diff-Serv Routing in the ALTQ FreeBSD Kernel, Harvard University, January 1999. orvieto.eecs.harvard.edu/ALTQ_Diff3.html

[Cho98] 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. www.csl.sony.co.jp/person/kjc/kjc/papers/usenix98

[Cho99] Kenjiro Cho, Managing Traffic with ALTQ, in Proceedings of USENIX 1999 Annual Technical Conference: FREENIX Track, Monterey CA, June 1999. www.usenix.org/events/usenix99/technical_freenix.html

[NTTCP] New TTCP Program, www.leo.org/~bartel/nttcp/

[Liebeherr99] Jorg Liebeherr, Erhan Yilmaz, Workconserving vs. Non-Workconserving Packet Scheduling: An Issue Revisited, in Proceedings of IWQoS'99, London, May 31-June 4, 1999.

[Quadros99a] Goncalo Quadros, Antonio Alves, Edmundo Monteiro, Fernando Boavida, The role of packet-dropping mechanisms in QoS Differentiation, Technical Report, CISUC, July 99. lct.dei.uc.pt/papers/RPDM_TR.PDF

[Quadros00b] Goncalo Quadros, António Alves, Edmundo Monteiro, Fernando Boavida, An Effective Scheduler for IP Routers, in proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), Antibes, France, 4-6 July 2000. lct.dei.uc.pt/papers/NewScheduler_TR.PDF

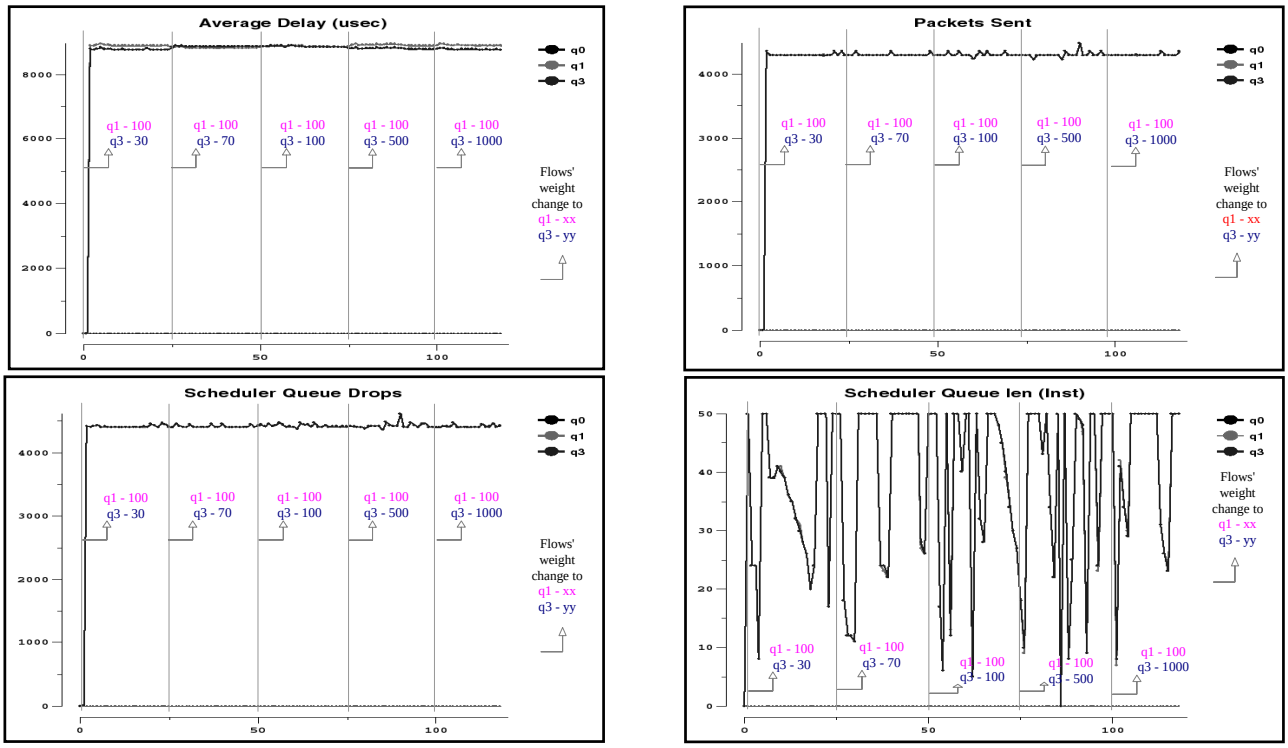

Figure 3 – Variation of the following values with class weight: ↖) Average IP transit delay (over 1 second intervals); ↗) Number of packets sent per second; ↙) Number of dropped packets per second; ↘) Instantaneous IP output queue length (sampled every second)

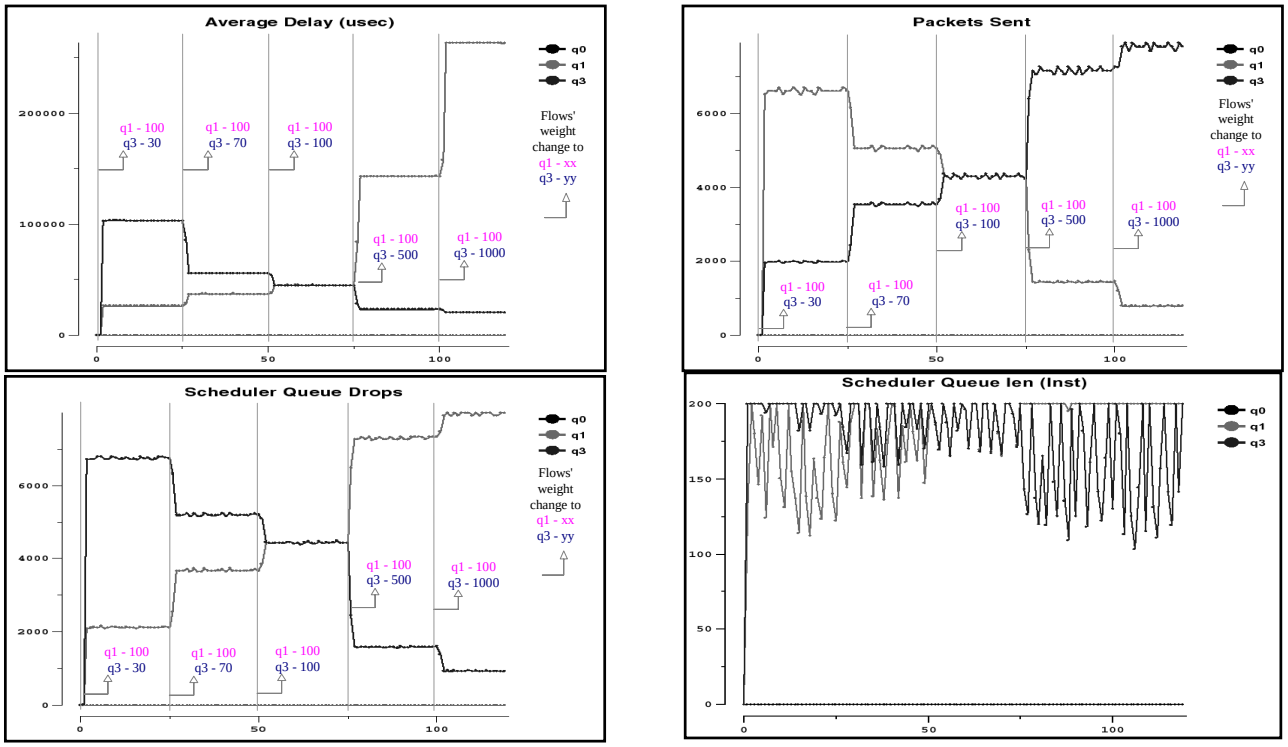

Figure 4 – Variation of the following values with class 3 weight, for IP output queues with maximum length of 200 packets: ↖) Average IP transit delay (over 1 second intervals); ↗) Number of packets sent per second; ↙) Number of dropped packets per second

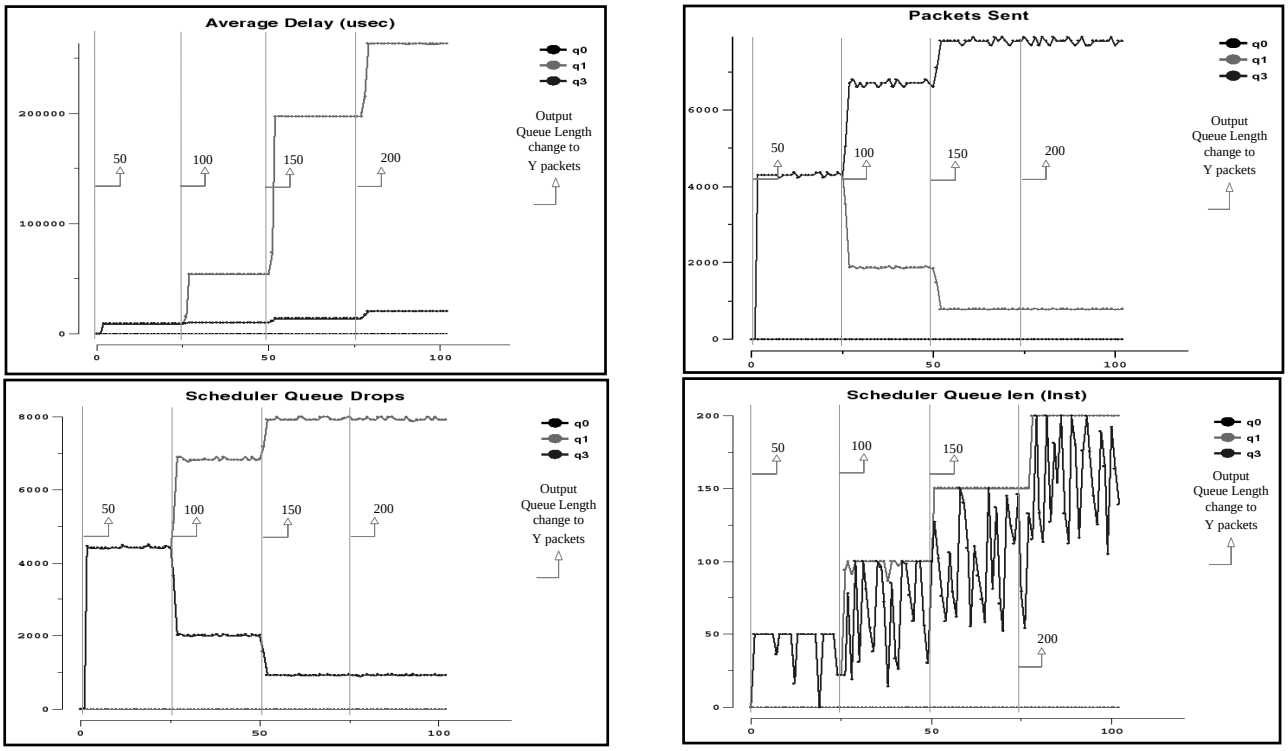

Figure 5 – Variation of the following values with IP output queue maximum length: ↖) Average IP transit delay (over 1 second intervals) ↗) Number of packets sent per second; ↙) Number of dropped packets per second; ↘) Instantaneous IP output queue length (sampled every second)