Evaluation of Integrated Services Support on Linux

Elisabete Reis[1, 2], Edmundo Monteiro[2]

[1]Polytechnic Institute of Coimbra
ISCAC, Quinta Agrícola, Bencanta
2040-316 Coimbra – Portugal
`http://www.iscac.pt`
`elreis@dei.uc.pt`

[2]University of Coimbra
Laboratory of Communications and Telematics
CISUC / DEI, Pólo II, Pinhal de Marrocos
3030-290 Coimbra – Portugal
`http://lct.dei.uc.pt`
`edmundo@dei.uc.pt`

Abstract. Linux operating system kernel offers a wide variety of traffic control functions, including the mechanisms required to support the Integrated Services architecture developed in the IETF. The main objective of this work is the evaluation of the Linux Traffic Control IntServ implementation. The evaluation is focused in the Guaranteed Service due to the more stringent needs of this service. The evaluation assessed performance behaviour under different traffic loads, scalability and stability. The results shown that the Linux IntServ implementation has some limitations in the support of the Guaranteed Service loss and delay requirements, especially for a high number of flows with small packet sizes.

1 Introduction

The increasing use of applications with stringent requirements on transit delay, bandwidth and losses, raised the need for the development of communication systems with Quality of Service (QoS) capabilities. The Integrated Services Model (IntServ) was developed by the Internet Engineering Task Force to answer to some of these needs in the Internet.

Currently the IntServ model supports two kinds of services: the Guaranteed Service (GS) [1] and the Controlled-Load Service (CL) [2]. The CL service emulates the Best-effort service over an unloaded network and is useful to support elastic applications. GS offers stringent loss and delay guarantees and was designed to support real-time traffic.

Although some negative impact in the performance of the routers in backbone networks, the IntServ model has interesting characteristics either for Internet Service Providers (ISPs) or for end users.

This is due to the possibility of the differentiated treatment of application flows provided by this model that enable the support elastic and real-time traffic. This capability allows ISPs to offer new services, with QoS guarantees, to the end users. Nevertheless this described advantages, IntServ is usually associated with performance and scalability problems due to the need of per flow treatment inside the network.

The main objective of this work is the experimental study and evaluation of the IntServ model, aiming at the assessment of its performance, stability and scalability. The study was focused in GS because it is the most exigent service regarding resources consumption. The Linux IntServ implementation was chosen to build the test platform because of its openness and wide dissemination.

Linux kernel supports a number of advanced networking features, including QoS. The QoS support provides a framework for the implementation of various IP QoS models like Integrated Services and Differentiated Services, in a module generically denominated Traffic Control (TC).

The Linux Traffic Control module consists of four building blocks: queuing disciplines, classes, filters and policing. Currently there are many queuing disciplines supported in Linux [3], including Class Based Queuing (CBQ), Clark-Shenker-Zhang (CSZ), Priority, Token Bucket Flow (TBF), Stochastic Fair Queuing (SFQ), Random Early Detection (RED) and First In First Out (FIFO).

In the Linux implementation tested, CBQ is the only discipline that is able to handle RSVP and Integrated Services, and thus it is the discipline used in this work.

The CBQ discipline assumes that a flow that has been accepted by the admission control module for GS will be assigned its own class with the highest priority [4]. Since all flows belong to classes that have the same priority, they will be served in weighted round robin scheduling (WRR).

The rest of the paper is organized as follows. Section 2 describes the experimental testbed; Section 3 performance experiments, Section 4 scalability evaluation and Section 5 stability evaluation. Conclusions and future work topics are presented in Section 6.

2 Experimental Testbed

In order to evaluate the Linux Traffic Control modules in the support of the Integrated Services model a Ethernet testbed was built with two end-hosts (Sender and Receiver) interconnected through three serially interconnected routers (Router A, B and C). All routers run the Linux operating system, Red-Hat 5.2, kernel 2.2.8, with RSVP installed. The Linux Traffic Control modules were installed and configured in the kernel of the routers to support RSVP operation.

The link between the Router A (the router next to the sender) and Router B was configured to operate at 10 Mbps to create a congestion bottleneck. All the other links were configured at 100 Mbps.

The Class-Based Queuing service discipline was activated and configured on the output interfaces of all routers. The link-sharing structure configured such the total capacity of the link was distributed as follows: 45% of bandwidth was allocated to best-effort traffic, 5% to RSVP control messages and 50% to IntServ GS flows. With this setup several combinations of data flows of GS and best-effort flows where generated. Best-effort traffic was generated with the *mgen* tool. For GS traffic flow generation the Chariot tool was used.

During the experiments, a number of parameters were measured and analysed to achieve insight on the service provided by routers. These parameters included: queuing delay, loss rate and bandwidth (parameters that reflect the characteristics of GS) and also queue lengths, number of queued packets and number of reclassified packets. To capture these parameters, a number of tools were used including *ttt* (Tele Traffic Taper) and an improved version of *tc* (Linux Traffic Control configuration tool). The Linux kernel was modified to enable the measurement of maximum and average packet queuing delays. The measures were taken in the router where the bottleneck was created (Router A), since the other routers of the testbed are connected at 100 Mbps and all traffic, including best-effort traffic, is forwarded without significant delays or losses.

After the preliminary validation tests, three different set of experiments were done. The first set aimed at the evaluation of the performance of the service provided to flows under different traffic loads. The second set was dedicated to scalability evaluation. The capability of routers to provide the required level of service for a variable and large number of flows was evaluated. Finally, the third group of testes had the objective of stability evaluation.

In the next sections the results of the experimental evaluation done are described. Some previous results of this study are included in references [5, 6].

3 Performance Evaluation

The main objective of this test set is the performance evaluation of the Linux Traffic Control provision of GS. This evaluation was done for different traffic loads, when all guaranteed flows were conforming with traffic specification and also when some guaranteed flows were non-conforming to this specification. The performance of network resources usage was also evaluated.

In all tests 4 GS flows were generated with a reserved bandwidth of 0,5 Mbps each. In the tests with non-conforming flows, the last three flows where the non-conforming. A reclassified traffic class was created to accommodate non-conformant reclassified GS traffic. The total load generated in each test was a combination of GS flow traffic and best-effort traffic to induce congestion.

A total load in the range from a small to a large percentage of the capacity of the output interface of Router A (bottleneck capacity) was generated (20% to 200%). Since the load of GS flows remained constant, the best-effort traffic was responsible for the increasing in the network load.
The analysis of the results shows that GS flows didn’t suffer losses, even under high loads (Fig 1). However, best-effort traffic has a considerably different behaviour since they don’t suffer losses until congestion occurs, around 90% of bottleneck capacity. After this value, the percentage of best-effort dropped packets raises significantly and increasingly, with the level of congestion in the router.

The analysis of Fig. 1 also shows that conforming and non-conforming GS have a different behaviour. Even though, non-conforming flows violate the established agreement, these flows don't suffer losses until congestion occurs. This is due to the fact that non-conformant traffic is not immediately dropped but is reclassified to a lower priority class (Reclassified Class). When congestion starts, the amount of dropped packets from non-conforming flows rises significantly until best-effort traffic begins to loose packets. It can be concluded that best-effort traffic is warmed with the presence of non-conforming GS flows.

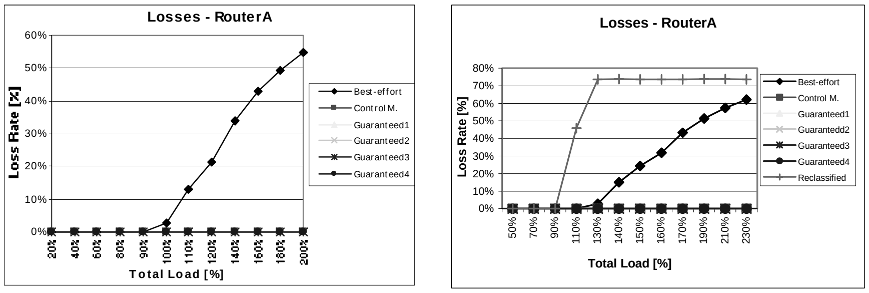

Fig. 1. Loss rate with conforming (left) and non-conforming (right) flows

The analysis of delay behaviour in Fig. 2 shows that when the router becomes congested, GS flows also suffer a noticeable increase in the maximum queuing delay. After the congestion point is reached and as the load in the router increases, the queuing delay experienced by conforming GS flows remains approximately constant, while the maximum delay experienced by non-conforming GS flows and by best-effort traffic increases significantly.

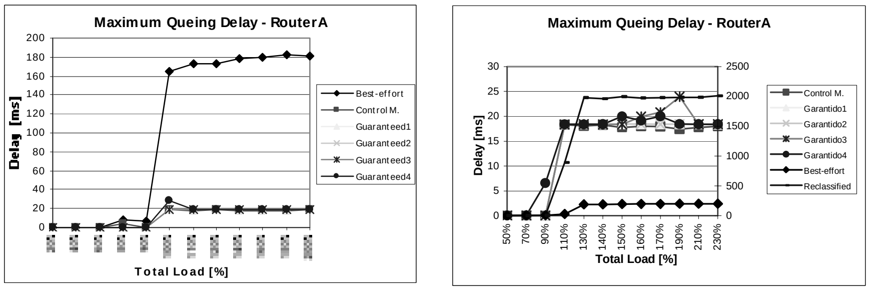

Fig. 2. Maximum queuing delay with conforming (left) and non-conforming (right) flows

From Fig. 2 it can also be concluded that best-effort traffic is warmed by the existence of non-conforming GS flows. The maximum queuing delay and loss rate, experienced by the best-effort traffic is higher when non-conforming flows are present.

The analysis of the tests in this section showed that, under all traffic loads generated, the bandwidth guarantees of GS flows were respected (0,5 Mbps). This commitment was verified even in the presence of non-conforming GS flows.

4 Scalability Evaluation

The deployment of Quality of Service in the Internet requires scalable solutions. In the present section the scalability characteristics of the Linux Traffic Control were evaluated for the support of the IntServ Guarantee Service.

For this purpose, three groups of tests were done with three different packet lengths: 1500 bytes (maximum length supported by Ethernet networks), 256 bytes (intermediate length) and 64 bytes (small length). In each test group a constant load of 10 Mbps was generated into the network while the number of active flows was changed. The maximum number of GS flows used in the experimentation was 15. This limit is imposed by the Linux Traffic Control implementation. Tests were carried out with 4, 6, 8, 10, 12 and 14 GS flows. All the flows (best-effort and GS) were generated with UDP traffic. The total reservation didn’t exceed 50% of total link bandwidth (5 Mbps). The evaluation was based on the behaviour of GS flows in the presence of best-effort traffic.

Different bandwidth reservations have made to each GS flow to make the analysis of results easier: the reservations were made from 0,11 Mbps to 0,52 Mbps, with 0,03 Mbps intervals (with the exception of the last flow that was limited by the admission control functions).

4.1 Tests with 1500 bytes packets

The results of the tests with 1500 byte packets (Fig. 3) show that GS flows didn’t experience losses, even for a large number flows. The Linux implementation of GS satisfies the commitment of providing a service without losses for these test specifications.

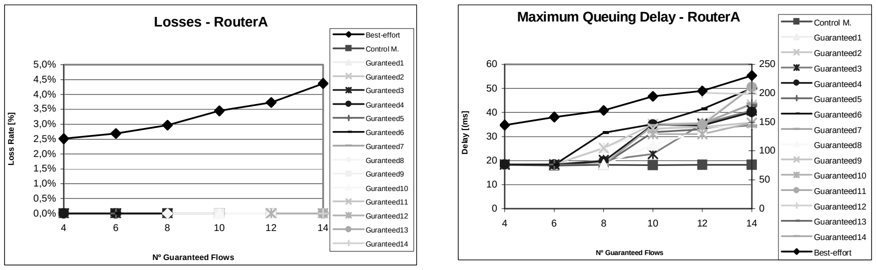

Fig. 3. Loss rate (left) and queuing delay (right) with 1500 byte packets

Concerning loss, the left side of Fig. 3 shows that, as the number of GS flows increased the percentage of dropped best-effort packets rose significantly. This was due to the fact that the lower priority flows are only served after the higher priority ones. In this case, the scheduler will spend more time processing high priority flows, degrading the performance of best-effort traffic.

The right side of Fig. 3 shows that the maximum queuing delay experienced by the guaranteed flows increases with the number of GS flows (left axis), causing service degradation. Best-effort traffic (right axis) performance was also damaged with the presence of a higher number of GS flows. This fact is even more significant because the degradation occurs for small loads of best-effort traffic. This behaviour is due to the fact that with more high priority flows the scheduler spends less time processing packets from the best-effort queue, and this traffic is delayed even for lower loads.

The bandwidth guarantees provided were also evaluated according to the test conditions described. It was verified that with a larger number of GS flows some of these didn't received the bandwidth that was previously reserved. Although the bandwidth commitment wasn't satisfied, the difference was not relevant and loses didn't occur.

4.2 Tests with 256 bytes packets

Since the applications with GS requirements normally generate small packets (e.g. audio and real time video), the impact of packet length in the behaviour of GS and best-effort flows was evaluated. The tests in the previous section were repeated using 256 bytes packets for GS flows and keeping 1500 bytes packets to best-effort flows. The results are shown in Fig. 4.

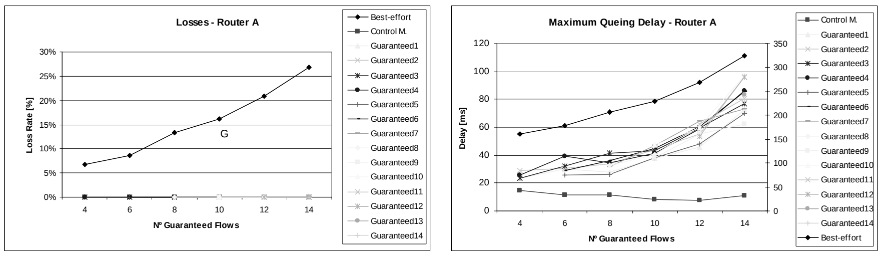

Fig. 4. Loss rate (left) and queuing delay (right) with 256 byte packets

Concerning losses, Fig 4 shows that, like with 1500 bytes packets, GS flows didn't suffer losses, even with the maximum number of flows (14 flows). However, and unlike in the previous tests, the number of packets in GS queues increased. Form the comparison of Figs. 3 and 4, it can be concluded that, for the same number of flows, the amount of best-effort dropped packets is significantly higher in the test with GS flows 256 bytes packets.

Regarding the maximum queuing delay, the right side of Fig. 4 shows an increase in delay experienced by all guaranteed flows and also by best-effort traffic. This fact is more evident with a larger number of GS flows.

Concerning throughput, once more it was verified, that with a high number of GS flows some of theses flows didn't received the bandwidth that was previously reserved. A higher number of GS packets, even while the load was kept constant, imposed a larger computational processing overhead on the real time scheduler.

4.3 Tests with 64 bytes packets

In the previous tests, GS QoS requirements were globally satisfied, although it was already noticeable that the degradation increased with the number of flows and with the reduction in the packet size. So, it is somewhat predictable that, for small packet sizes, GS flows requirements can no longer be satisfied by the Linux Traffic Control implementation. To verify this assumption, the tests were repeated once more with 64 bytes packets GS flows. The results are shown in Figs. 5 and 6.

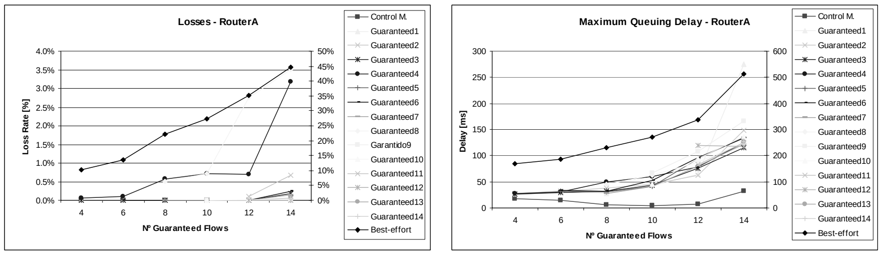

Fig. 5. Loss rate (left) and queuing delay (right) with 64 byte packets

The analysis of the results (Fig. 5) confirms the failure of the Linux Traffic Control implementation to support the requirements of a high number of GS flows with small packet sizes.

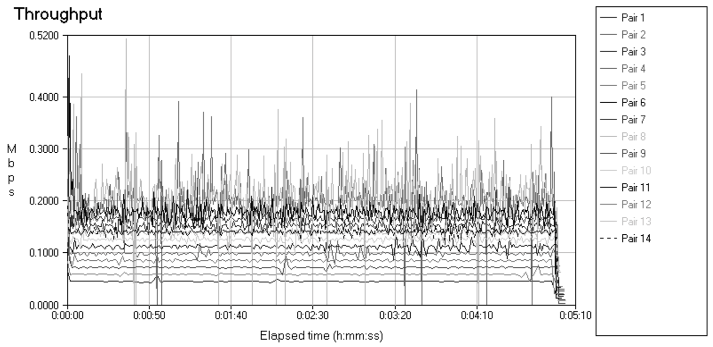

Fig. 6. Throughput of GS flows of 64 bytes packets

Like in the tests with larger packets (1500 and 256 bytes), the maximum queuing delay experienced by GS and best-effort flows increased significantly with the number of active GS flows, even when the total load was kept constant. However, and unlike the tests discussed before, some GS flows experienced losses in the tests with a high number of flows. It is also interesting to notice that GS flows with more losses were the flows with larger reservations.

The guarantees concerning bandwidth were also evaluated in this test (Fig. 6) and it was verified that flows didn't receive the bandwidth that was reserved.

To conclude the results of the tests in this section shown that the Linux Traffic Control IntServ implementation doesn’t support GS loss and bandwidth requirements and doesn’t scale with the number of flows for small packet sizes.

5 Stability Evaluation

The stability evaluation of the Linux IntServ implementation support of Guaranteed Service was done considering the parameters that can induce instability, namely: traffic mix, presence of Controlled Load traffic, bandwidth assignment to classes with reservations and network structure. Reservation were made from 0,11 Mbps to 0,52 Mbps with 0,03 Mbps intervals.

The results discussed in this section are mainly related with the evaluation of stability concerning the traffic mix. The aim was the evaluation of the influence of the traffic type in the stability of the service provided to GS flows.

The behaviour of GS was evaluated under different combinations of TCP-TCP and TCP-UDP flows in competition for the same router resources. The configuration with 14 GS flows of 64 bytes packets was chosen because previous tests showed that this was the most demanding configuration.

5.1 Tests with TCP traffic only

Since all the previous tests were done with UDP, the main objective of the present tests is the evaluation of GS behavior with TCP traffic. To evaluate stability behaviour regarding traffic types, 14 GS flows were generated with TCP traffic. Queuing delay, loss and throughput were measured for each flow. The results are shown in Fig. 7.

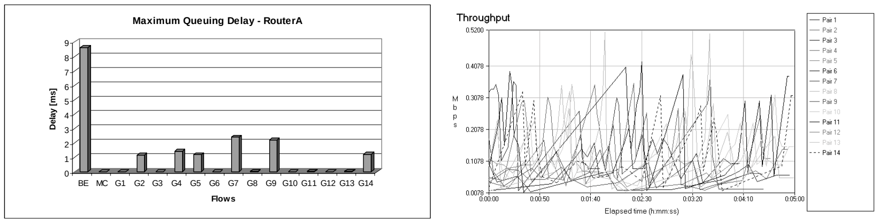

Fig. 7. Maximum queuing delays (left) and throughput (right) with TCP traffic

The analysis of queue sizes revealed that, due to the effect of TCP flow control, queues were empty most of the times and the scheduler didn’t use it’s ability to differentiate between different flows. These results show that TCP flow control mechanisms hinder IntServ Traffic Control modules to operate correctly. The results also show that the bandwidth provided to GS flows was protected from best-effort flows. Nevertheless this protection, the differences among GS flows were significant due to the TCP congestion control mechanisms.

5.2 Tests with TCP and UDP traffic

The behaviour of Guaranteed Service flows with UDP and TCP traffic was also evaluated with TCP traffic on odd flows and UDP traffic on even flows. The results are shown in Figs. 8 and 9.

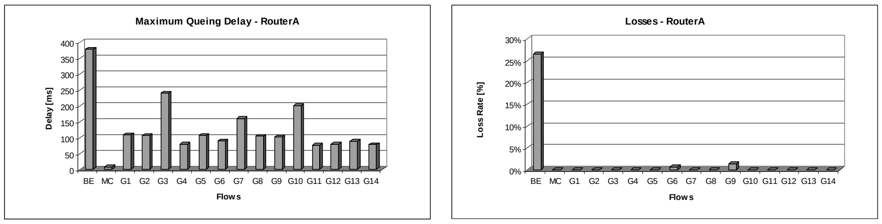

Fig. 8. Queuing delay (left) and loss (right) of TCP (odd) and UDP (even) flows

The analysis of Fig. 8 shows that queuing delays experienced by TCP and UDP flows were not significantly different. As stated before, and also in this case, the implementation was unable to provide service differentiation, independently of the traffic type. Regarding losses the difference between UDP and TCP flows was not noticeable.

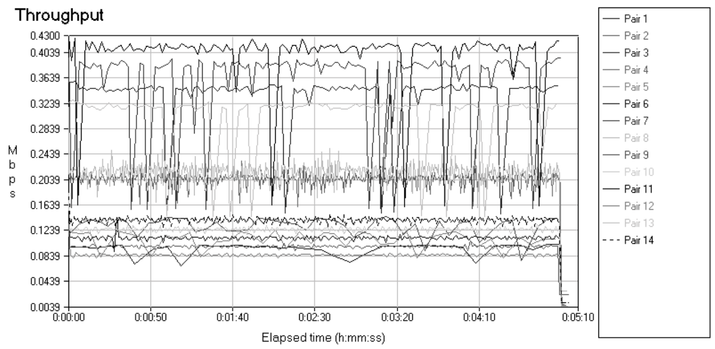

Fig. 9. Throughput of TCP (odd) and UDP (even) flows

Given the different characteristic of TCP and UDP traffic it seamed obvious that the UDP flows should get the major part of system resources. That is, due to the adaptive characteristics of TCP flows, the resources not used by TCP traffic should be used by UDP flows. However, the results shown that TCP flows got better service regarding bandwidth (Fig. 9). This behaviour was unexpected and can be attributed to the Linux Traffic Control IntServ scheduler configuration to protect TCP flows when in the presence of UDP traffic.

6 Conclusions and Future Work

In this paper an evaluation of the Linux Traffic Control implementation of the IntServ Guaranteed Service support was made. The evaluation addressed performance, scalability and stability issues.

Concerning performance, the implementation was able to control of GS QoS parameters independently of the state of load in the routers. The results also show that, unlike what was expected, TCP flows got better service than UDP flows, especially in regarding to throughput.

Regarding scalability, the experiments show some limitations of the Traffic Control mechanisms to guarantee the requirements QoS of GS flows in the presence of a larger number of flows of small packets.

The evaluation of stability regarding the traffic mix revealed some unexpected behaviour to protect TCP flows from of UDP traffic.

The experiments reported in this paper were conducted on a relatively small testbed and at low line speeds. From the identified limitations it can be induced that the implementation under study suffers from scalability problems when used in wider scenarios at high speed. Future work will address the evaluation of Guaranteed Service in WAN scenarios and the integration of IntServ with a DiffServ networks.

Acknowledgements

This work was partially supported by the Portuguese Ministry of Science and High Education (MCES), under POSI Programme project QoS II – Quality of Service in Computer Communication Systems.

References

1. S. Shenker, C. Partridge, R. Guérin, “Specification of Guaranteed Quality of Service”, RFC 2212, IETF, September 1997.
2. J. Wroclawski, “Specification of the Controlled-load Network Element Service”, RFC 2211, IETF, September 1997.
3. S. Radhakrishnan , “Linux – Advanced Networking Overview – Version1”, August 1999.
4. S. Floyd, M. Speer, “Experimental Results for Class-Based Queueing”, September 1998, not published.
5. E. Reis, F. Melo, M. Oliveira, G. Quadros, E. Monteiro, “Quality of Service on the Internet: Evaluation of the IntServ Architecture on the Linux Operating System”, Proceedings of the 3rd Conference on Telecommunications (ConfTele2001), Figueira da Foz, 22-23 de Abril, 2001.
6. E. Reis, E. Monteiro, “Estudo da Escalabilidade da Implementação do Modelo IntServ em Linux”, Proceedings of the 5th Conferência sobre Redes de Computadores - Tecnologias e Aplicações (CRC’2002), FCCN, Universidade do Algarve, Faro, Portugal, 26-27 Setembro, 2002.