A Buffering Strategy for End-to-end Quality of Service Using the Real Time Protocol *

Dario Maggiorini, Elena Pagani, Gian Paolo Rossi
Dip. di Scienze dell'Informazione - Università degli Studi di Milano
{dario,pagae,rossi}@dsi.unimi.it

Extended Abstract

The sudden and growing diffusion of Internet and of network-based applications is making IP an attractive platform to support multimedia traffic. Unfortunately, despite the continuous growth of the links capacity and of the switching power of the routers, the best-effort nature of IP is not sufficient alone to ensure the required quality to that traffic.

The problem of providing Quality-of-Service (QoS) impacts on all the network architecture layers. This problem may be solved by working either at hop-by-hop layer or at end-to-end layer. The latter approach is more promising to achieve QoS in a short term period, because it does not require any changes to the current Internet structure. At the end-to-end layer QoS can be supplied by providing both (i) call admission control functionalities in the set-up phase for new service requests, and (ii) flow management functionalities at the receivers to guarantee that data are delivered with the requested quality. With respect to point (ii), in this work we focus on the problem of characterizing buffering strategies for the delivery of CBR traffic. We describe the structure of the testbed that we have deployed to that purpose and we present the results that we have achieved by performing preliminary experiments with it. The reference architecture considered in the testbed implementation is that described in [1].

The experimental framework exploits the duplication of the TCP/IP stack at the user level. The experiments are performed by using that copy. The copy, in its turn, relies on the services of the system stack, as a usual user application. The behaviour of the protocols under test embedded into the stack are evaluated by means of a set of measurement tools aimed at recording a real-time trace of the protocol activities that can be successively analyzed. The measurement tools run both at the sender and the receiver sites. The observed performance indexes are network throughput, delay and jitter [2]. The measurement tools produce dump files describing the observed traffic. Generated dumps are evaluated from an external program that downsamples the data to a required timeslice, and generates data to be feeded to the `gnuplot` tool.

Our user stack is currently composed of a freeware implementation of RTP [3, 4] which is interfaced with the copy at user level of the UDP kernel implementation. RTP is an end-to-end transport protocol for real-time traffic. The use of RTP allows us to achieve QoS in terms of the delivered jitter. The information provided by RTP can be used by a CBR application to guarantee a constant rate delivery of the data packets at the destination. At the sender host, RTP marks the generated messages with a timestamp and a sequence number. At the receiver, the messages are stored in a *playback* buffer, according to the sequence number order. Sequence numbers are as well used to detect duplicates, which are discarded. At the receiver, the playback process runs, that implements the buffering strategy under test. The playback process performs the filtering and buffering of the packets received from the UDP protocol. Messages are extracted from the playback buffer at a constant rate and can be delivered to a player process. We measure the traffic profile both when the messages are received at the destination host and at the output of the playback process. We are using RTCP reports to guarantee end-to-end QoS while observing UDP level measures to evaluate performance improvement from the traffic coming from the net. The RTCP reports contain, beside of the default QoS estimates described in [3], some application-dependent data. In particular, we evaluate the

*This work was supported by the MURST under Contract no.9809321920 "Transport of multicast packets with QoS guarantees", and by the Microsoft Research grant for the project "A Framework for the Testing and Evaluation of QoS-aware Protocols".

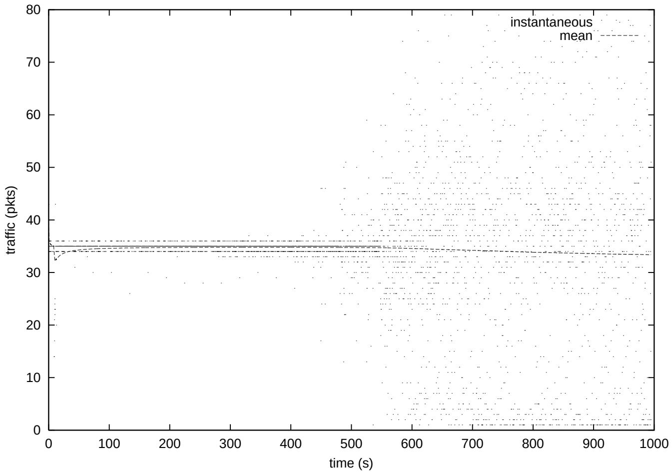

Figure 1: Incoming traffic at destination.

end-to-end delay perceived at the receiver site. Using the receiver reports, the sender measures the round-trip-time; by that measure it estimates the end-to-end delay, which is sent to the receiver in the next sender report.

We started to implement a trivial buffering strategy. Delivery starts when the number of messages in the playback buffer exceeds a high-water mark, and is performed according to the sequence numbers. The playback process makes use of a timer to estimate when the next message should be delivered. If the message with the appropriate sequence number is not in the buffer when the timer expires, the playback process delivers a `dummy` message to the player, increments the sequence number to be delivered next and restarts the timer. If a message m is received, whose sequence number is lower than that of the next message to deliver (i.e., m is a late message), m is discarded. We plan to devise an adapting buffering strategy, able to dynamically adequate the delivery process according to the current network congestion as reported by the end-to-end delay.

The framework has been implemented on UNIX OS's. All the modules have been implemented in C language. Our testbed network is currently constituted by a subsection of the departmental network.

We performed measures in the absence of background traffic. The traffic we used for the measures was mathematically generated and had a CBR profile. The QoS traffic is generated at a rate of 1.5 Mbps. We generated synthetic noise traffic so that the traversed routers are congested. The noise generator gradually increases the produced traffic, starting from 0 Mbps, until a rate of 10 Mbps is reached. Router congestion is achieved within approximately 1000 sec. Both data and noise packets have size 1KB.

The traffic profile received at the destination is shown in figure 1. After 500 sec. links and switches are congested by noise traffic, that affects the regular profile of the CBR traffic by introducing irregularities in the QoS flow. QoS packets are received in late, and sometimes bursts of packets are received. The mean intra-packet delay increases because of noise packets that are forwarded interleaved with the QoS packets. Because of this interleaving, the spacing between subsequent QoS packets becomes variable, and as the consequence the jitter increases.

In the presence of network congestion, the playback process must anyway provide a regular profile of the traffic delivered to the player. While the received QoS flow is regular, every time the playback process awaken to deliver a message, it finds the expected message in the buffer. When QoS flow starts to be negatively affected by noise, the length of the playback buffer drops under 50 packets and shows a saw-tooth behaviour with peaks in correspondence of the reception of a packet burst. The playback process has to generate dummy packets at an increasing rate to guarantee CBR delivery in spite of the gaps in the buffer. In figure 2 we show the throughput of the data packets delivered to the player. The mean throughput decreases as the noise increases, because QoS packets are received at a rate lower than that adopted by the playback process. By contrast, the throughput of the dummy packets increases.

In congested networks the implemented buffering strategy delivers a low-quality data flow. We are

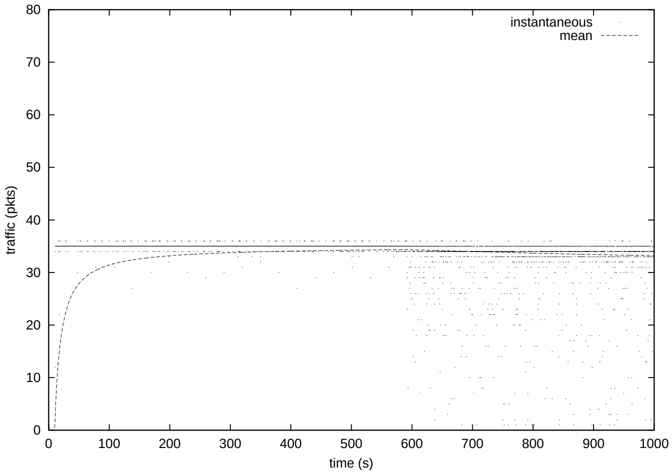

Figure 2: Throughput of the data packets delivered to the application.

currently performing experiments to compare different playback policies that guarantee better quality of the information flow delivered at the player. Those policies try to dynamically adapt to the changing network conditions. For instance, the packet delivery is suspended when either the number of packets in the playback buffer drops below a low-water mark or the end-to-end delay increases. Delivery is resumed when the buffer length exceeds a high-water dynamically recomputed according to the current estimation of the end-to-end delay.

References

[1] Maggiorini D., Pagani E., Rossi G. P., "A Testbed Environment for the Performance Evaluation of Modular Network Architectures," *To appear Proc. IEEE ECUMN 2000*, Oct. 2000.

[2] Bagnall P., Briscoe R., Poppitt A., "Taxonomy of Communication Requirements for Large-scale Multicast Applications," *Internet Draft, draft-ietf-lsma-requirements-02.txt*, Nov. 1998. Work in progress.

[3] Schulzrinne H., Casner S., Frederick R., Jacobson V., "RTP: A Transport Protocol for Real-Time Applications," *RFC 1889*, Jan. 1996. Work in progress.

[4] Schulzrinne H., "RTP: an overview." `http://www.cs.columbia.edu/~hgs/rtp/`.