A Methodology for Performance Analysis of Real-Time Continuous Media Applications

José Luiz A. da Fonseca[†] and Michael A. Stanton[‡]
[†]*Tecnologia da Informação – TI, Petróleo Brasileiro S/A - PETROBRAS, Av. República do Chile, 65 suíte 1602-A, 20035-900 Rio de Janeiro RJ, Brazil. E-mail: joseluiz@petrobras.com.br*
[‡]*Instituto de Computação – IC, Universidade Federal Fluminense – UFF, Rua Passo da Pátria, 156 bloco E sala 350, 24210-240 Niterói RJ, Brazil. E-mail: michael@ic.uff.br*

Present day continuous media applications have real-time requirements which must be satisfied by the transport service of the network infrastructure used, and the degree to which these requirements are met have a direct impact on performance, perhaps rendering the application inoperative. Therefore it is desirable to perform performance monitoring, through such performance parameters as delay, jitter, packet loss and throughput. Additionally, the awareness of the current values of these performance parameters may be of importance, in that a continuous media application may be able to tolerate the less than perfect satisfaction of its initial real-time requirements, and adapt itself to network conditions within certain limits. This paper describes a simple methodology for measuring these performance parameters using information generated by the RTCP protocol. The use of this methodology is illustrated by results from point-to-point desktop videoconferencing sessions.

Keywords: QoS; performance monitoring; real-time continuous media applications, RTCP.

1. Introduction

Internet utilisation has continued to grow exponentially with the appearance of commercial Web applications. In the present phase, there is a movement to equip websites with multimedia features, integrating data applications with audio and video. So far, most of these multimedia applications have been confined to downloading audio and video files to the local file system, for later reproduction after completion of the file transfer.

However, it has now become possible to use multimedia applications which require on-the-fly reproduction of the media stream before it has been completely received at the destination. Such *streaming media* applications have more stringent requirements of the transport service, especially of minimum throughput, in order to meet their objectives. More demanding still are interactive multimedia applications, such as desktop videoconferencing. In addition to throughput requirements, these applications also have temporal requirements, such as maximum end-to-end delay, and increased delay means worse performance and possibly denial of service.

These continuous media applications are not catered for by the current service model of Internet transport, and a number of new multi-service models, such as DiffServ [BLA 98] and IntServ [BRA 94], have been proposed to replace the best-effort service model of traditional IP networks.

This article examines ways of monitoring the performance of real-time continuous media applications, which are based on use of the Real-Time Protocol (RTP) [SCH 96]. Such applications are increasingly common, as RTP has been widely adopted as a media transport protocol in video conferencing and voice over IP. Such monitoring may be useful for evaluating alternative multi-service architectures, or for constructing management tools for dynamic adaptation of such applications to changes in network conditions.

In section 2 we identify the relevant characteristics of real-time media flows, which we refer to as "performance parameters". In section 3, the relevant characteristics of RTP and the related RTCP protocol are described, and in section 4 a methodology is developed for deriving values of the

O. Festor and A. Pras (Eds.) : 12th International Worshop on Distributed Systems: Operations and Management DSOM'2001 Nancy France, October 15-17, 2001.

performance parameters from observation of RTCP packets. In section 5, some sample results are demonstrated, and a summary of conclusions is presented in section 6.

2. Performance Parameters

Packet-switching technology, on which data networks in general, and IP networks in particular, are based, possesses properties not found in circuit-switching technology, as used in the telephone network. We have to identify those characteristics of IP networks that directly influence the performance of real-time continuous media applications, including desktop videoconferencing. In order to measure the performance of real-time continuous media applications, a number of parameters may be defined, which can be used as feedback by the applications, so that these may adapt themselves within certain limits to network conditions.

Current IP networks offer a best-effort packet delivery service, which offers no performance guarantees to the users, and packets may be lost in transit. Traditional applications, such as WWW, e-mail and file transfer, require reliable delivery, provided by use of the transport-level protocol TCP. Such applications are called *elastic*, since they have no delay or bandwidth restrictions which prevent them from functioning correctly.

Delay, packet loss and transmission capacity are very important transport characteristics for real-time continuous media applications, for, although they are usually able to tolerate a limited packet loss, their very feasibility depends on satisfying rigid limitations on maximum delay or minimum throughput.

In this article we will characterise the quality of service (QoS) provided for the application by means of the following **performance parameters**: (1) end-to-end delay, (2) jitter, (3) packet loss, and (4) throughput.

The **end-to-end delay** is the time spent by a packet in traversing the network from source to destination. This is the sum of the individual delays associated with switches and routers and their intervening links, which may generally be classified as delays due to processing, queuing, transmission and propagation. Excessive link demand results in congestion, associated with an increase in queue sizes and resulting queuing delay. Some applications may be affected by **jitter**, which is the variation in end-to-end delay. A continuous media application can usually eliminate the effects of jitter by use of a (possibly very large) reception buffer, which delays the "reproduction point" of the incoming media stream, at the cost of a further increase in end-to-end delay for the user of the application.

The **packet loss rate** is measured by the receiver as the fraction of transmitted packets which are successfully received in a given time interval. Packet loss in real-time continuous media applications are usually due to a mismatch between the source transmission rate (or demand) and the transmission capacity of the transport channel used, which leads to congestion at one or more network routers, increases in queue length and associated delays, and finally the dropping of packets, when queuing resources are overloaded.

One of the main attributes of a transport service is its capacity to carry bits, referred to as the **nominal bandwidth** of the channel. The **throughput** corresponds to the bitrate effectively used by the application, which is bounded by the nominal bandwidth.

3. The RTP and RTCP Protocols

It is not advisable to use TCP for transporting real-time continuous media in IP networks, because this reliable protocol recovers lost packets through retransmission and consequent delays, making it more difficult to satisfy the application's real-time requirements. This is because an application also considers a packet to be lost if it arrives at its destination later than its corresponding reproduction point. When the transport layer retransmits a lost packet, such that the retransmitted packet arrives after the reproduction point, the associated effort has been in vain, and may possibly have had the undesirable effect of increasing network demand and congestion.

Thus, it is essential to avoid the use of reliable transport protocols like TCP, that include automatic retransmission mechanisms. In the TCP/IP protocol suite, real-time continuous media applications generally use the protocol UDP.

UDP is a very simple protocol, and, for real-time continuous media applications, it is necessary to add further functionality, such as media encoding, packet sequence numbering and source timestamps. In order to provide a standard way of providing these additional functions, the IETF has adopted the Real-time Transport Protocol (RTP), defined in RFC 1889 [SCH 96]. RTP permits a number of standard media encoding schemes, such as G.711 [ITU 88] or G.723.1 [ITU 96] for audio and H.263 [RIJ 96] for video, as well as non-standard (proprietary) schemes.

Applications normally use RTP together with the UDP protocol. That is to say, continuous media data are encapsulated in RTP messages, which in their turn are encapsulated in UDP datagrams. Since RTP provides transport services for applications, it may be looked at as a sublayer of the transport layer, as shown in Figure 1.

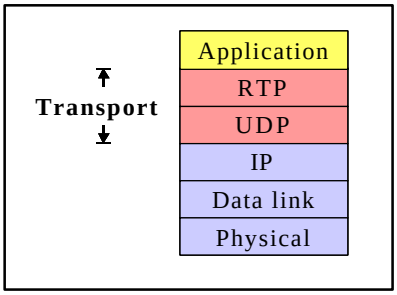

Figure 1 - RTP as a sublayer of the transport layer

It should be emphasised that RTP on its own does not provide any mechanism for guaranteeing timely packet delivery or QoS for the application. On the contrary, packets may be lost or delivered out of order.

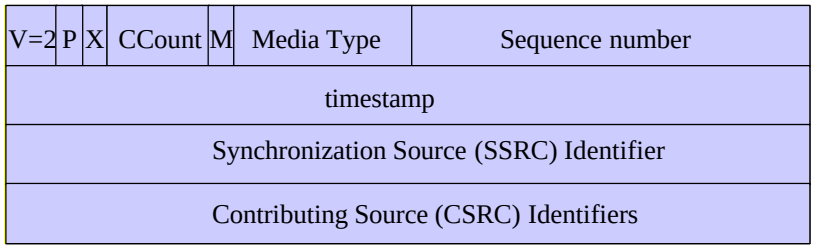

Figure 2 - RTP message header

As shown in Figure 2, there are four main fields in the RTP message header: media type, sequence number, timestamp, and media source identifiers.

In order to evaluate the performance of real-time continuous media applications using RTP, it is necessary to monitor the protocol RTCP (RTP Control Protocol), discussed next, which enables us to obtain measurements of the performance parameters which were introduced in the previous section.

RTCP, which is also defined in RFC 1889, is normally used together with RTP in real-time applications, and may be used when it is necessary to monitor the QoS provided by the network for continuous media transmission. As is shown in Figure 3, RTCP messages are transmitted by each participant, sender or receiver, in an RTP session to all the other participants of this session.

Like RTP, RTCP messages are transported using UDP over IP. RTP and RTCP messages are distinguished from each other by the use of different UDP port numbers. Thus a single media type uses a pair of adjacent UDP port numbers (2n, 2n+1), where the lower (even) port number is used for RTP and the higher (odd) number is used for RTCP.

RTCP messages are sent periodically, and contain only management information, such as the latest message sequence number sent or received, the number of missing messages, and the measured jitter of the received messages. RFC 1889 does not state how this information is to used, and this is up to

the application designer. For example, a media source may use information on packet loss to detect congestion, and to alter its transmission rate accordingly.

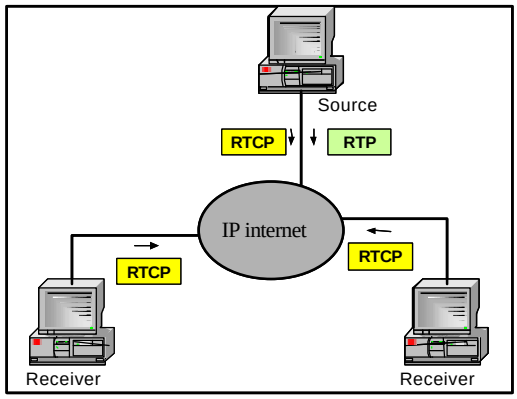

Figure 3 - Transmission of RTCP messages

The two main RTCP message types are SR (Sender Report) and RR (Receiver Report). SR messages are sent by a transmission source, whilst RR messages are sent by a receiver. All messages are received by all participants in a multicast session.

The SR messages contain, amongst other items, the following information, which may be used for calculating the performance parameters:

a. *NTP timestamp* (64 bits) – indicates the local time at the moment the present SR message was sent, using the NTP (Network Time Protocol) format, which uses the number of seconds since zero hours UTC on 1/1/1900 [MIL 92].
b. *Packet count* (32 bits): indicates the cumulative total of RTP packets sent by the source since the beginning of the session, until the moment the present SR packet is generated.
c. *Byte counter* (32 bits): indicates the cumulative total of media payload transmitted by RTP (the headers are not included) since the beginning of the session, until the moment the present SR packet is generated.

The RR messages include, amongst other items, the following information, which may be used for calculating the performance parameters:

a. *Number of packets lost* (24 bits): indicated the cumulative number of RTP packets lost since the beginning of the session, until the moment the present RR packet is generated.
b. *Largest sequence number* (32 bits): indicates the highest RTP packet sequence number received since the beginning of the session, until the moment the present RR packet is generated.
c. *Jitter* (32 bits): indicates the measured value of jitter, in accordance with a formula provided in RFC 1889, and measured in RTP timestamp units.
d. *Latest SR timestamp* (32 bits): indicates the middle 32 bits of the NTP timestamp of the most recently received SR message. This field is zero if no SR messages have so far been received.
e. *Emission delay of this RR message* (32 bits): indicates the elapsed time since the receipt of the most recent SR message. This field is zero if no SR messages have so far been received.

4. Experimental Methodology for Performance Evaluation

In order to measure experimentally the values of the performance parameters, a number of laboratory experiments were carried out, using a network which had been set up specially with the requisite hardware and software, as illustrated in Figure 4.

The routers Rot1 and Rot2 used in the experimental network were Cisco 2610 using IOS 12.1.5(T), and were interconnected through a serial link, using the Frame Relay link protocol. The routers were linked to different ports of a single 10 Mbps Ethernet switch, which was partitioned into two VLANs, and represented in Figure 4 as two separate switches.

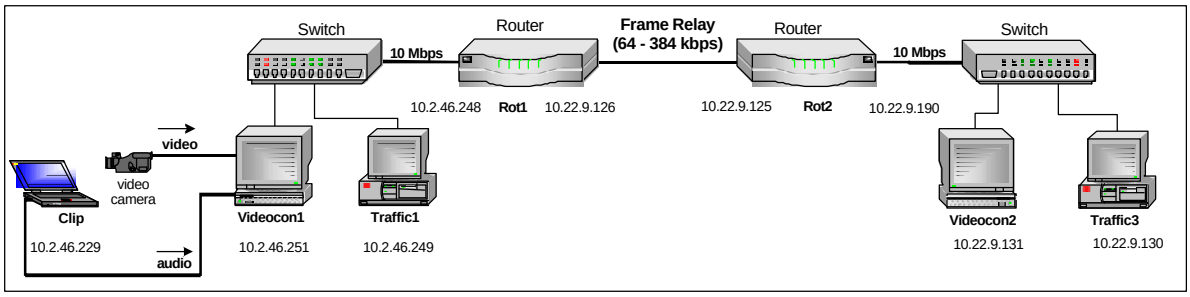

Figure 4 – Network used for measuring performance parameters

The workstations Videocon1 and Videocon2 were used for a desktop videoconferencing session using Microsoft's NetMeeting 3.01 [MIC 99], whilst the workstations Traffic1 and Traffic3 were used to generate cross-traffic, using the software NetSpec 3.0 [JON 94], with the objective of competing with the videoconferencing traffic for access to the link connecting the two routers.

The notebook Clip was used to generate audio and video content for the videoconference, transmitted from Videocon1 to Videocon2. Video content was captured using a WebCam II video camera pointed at the notebook screen, whilst audio content was captured directly by a cable connecting the sockets line-out of Clip and line-in of Videocon1.

4.1. Videoconference Session Monitoring

The workstations Videocon1 and Videocon2 were instrumented to capture RTCP packets during the videoconference sessions. RTCP packets were captured using WinDump 2.1 [DEO 00], which is a Windows version of the well-known Unix tool TCPDump [TCPD], and generates a binary file containing images of all the captured packets. RTCP packets are selected by filtering packets to (and from) the UDP ports 49607 and 49609, used respectively for video and audio media by both participants of the NetMeeting videoconference. The WinDump capture commands used are illustrated in Figure 5, where the files XXXtx.dmp[1] e XXXrx.dmp are the capture files generated by the workstations Videocon1 and Videocon2, respectively.

Workstation Videocon1: **windump –s**100 **-w**XXXtx.dmp "dst port 49607 or 49609"
Workstation Videocon2: **windump -s**100 **-w**XXXrx.dmp "dst port 49607 or 49609"

Figure 5 - WinDump commands for capturing RTCP packets

The capture files XXXtx.dmp and XXXrx.dmp were then formatted offline by TCPDump to render the capture files as text, as shown in Figure 6. These text files were then collated and processed by a custom program, which calculated time series of the instantaneous values of the performance parameters, using the methods described in section 4.2. Finally, graphic output of the different time series was generated using Microsoft Excel.

In Figure 6, the two RTCP packets displayed are of types SR and RR, respectively. The capture timestamp was generated by the WinDump capture tool. The source and destination nodes were identified from the IP addresses contained in the IP headers, which were converted into names by WinDump, whilst the port numbers were taken from the UDP headers. The measurement data described in section 3 was obtained from the RTCP packet. The uncommented field in the SR packet identified the origin of the transmission, and was not used by us. In order to reduce the effect of rounding errors in calculating the delay, the source code of TCPDump was modified to increase to 10 decimal places the precision of the fractional part of the timestamp data.

1 XXX corresponds to a coding used to identify about 40 different environmental configurations used in laboratory experiments.

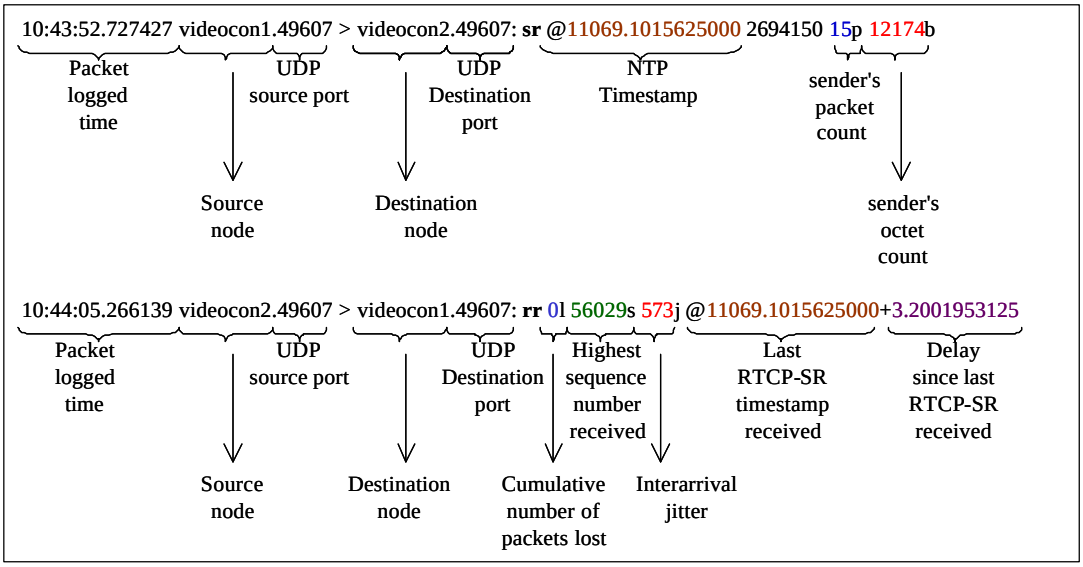

Figure 6 – Formatted version of capture file generated by TCPDump

4.2. Calculation of the Performance Parameters

Delay was calculated using the capture data generated at the source node, whilst the remaining parameters were calculated from capture data generated at the destination. In order to demonstrate the method used, we will consider the following example capture files generated for the audio media at both workstations.

```
08:13:54.212209 videocon2.49609 > videocon1.49609: rr 10l 61209s 227j @4984.5664062500+2.7451171875
08:14:00.778594 videocon2.49609 > videocon1.49609: rr 15l 61413s 353j @4989.4814453125+4.0458984375
08:14:05.976938 videocon2.49609 > videocon1.49609: rr 19l 61558s 436j @4996.8466796875+2.6220703125
08:14:09.201874 videocon2.49609 > videocon1.49609: rr 22l 61646s 273j @4996.8466796875+5.8476562500
```

Figure 7 - Receptor's capture file - audioXXXrx.dat

```
08:13:50.831126 videocon1.49609 > videocon2.49609: sr @4984.5664062500 44854910 162p 41472b
08:13:55.798128 videocon1.49609 > videocon2.49609: sr @4989.4814453125 44894326 315p 80640b
08:13:56.443764 videocon1.49607 > videocon2.49607: sr @4990.0693359375 6078060 79p 28776b
08:14:02.350009 videocon1.49607 > videocon2.49607: sr @4996.0898437500 6619950 134p 45546b
08:14:03.140933 videocon1.49609 > videocon2.49609: sr @4996.8466796875 44953710 547p 140032b
```

Figure 8 - Sender's capture file - refXXXtx.dat

```
08:13:43.880866 videocon1.49609 > videocon2.49609: sr @4984.5664062500 44854910 162p 41472b
08:13:46.627595 videocon2.49609 > videocon1.49609: rr 10l 61209s 227j @4984.5664062500+2.7451171875
08:13:49.145088 videocon1.49609 > videocon2.49609: sr @4989.4814453125 44894326 315p 80640b
08:13:53.194153 videocon2.49609 > videocon1.49609: rr 15l 61413s 353j @4989.4814453125+4.0458984375
```

Figure 9 - Collated combination of both capture files - audioXXXtx.dat

4.2.1. Delay

There is a practical difficulty in estimating the end-to-end delay of a transport channel, due to the lack of synchronisation between the clocks at the channel endpoints. One possible solution is to calculate the round trip delay at the sender's endpoint, through the use of a pair of RTCP packets (one

SR and one RR) which carry timing information. As the return traffic is limited to merely a few RTCP packets generated on average every 5 seconds, it may be supposed that the end-to-end transmission delay is almost equal to the round trip delay.

NetMeeting has a non-standard implementation of the SR message. According to RFC 1889, the RTCP timestamp field should contain the time in NTP format, based on time since 1st January 1900. However, this is not implemented by NetMeeting, which uses another time base. As this field appears to have been zeroed after reinitialisation of the operating system, it seems that it has been programmed to indicate the number of seconds since the most recent reinitialisation of the operating system. In any case, in order to calculate the round-trip time using the SR data, one would require access to the clock as used by RTP/RTCP on the source system. As this is unavailable for monitoring, instead one can use the timestamps generated in the capture file of WinDump, and use the SR originating timestamp merely as a label to connect the pairs of SR and RR messages.

If we observe the example shown in Figure 10, we note that the SR message carries the originating RTCP timestamp of 4984.5664062500, and a WinDump timestamp of 08:13:50.831126. The corresponding RR message has a RTCP timestamp of 4984.5664062500+2.7451171875, indicating that this message was generated 2.7451171875 seconds after receipt of the SR message. At the source, the RR message has a WinDump timestamp of 08:13:54.212209. Our estimate of the effective round-trip time is then the sum of t1 and t2, which is seen to be 08:13:54.212209 – (08:13:50.831126 + 2.7451171875) = 0.635966 seconds.

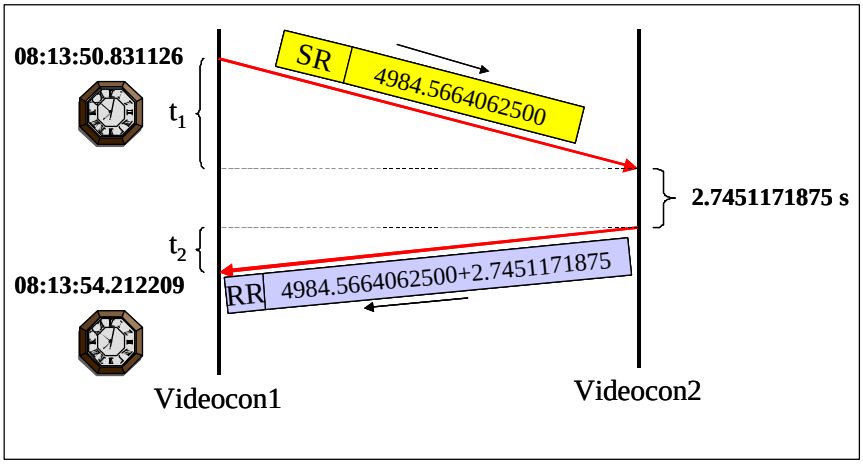

Figure 10 - Calculation of the Round-trip delay

4.2.2. Jitter

Jitter is measured by RTCP software and included in the RR messages sent by the receiver. As this value is measured in sampling units, in order to convert to time units, one must divide by the sampling rate of the media codec. Again using the example data, we have jitter values of 227 and 335 sampling units, which, when divided by ITU-T G.711 sampling rate of 8 kHz, correspond to 28.375 ms and 44.125 ms, respectively.

4.2.3. Packet Loss Rate

The packet loss rate is calculated from the data in two successive RR messages, which include both the highest packet sequence number received, and the cumulative total of lost packets. Using the example data, in the interval between the two RR messages, the total number of transmitted RTP packets was (61413 - 61209) = 204, and of these the number of lost packets was (15 - 10) = 5, giving an instantaneous loss rate of 5/204 = 2.45%.

4.2.4. Throughput

The throughput measured here is the effective channel bitrate, including protocol headers, received by the router Voz2 at the end of the serial link. The data used are an SR message, which permits us to calculate the average RTP media payload size, together with two successive RR messages, which yield the effective rate of packet delivery in the corresponding interval. Using the same example data, we calculate the average RTP payload as (41472 / 162) = 256 bytes per packet, to which must be added the headers corresponding to RTP (12 bytes), UDP (8 bytes), IP (20 bytes) and Frame Relay (6 bytes), making a grand total of 302 bytes per packet.

The two RR messages are separated by a time interval of 6.566558 seconds. In this interval, as we have already seen, the number of packets sent was (61413 – 61209) = 204, of which (15 –10) = 5 were lost. Thus the channel throughput can be calculated to be (199 x 302 x 8 / 6.566558) = 73.22 kbps.

5. Some Results of Experimental Monitoring

The results of experiments (011) and (021), which are shown expressed as time series in Figures 11 and 12, respectively, had as objective measure the behaviour of real-time continuous media traffic sharing a common link with other, best-effort cross-traffic.

This cross-traffic was generated using UDP, rather than TCP, so that it could be rate-controlled, and not subject to the TCP congestion control algorithms [JAC 88], which would probably interfere in our evaluation as we did not directly measure the cross-traffic. This cross-traffic was generated suing the software NetSpec 3.0 [JON 94], and was configured to generate a variable bitrate data stream with an average bitrate of 128 kbps, starting 4 minutes after the beginning of the videoconference session.

This videoconference used two media streams from Videocon1 to Videocon2, transmitting "low quality" video with resolution QCIF (176 x 144 pixels) and H.263 encoding, and G.711 audio (64 kbps). Here "low quality" refers to NetMeeting's user interface, and is a relative term. The quality is, of course, determined in this case by the frame rate.

In experiment (011) the serial link bandwidth between the two routers was configured to be 128 kbps, whereas in experiment (021) a nominal bandwidth of 256 kbps was used.

The results of experiment (011) demonstrate that the cross-traffic has considerable impact on the performance of the videoconference. With the start of the cross-traffic data stream, the total demand for the shared link exceeded its nominal bandwidth, resulting in extreme congestion and packet loss. Monitoring of the videoconference media streams showed that for video there was a packet loss of around 30% and a reduction in throughput of around 50%. In the case of audio, the packet loss was around 26%, and the round-trip time increased to 1600 ms, making it impractical to maintain interactive voice communication.

On the other hand, the results of experiment (021) show clearly that the provision of sufficient link bandwidth to accommodate the data streams of both contending applications produced acceptable performance of the videoconference. There was practically no packet loss for either media stream, and the media throughput remained constant in the presence of cross-traffic. There was, however, an increase in round-trip delay, but this was maintained below 250 ms, considered acceptable for interactive voice communication [CRO 99].

6. Conclusion

This article has presented a simple methodology for measuring the values of performance parameters, which may then be used for evaluation of different multi-service models, such as DiffServ [BLA 98] and IntServ [BRA 94], which have been proposed to replace the best-effort service model of traditional IP networks. In such service models, mechanisms are introduced to differentiate between different types of traffic, permitting the provision of better QoS to certain applications, such as real-time continuous media applications, by "protecting" their traffic from being affected by competing, less urgent application traffic. Such an application of this methodology has already been carried out, and has been reported elsewhere [FON 01A], [FON 01B].

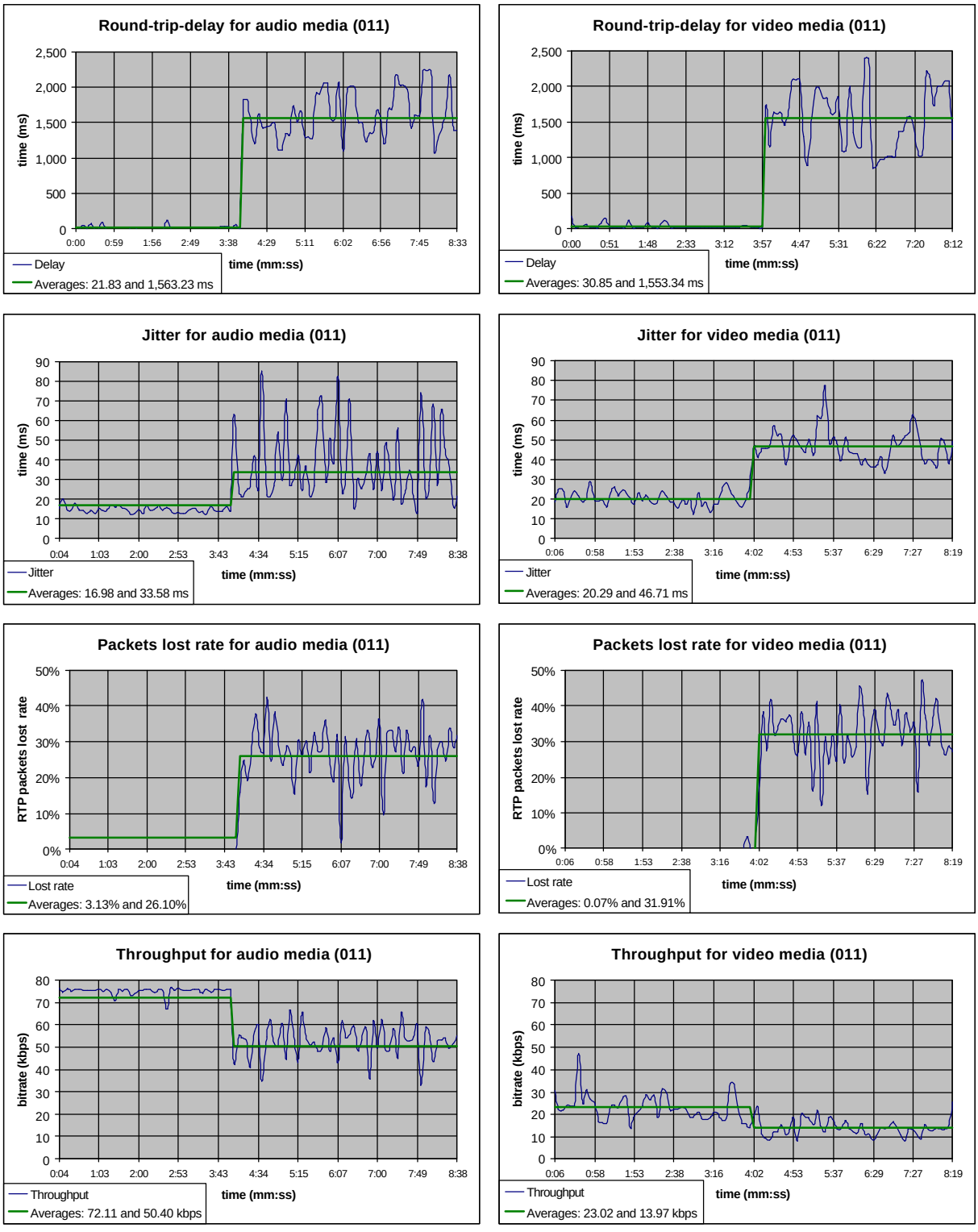

Figure 11 - Monitoring of the performance parameters with a cross-traffic of 128 kbps and shared link bandwidth of 128 kbps

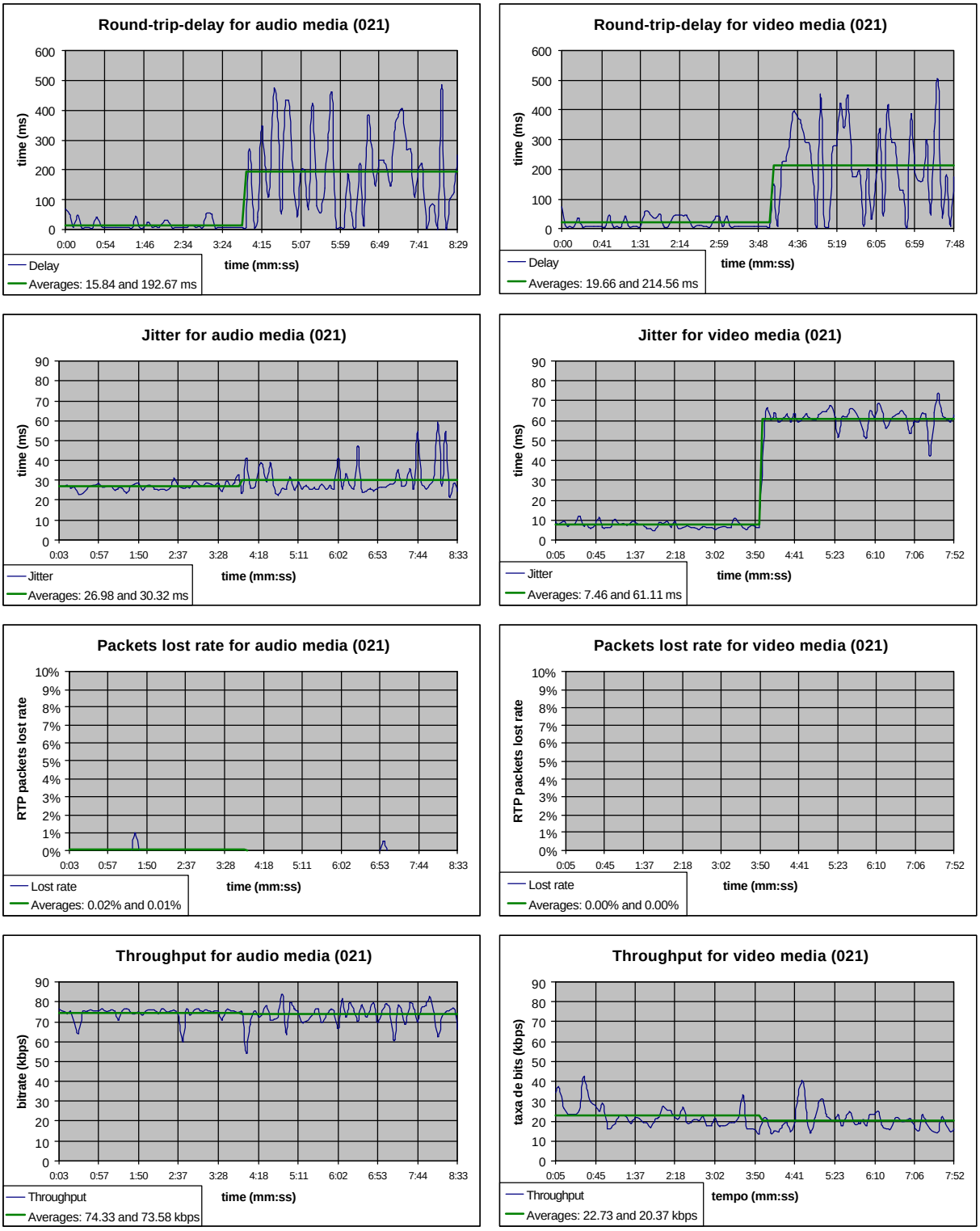

Figure 12 - Monitoring of the performance parameters with a cross-traffic of 128 kbps and shared link bandwidth of 256 kbps

An alternative application of this methodology could be to the development of tools to permits monitoring and adaptation of real-time continuous media applications to varying network conditions. Such an application is reported in [LUN 01].

The experimental work described here was carried out in the laboratories of Petróleo Brasileiro S/A, as part of the master's dissertation of the first author at the Universidade Federal Fluminense. The routers used were kindly provided by Cisco Systems Inc., which also provided technical assistance for configuring the equipment.

The second author acknowledges partial support by Project #0626/96 FINEP/RECOPE/SAGE.

7. References

[BLA 98] BLAKE, S. et al. **RFC 2475: An Architecture for Differentiated Services**. IETF, December 1998.

[BRA 94] BRADEN, R., Clark D. and Shenker S. **RFC 1633: Integrated Services in the Internet Architecture: an Overview**. IETF, June 1994.

[CRO 99] CROLL, A. and PACKMAN, E. **Managing Bandwidth: Deploying QoS in Enterprise Networks**. Prentice Hall, 1999.

[DEO 00] DEOGIOANNI, L. et al., **WinDump: TCPdump for Windows**, Politecnico di Torino, Italia, March 2000. *http://netgroup-serv.polito.it/windump/*

[FON 01A] FONSECA, J.L.A., **Network Architecture for Real-time Multimedia Applications** (in Portuguese), Master's dissertation, Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, 2001.

[FON 01B] FONSECA, J.L.A., Stanton, M.A., **An experimental study of desktop videoconferencing in IP internets with QoS.** ITCOM 2001, Denver, Colorado, USA, August 2001.

[ITU 88] International Telecommunication Union, **ITU-T Recommendation G.711: Pulse Code Modulation of Voice Frequencies,** Geneva, 1988.

[ITU 96] International Telecommunication Union, **ITU-T Recommendation G.723.1: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbps,** Geneva, 1996.

[ITU 99] International Telecommunication Union, **ITU-T Recommendation H.323: Packet-based Multimedia Communications Systems**, September 1999.

[JAC 88] JACOBSON, V., **Congestion Avoidance and Control**, Computer Communication Review, Vol. 18, nº 4, pp. 314-329, August 1988. *ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z*

[JON 94] JONKMAN, R., **NetSpec: Philosophy, Design and Implementation**, University of Kansas, Holland, 1994. *http://www.ittc.ukans.edu/netspec/docs/roel.ps*

[LUN 01] LUNARDI, S. and Dotti, F.L., **A QoS Adaptation Layer for Internet Multimedia Applications** (in Portuguese), Brazilian Symposium on Computer Networking, Florianópolis, Brazil, May 2001.

[MIC 99] Microsoft Corporation**, Microsoft NetMeeting 3 Resource Kit**, December 1999.

[MIL 92] MILLS D. **RFC 1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis.** IETF, March 1992.

[RIJ 96] RIJKSE, K., **H.263: Video Coding for Low-Bit-Rate Communication**, IEEE Communications Magazine, December 1996.

[SCH 96] SCHULZRINNE, H., et al., **RFC 1889: RTP - A Transport Protocol for Real-Time Applications**, IETF, 1996, 75pp.

[TCPD] TCPDUMP/LIBPCAP homepage, *http://www.tcpdump.org*