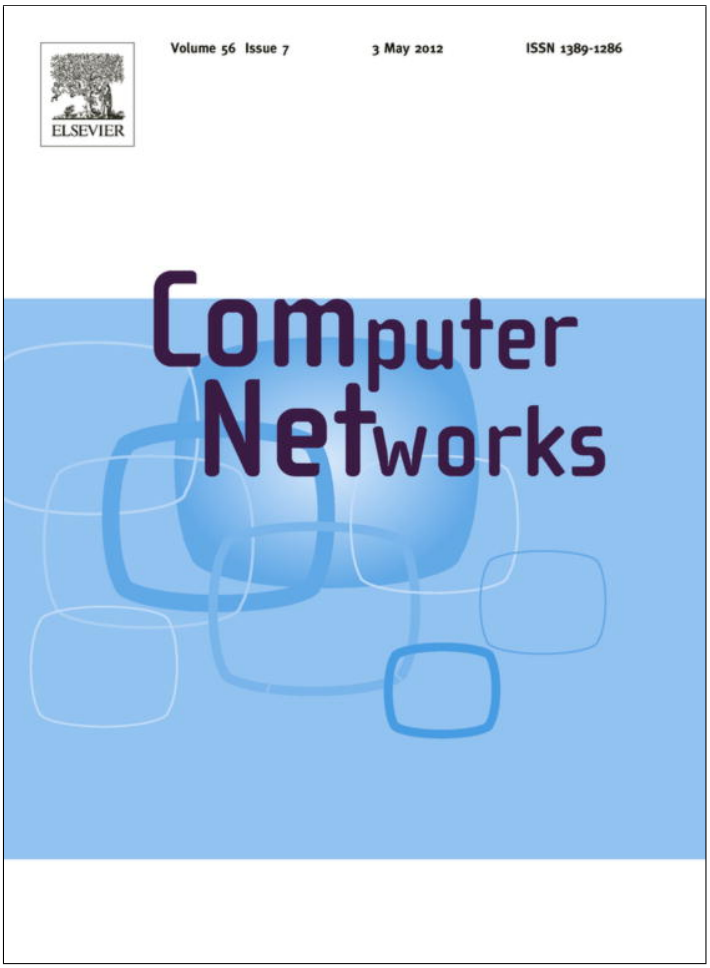

Computer Networks 56 (2012) 1893–1919

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Evaluating the influence of multiplexing schemes and buffer implementation on perceived VoIP conversation quality ☆

Jose Saldana *, Julián Fernández-Navajas, José Ruiz-Mas, Jenifer Murillo, Eduardo Viruete Navarro, José I. Aznar

Communication Technologies Group (GTC) – Aragon Institute of Engineering Research (I3A), Dpt. IEC. Ada Byron Building, CPS Univ. Zaragoza, 50018 Zaragoza, Spain

ARTICLE INFO

Article history:
Received 11 November 2010
Received in revised form 31 January 2012
Accepted 1 February 2012
Available online 10 February 2012

Keywords:
RTP
Multiplexing
QoS
VoIP
R-factor
Router buffer

ABSTRACT

This work presents a study of RTP multiplexing schemes, which are compared with the normal use of RTP, in terms of experienced quality. Bandwidth saving, latency and packet loss for different options are studied, and some tests of Voice over IP (VoIP) traffic are carried out in order to compare the quality obtained using different implementations of the router buffer. Voice quality is calculated using ITU *R*-factor, which is a widely accepted quality estimator. The tests show the bandwidth savings of multiplexing, and also the importance of packet size for certain buffers, as latency and packet loss may be affected. The customer's experience improvement is measured, showing that the use of multiplexing can be interesting in some scenarios, like an enterprise with different offices connected via the Internet. The system is also tested using different numbers of samples per packet, and the distribution of the flows into different tunnels is found to be an important factor in order to achieve an optimal perceived quality for each kind of buffer. Grouping all the flows into a single tunnel will not always be the best solution, as the increase of the number of flows does not improve bandwidth efficiency indefinitely. If the buffer penalizes big packets, it will be better to group the flows into a number of tunnels. The router processing capacity has to be taken into account too, as the limit of packets per second it can manage must not be exceeded. The obtained results show that multiplexing is a good way to improve customer's experience of VoIP in scenarios where many RTP flows share the same path.

1. Introduction

The use of the Internet for multimedia transmission is growing as bandwidth increases. Many of these new services, like Voice over IP (VoIP), videoconferencing, online gaming, etc. have very stringent real-time requirements, so network impairments may affect the interactivity of the service. For example, IP telephony customers expect the service to have the same interactivity as traditional telephony. As the use of IP telephony is growing, and best-effort networks without real-time delivery guarantees are often used, there is a concern regarding the quality perceived by the users of these services.

RTP is the most used protocol for real-time media transport. It has many profiles, and it is able to carry voice with different codecs, video and other real-time services. Due to real-time requirements, multimedia information has to be fragmented into small pieces of information, which are

☆ A preliminary version of this paper [1] appeared in Consumer Communications and Networking Conference (CCNC), 2011 IEEE, Las Vegas, 9–12 Jan 2011, pp. 378–382. Other parts [2] appeared in Consumer Communications and Networking Conference (CCNC), 2011 IEEE, Las Vegas, 9–12 Jan 2011, pp. 689–690.

* Corresponding author. Tel.: +34 976 76 2698.
E-mail addresses: jsaldana@unizar.es (J. Saldana), navajas@unizar.es (J. Fernández-Navajas), jruiz@unizar.es (J. Ruiz-Mas), jenifer.murillo@unizar.es (J. Murillo), eviruete@unizar.es (E.V. Navarro), jiaznar@unizar.es (J.I. Aznar).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2012.02.004

then transported into RTP packets using a small period. This fact implies that the overhead can be significant if the information carried by a packet is only a few tens of bytes, decreasing bandwidth efficiency. For example, a voice codec like G.729a can significantly compress information, generating a 10-byte sample every 10 ms. Thus, if two voice samples are included into an RTP packet, it will have 20 bytes of information, plus 40 bytes corresponding to IPv4/UDP/RTP headers. As a result, only one third of the bytes will carry voice information. Of course, if IPv6 is used, the efficiency becomes even worse.

There exist certain scenarios in which many RTP flows share the same path (Fig. 1): for example, a number of computers of the same office may use a PBX located at the data center of an enterprise; or different hosts of two offices of a SME (Small and Medium Enterprise) can establish simultaneous calls from one to the other. If this path includes the access network, which is normally a bottleneck, the deployment of solutions to reduce this overhead can be interesting. Two of them are header compressing, and grouping more samples into a single packet.

With regard to header compression, some schemes have been proposed, as we will see. They use a "context" shared by the sender and the receiver, which includes the protocol fields that are the same on every packet. As different flows can share the same origin and destination, each compressed packet has to include a Context Identifier (CID). The protocol also uses delta compression for the fields that increase from one packet to the next. Logically, this compression has to be applied in a hop-by-hop way.

Overhead can also be avoided by placing multiple samples into one packet [3], so as to increase the number of samples that share the same header. This can be achieved by bundling more voice samples of the same flow into a single packet (Fig. 2a), but this has a counterpart: each added sample will increase the packetization delay in the sender. There also exists the possibility of multiplexing samples of different conversations into the same RTP packet (Fig. 2b). This solution sends the same number of samples with the same frequency, so packetization delay is not increased. But it may add other delays which have to be studied.

RTP multiplexing combines these two techniques: header compressing, and bundling multiple samples into the same packet, but it has some disadvantages, i.e. new delays and processing charge. Multiplexing reduces the

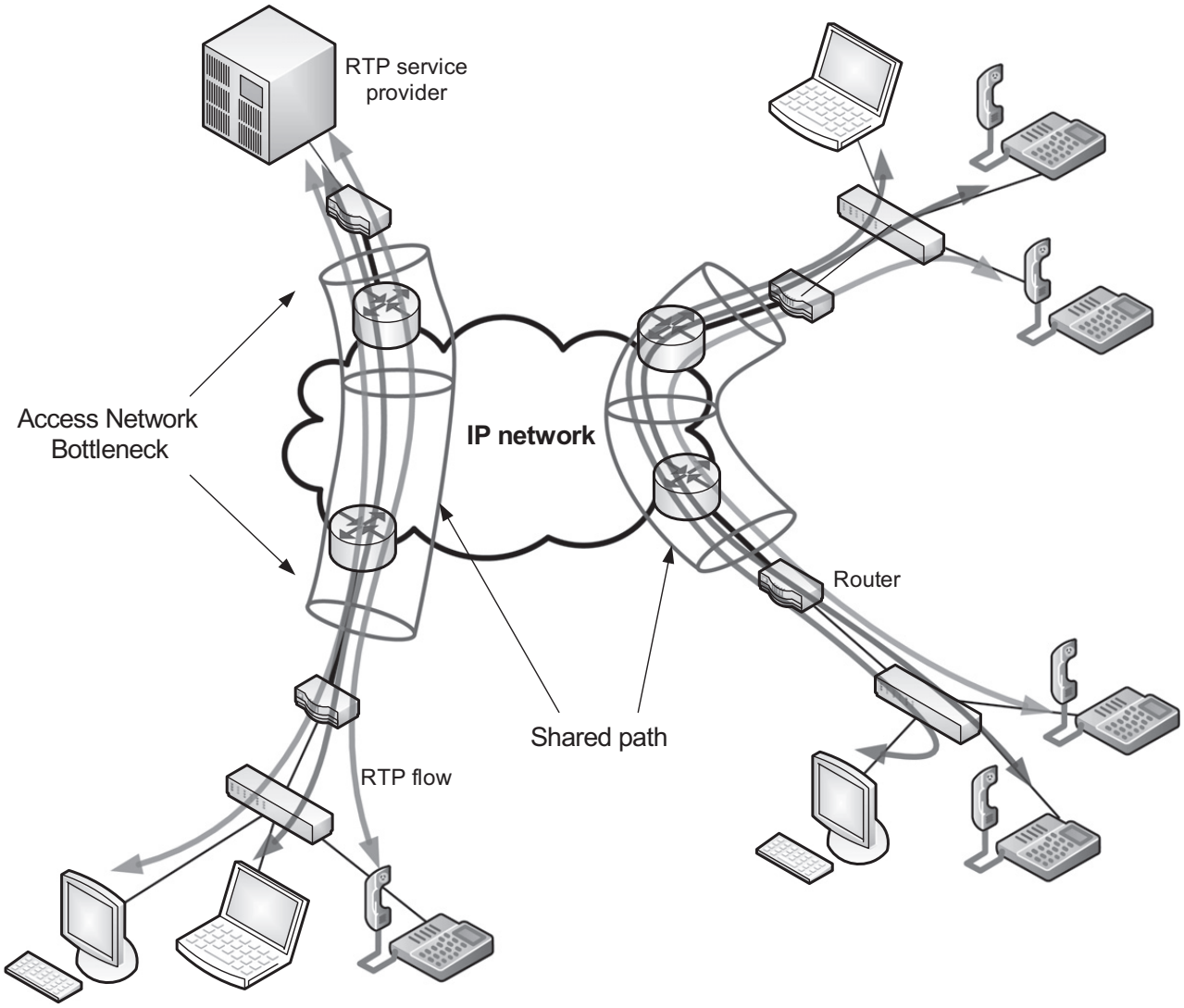

Fig. 1. RTP flows sharing the same path.

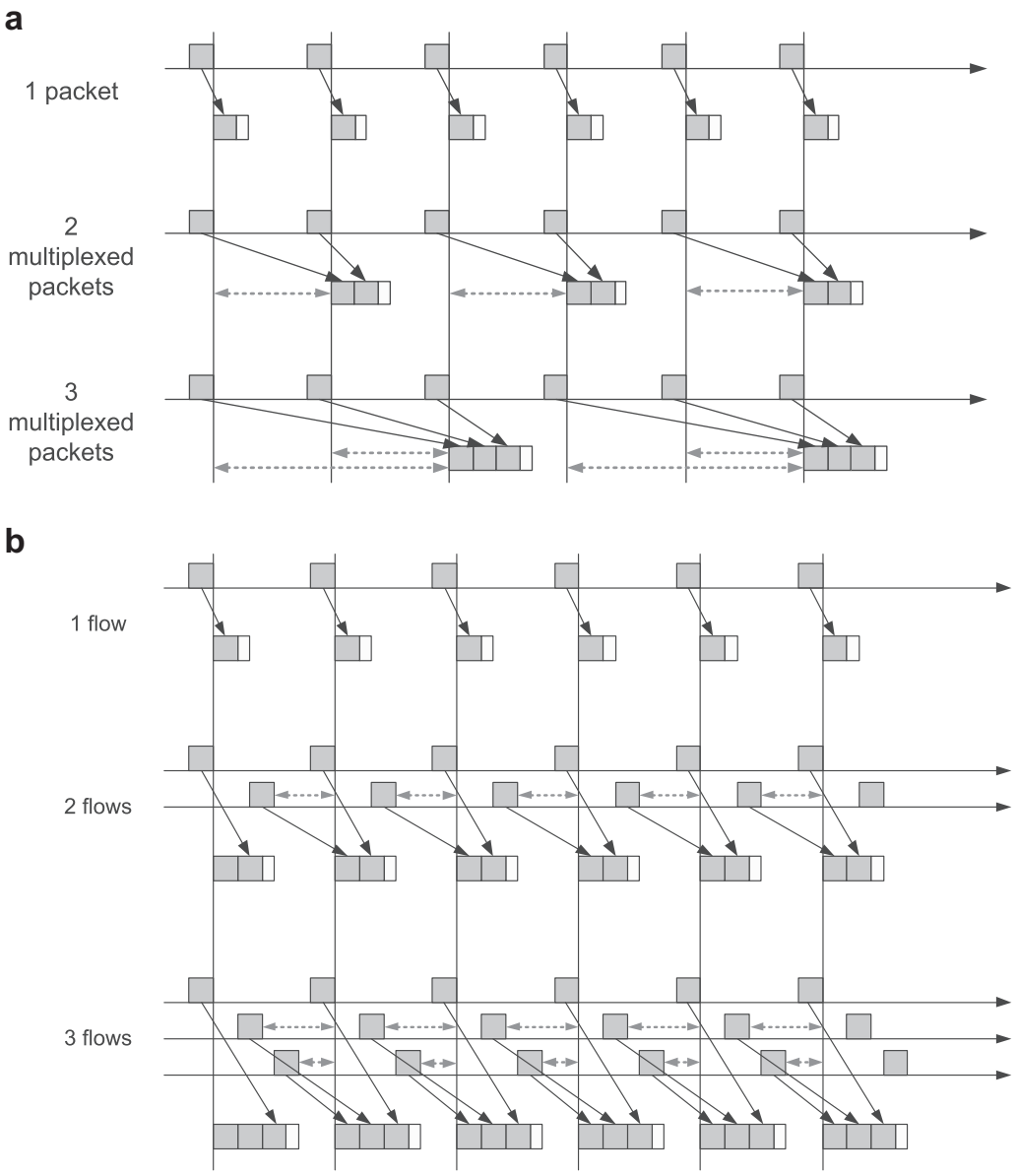

Fig. 2. (a) Bundling packets from the same flow. (b) Multiplexing packets from different flows. Discontinuous lines represent added delays.

number of packets per second (also denoted as pps), but it increases the average packet size. This effect has to be taken into account, as the buffer of the router has a big influence on the delays and losses of RTP packets and certain implementations may penalize big packets.

The motivation of this work is to study the effect of multiplexing on the quality perceived by the users of real-time services. The buffer has a central role, as its behavior has a decisive effect on the parameters that mainly define the quality, i.e. delay and packet loss, and multiplexing can codify these parameters.

In this paper we have selected VoIP to test RTP multiplexing. The first reason is that it is a very common service for which multiplexing can be a good improvement, as there are scenarios where many voice flows share the same path. Other reasons for using VoIP are the very stringent time requirements it has, which makes it interesting the study of delays, and the small size of RTP voice packets, which may imply a big gain when multiplexing. Finally, the existence of a widely accepted quality estimator, like ITU *R*-factor [4] which, based on latency and packet loss parameters, makes it easier to evaluate the improvements that can be achieved.

This paper is organized as follows: Section 2 discusses the related works about RTP multiplexing proposals and uses. The problem of sizing the router buffer is also issued. An analytical study of RTP multiplexing is presented in Section 3. The test methodology and the most significant parameters are presented in Section 4. The next section covers the measurements and tests. Section 6 discusses the obtained results, and finally the conclusions of the present work are presented.

2. Related works

In this section we will summarize the state of the art of the main issues considered in this work. Header

compressing standards and RTP multiplexing will be summarized, and also some uses of multiplexing. Finally, we will talk about the problem of sizing router buffers and its relationship with this work.

2.1. RTP multiplexing proposals

The problem of IP overhead was tackled many years ago. VJHC [5] was the first deployed standard in order to reduce TCP/IP overhead. Some years later, IPHC was defined [6]. CRTP [7] was based on IPHC, and it was able to compress IP/UDP/RTP headers. ECRTP [8] introduced some extensions in CRTP in order to enhance its behavior in scenarios with high packet loss, packet reordering and long delays. ROHC [9] is another compressing scheme designed to perform well in wireless and high-RTT links, as it reduces the impact of context desynchronization. It presents a higher complexity than CRTP. In [10] a survey of these standards can be found.

In 1998, the Audio/Video Transport Working Group of the IETF met to discuss different solutions for RTP multiplexing. There were different points of view and many proposals. Finally, IETF approved in 2005 TCRTP as RFC 4170 [11], with the category of "Best Current Practice". This standard does not define a new protocol, but it combines some of them. Its protocol stack [3] can be seen in Fig. 3. First, ECRTP compresses IP, UDP and RTP headers into a new header. Next, PPP multiplexing is used and finally packets are sent using an L2TP tunneling scheme. The use of a tunnel makes it possible to use ECRTP end-to-end, thus avoiding the need of decompressing and compressing at every router of the path.

Another multiplexing option was presented by Sze et al. [12]. It includes a number of RTP packets into a single UDP packet. RTP headers are compressed, so context-mapping tables are required at the multiplexer and demultiplexer in order to rebuild the original RTP packets. This solution was compared with TCRTP in [1].

There exist other multiplexing proposals in the literature. For example, in [13] a similar solution was proposed but it did not include RTP header compressing. Ref. [14] presented a system that adapts the throughput in response to congestion; GeRM [3] proposed the idea of including multiple RTP payloads, each one with a compressed header, into a single RTP packet; Ref. [15] assembles audio samples from different users into an RTP payload, using a 2 byte MINI-Header in order to identify each flow.

In this work TCRTP has been used, as it is an approved standard that binds a header compressing scheme with RTP multiplexing and tunneling. It is compared with native RTP.

2.2. RTP multiplexing uses

Real-time applications are being widely used, and bandwidth consumption is a concern for researchers. Multiplexing can simultaneously reduce bandwidth and the number of packets per second, while adding new small delays. Based on measurements of commercial routers, Ref. [16] discovered that, in certain conditions, the maximum call load is bounded by the router capacity rather than the link capacity, i.e. the number of packets per second a router can manage is limited. So they recommended the consideration of both packet and bit throughput. This conclusion was also reported in [17].

We have found some works in the literature interested on RTP multiplexing. Ref. [18] presents a study of an adaptive multiplexing system based on E-model. They mainly use multiplexing in order to decrease the overhead caused by IPSec.

Ref. [12] presented a system that multiplexes a number of RTP packets into a UDP one. They also study the delays that appear when a multiplexing scheme is used. They conclude that multiplexing can increase bandwidth efficiency by as much as 300%.

Multiplexing has also been proposed in other levels. In [19] two different systems were compared: sample application-layer aggregation, in which many RTP samples are included into an RTP packet, and the use of a performance enhancing proxy at IP level, which concatenates complete VoIP packets from multiple flows.

The present work not only considers the bandwidth reduction achieved by multiplexing, but also evaluates the effect of multiplexing schemes in terms of conversation quality, taking into account router's buffer implementation.

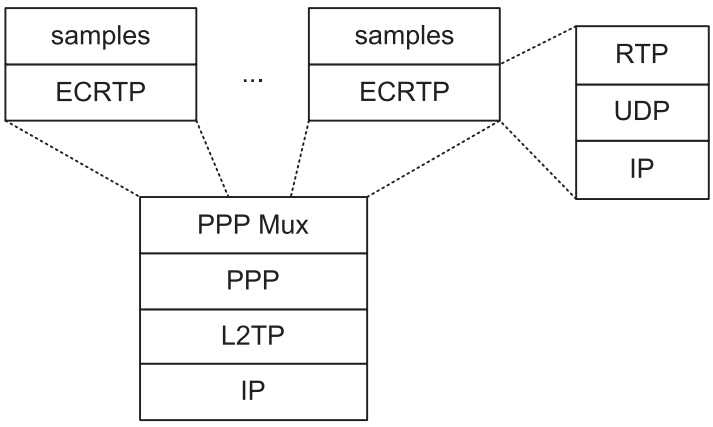

Fig. 3. TCRTP protocol stack.

2.3. The buffer size problem

The main objective of this article is to study the effect of multiplexing on perceived quality. Since many users connect to the Internet via an access network, the buffer of the router has a central role, as its behavior has a decisive effect on the parameters that mainly define the quality, i.e. delay and packet loss. On the other hand, the scenarios we are considering, in which a number of flows share a path, are mainly enterprises including a number of offices integrated into a central managed system. As a consequence, the routers considered have to fit this scenario. This is the reason why we will not consider backbone routers, but access ones, which means that they may not implement the advanced functionalities that can be present in high end routers: Random Early Detection (RED), a number of queues with different priorities, etc. In contrast, the studied buffers will only include FIFO queues.

In the last years, many studies have been published regarding the problem of sizing the router buffer. Although this problem has mainly been considered in the context of backbone routers, it also has some implications for access routers. A very complete survey of this problem can be found in [20]. For many years, the "rule of the thumb" for buffer sizing was the use of "Bandwidth-Delay Product" (BDP), i.e. the buffer size should be the product of the RTT (Round Trip Time) and the capacity of the link [21]. But in the last years, Appenzeller et al. [22] proposed the so-called "Stanford Model", which reduces the buffer size by dividing it by the square root of the number of TCP flows. After that, there has appeared a lot of literature studying optimal buffer sizes. Most of them are centered on the study of the behavior of a number of TCP flows in core routers. Ref. [23] presents a comparative study of different buffer sizes. One of them is the use of buffers of some tens of kilobytes, which are normally called tiny buffers. Other approach is the use of a time-limited buffer, which controls the maximum queuing delay at the target link. In fact, it is very similar to the limitation of buffer size, as the relationship between time and size limitation is given by the bandwidth. Other commercial buffers define their capacity in terms of a maximum number of packets, instead of bytes. In this paper we will study these approaches and the effects on perceived quality when they manage real-time traffic, taking into account the very strict time constraints it has, as delay is one of the parameters that determine *R*-factor [24].

The use of a time-limited buffer, in which packets are discarded if they exceed a fixed time at the buffer, is very convenient for this scenario, because it sets an upper bound for the delay, a very important parameter in real-time services. When using it, packet size may have an impact on the percentage of discarded packets. Big packets are expected to be discarded in a higher percentage than small ones, so multiplexed packets will have a higher loss probability than native RTP ones. On the other hand, multiplexing saves bandwidth and, as a consequence, for the same background traffic, multiplexed RTP may have better results than native RTP.

In the present study we will use this buffer implementation (denoted as *time-limited*), limiting its delay to 80 ms, comparing it with a *high capacity* buffer. We will consider that a buffer has high capacity when the delay it introduces is above the delays that can be tolerated by a real-time service like VoIP. In this study, we will use a delay of 800 ms in order to emulate this behavior. As we will see, this buffer implementation is not adequate for real-time services.

We will study two more buffers: first, the use of a *dedicated bandwidth* for VoIP traffic; in this case, the conversation quality will not be influenced by background traffic, as VoIP packets have an independent queue. And second, a router which can store a *fixed number* of packets instead of a fixed amount of bytes. When this buffer implementation is used, packet loss probability will no longer depend on packet size, as we can expect to happen when using other buffers.

These buffers are considered representative of the ones that can be found in low or mid end access routers. Our main objective is to know the conditions in which multiplexing represents a QoS improvement, taking into account the behavior of the access router buffer.

3. Analysis of multiplexing

In this section we present an analytical study of the influence of multiplexing on three parameters: packet size, efficiency in terms of IP/UDP/RTP overhead and packets per second.

3.1. Packet size

According to Fig. 3, we can define these parameters:

CH: Common header. It is the size of the header that is shared by the whole multiplexed packet. In TCRTP it corresponds to IP/L2TP/PPP headers.
RH: Reduced header. It refers to the size of the reduced header preceding the samples of each RTP flow. As the protocol has a number of possible reduced header sizes, we will calculate $E[RH]$ as the expected value of header size, taking into account the probability of having each one.
NH: Normal header. In a native RTP packet, this parameter includes IP, UDP and RTP headers.
S: Samples. It corresponds to the total size of all the voice samples included into an RTP packet.

So, the packet size for a native RTP flow is:

$$PS_{native} = NH + S \tag{1}$$

Let k be the number of multiplexed flows. The expected value of the size of a multiplexed packet is:

$$E[PS_{kflows}] = CH + k(E[RH] + S) \tag{2}$$

TCRTP considers three possible header sizes: COMPRESSED_RTP, which is the most compressed one; COMPRESSED_UDP, which does not compress RTP header, and FULL_HEADER, which does not compress at all. As we will see [25], more than 99% of the headers are compressed, so for simplification we will only consider here two values for

RH. Let RH_1 be the size of COMPRESSED_RTP header, and RH_2, the size of COMPRESSED_UDP header.

If we define p as the probability of having a COMPRESSED_RTP header, then the probability of having i COMPRESSED_RTP headers into a multiplexed packet of k flows will have a binomial distribution, so it can be expressed as:

$$Pr(i) = \binom{k}{i} p^i (1-p)^{k-i} \tag{3}$$

The average size of a reduced header can then be calculated as the expected value of a binomial distribution, so:

$$E[RH] = \frac{1}{k} \sum_{i=0}^{k} Pr(i)[RH_1 i + RH_2(k-i)] \tag{4}$$

If we operate with (3) and (4), we can obtain an expression for $E[RH]$ (5) which shows that it is independent from k. We could have expected this, as compressing techniques are independent of multiplexing.

$$E[RH] = pRH_1 + (1-p)RH_2 \tag{5}$$

So if there are more than two possible compressed header sizes, a multinomial distribution can be used instead of a binomial. If we define p_j as the probability of having header size RH_j, and M as the number of possible compressed header sizes, the expected value becomes:

$$E[RH] = \sum_{j=0}^{M} p_j RH_j \tag{6}$$

3.2. Bandwidth saving

In order to calculate the bandwidth saving achieved by means of multiplexing, we will next obtain the bandwidth relationship, which we will denote as *BWR*, between TCRTP and RTP. It should be noticed that, since the period is the same, the bandwidth relationship can be calculated as the relationship between the size of a TCRTP packet multiplexing k flows (2) and k non-multiplexed RTP packets, which is the product of k and (1). So *BWR* will be:

$$BWR = \frac{E[PS_{k\,flows}]}{k \cdot PS_{native}} = \frac{CH + k(E[RH] + S)}{k(NH + S)} \tag{7}$$

If we separate (7) in two parts, we obtain:

$$BWR_{Mux} = \frac{CH}{k(NH+S)} \tag{8}$$

$$BWR_{RH} = \frac{E[RH] + S}{NH + S} \tag{9}$$

We can see that BWR_{Mux} gets smaller as k increases, so it mainly depends on multiplexing. As BWR_{RH} does not depend on k, it will only be influenced by the capacity of the compressing algorithm to reduce the header, expressed in $E[RH]$, and the size of the samples S, which mainly depends on the codec, and the number of samples per packet.

In order to have a better idea of the numerical results that can be achieved, we have built some graphs of *BWR* as a function of k, S and the probability of having a reduced header p. The values for TCRTP using IPv4 are:

CH: 25 bytes.
RH_1: 4 bytes.
RH_2: 12 bytes.
NH: 40 bytes.
S: We will use 10, 20 or 30 bytes, which correspond to codec G.729a with one, two or three samples per packet respectively.

We have depicted *BWR* in Fig. 4, for $S = 20$ bytes, with k ranging from 1 to 20 flows, and p ranging from 0.7 to 1. As we will see in next sections, the value of 0.7 is very pessimistic. We have also depicted *BWR* in Fig. 5 for different number of samples per packet and a fixed value of $p = 0.95$. Finally, Fig. 6 represents *BWR* for $k = 10$ and different values of p.

We can extract a first conclusion from these three graphs: TCRTP multiplexing always saves bandwidth with respect to native RTP, as *BWR* is always below 1 even for $k = 1$ flow. The reason for this is that we are using an algorithm that compresses the 40 bytes of headers for every RTP packet, despite the 25 added bytes corresponding to the common header.

We can extract another consequence from Fig. 4: k has a stronger influence than p, i.e. the increase of the number of multiplexed flows is more important than the probability of having a COMPRESSED_RTP header.

BWR can be significantly reduced by the increase of k, but the graphs in Fig. 5 show an asymptote determined by the value of BWR_{RH}. The minimum values for BWR_{RH} are 0.32 for $S = 10$ bytes, 0.43 for $S = 20$ bytes, and 0.51 for $S = 30$ bytes. So these are the limits of bandwidth saving. In conclusion, the increase of k has an effect on *BWR*, but the existence of an asymptote makes it less important for big values of k.

For example, for $S = 20$ bytes, when $k = 20$ flows, the distance to the asymptote is only 0.04. So if we have a big number of flows to multiplex, perhaps grouping them into a single tunnel will not be the best solution: bandwidth saving only grows a little, but packet size will grow significantly, and this may harm the traffic depending on buffer and network behavior. So it would be more interesting to group the flows into a number of tunnels. We will study this effect in the results section.

Fig. 6 shows the influence of the compressing algorithm. If there are not many FULL_HEADER transmissions, which means that p is next to 1, then a smaller value of $E[RH]$ can be obtained. But we see in the fig. that this parameter does not strongly affect *BWR*, as it only achieves a small performance improvement.

On behalf of clarity, in Table 1 we have included the bandwidth values for different numbers of RTP and TCRTP multiplexed flows.

Table 1
Bandwidth of RTP and TCRTP at IP level in kbps.

Number of flows	5	10	15	20
RTP	120	240	360	480
TCRTP	62	115	168	221

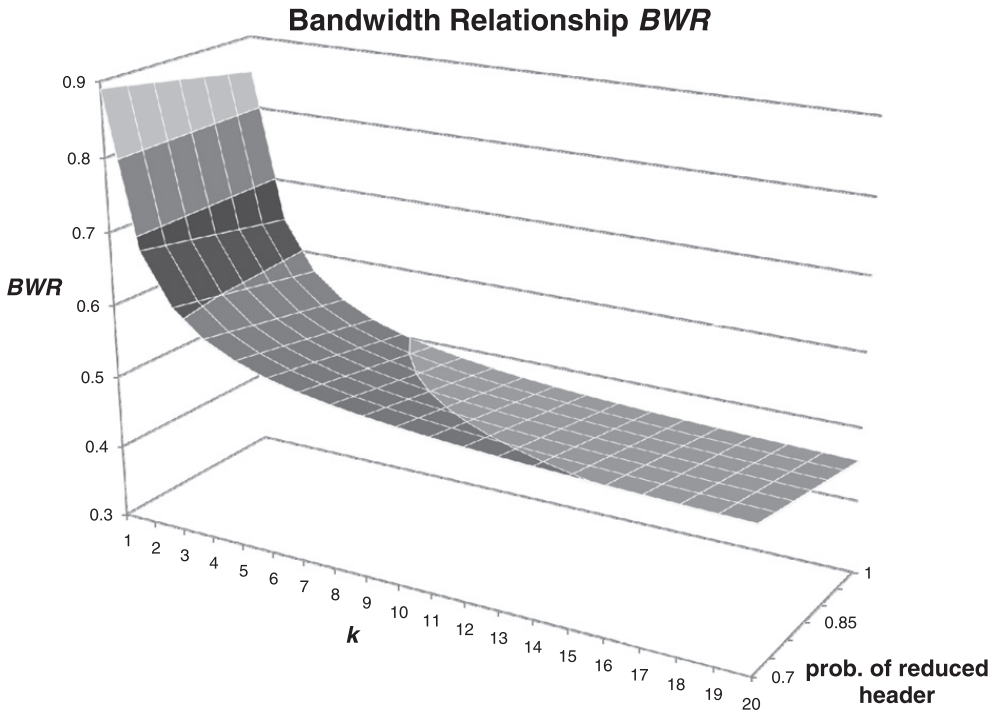

Fig. 4. Bandwidth relationship *BWR* for $S = 20$ bytes.

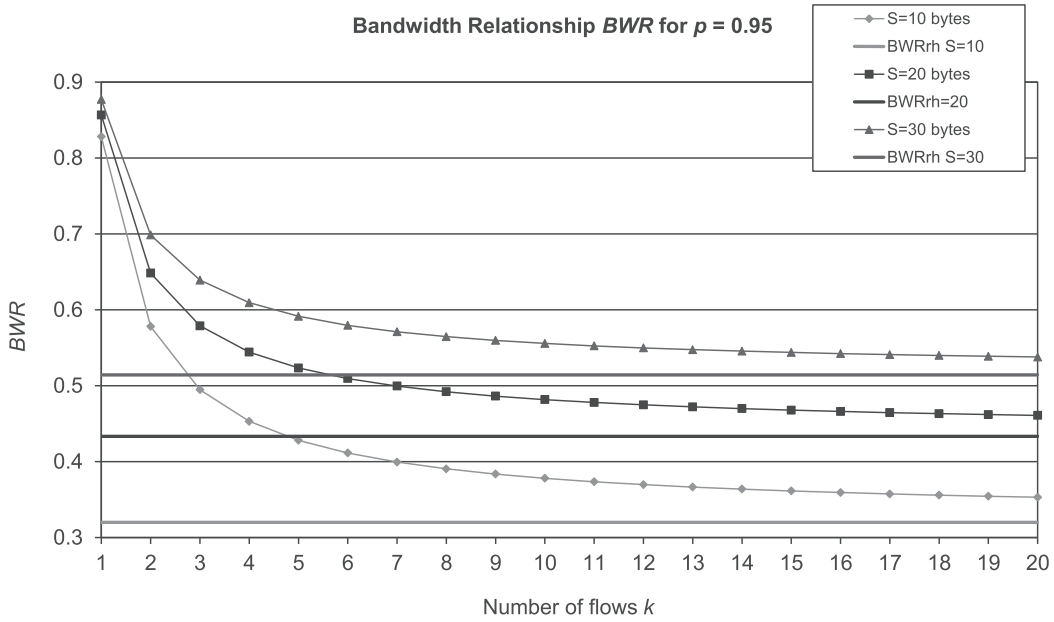

Fig. 5. Bandwidth relationship *BWR* for $p = 0.95$.

3.3. Packets per second

Finally, we study the decrease in terms of pps that can be achieved by means of multiplexing. As explained in previous sections, the reduction of this parameter can be interesting depending on the processing capacity of our router. The behavior of this parameter is quite simple: when multiplexing, the number of packets generated is divided by a factor of k, and this may be an advantage.

This gain will not affect RTCP, the protocol that works with RTP; as said in [26], its traffic must not exceed 5% of total RTP traffic. This is the reason why in this work we do not consider RTCP multiplexing, i.e. RTCP will work normally between the extremes of the communication. Thus, the number of RTCP packets will remain the same.

In Fig. 7 we present the number of packets per second which will be generated by each of the possible distributions when multiplexing 40 RTP flows. The parameter l represents the number of tunnels, and k is the number of multiplexed flows of each tunnel, always with $l \times k = 40$. These grouping options have been selected as a simple manner of comparing the different possible combinations.

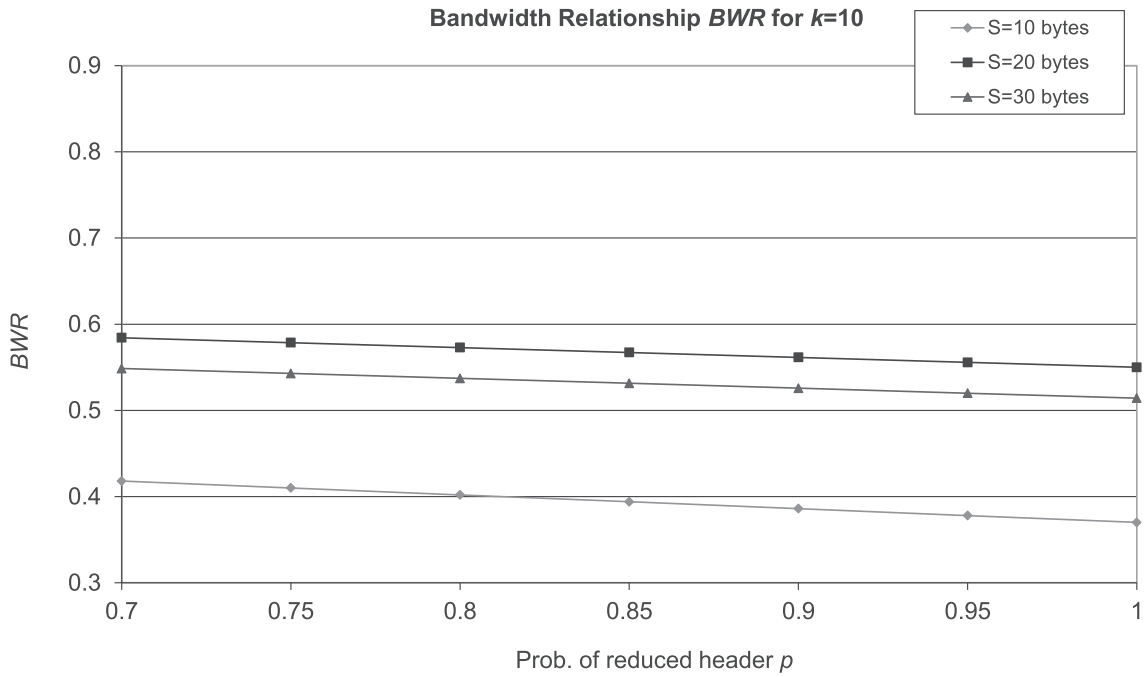

Fig. 6. Bandwidth relationship *BWR* for $k = 10$.

We have also added a 5% of traffic corresponding to RTCP packets, which are the same for every case, as they are not multiplexed.

The figure shows that, logically, the lowest amount of pps is achieved when only one tunnel is used and three samples are included in each packet ($S = 30$ bytes). It may also be noticed the very high packet rate of native RTP, if compared with multiplexing schemes. This is another advantage of multiplexing: the significant reduction of this parameter.

The saturation of the router in terms of pps has to be avoided, so we have to calculate the total amount of pps, adding voice and background traffic (e.g. for a background traffic of 1600 kbps, our traffic distribution [27] generates 290 pps), and assuring that it is not above the limit of the router.

3.4. Multiplexing tradeoffs

As a summary of this section, we will consider the triple tradeoff of multiplexing: bandwidth, packet size and pps are the three parameters we can modify by the use of different RTP multiplexing schemes, tuning the number of samples per packet and the distribution of the flows.

In Fig. 8 we present a diagram which includes the three variables. We have depicted it for 40 RTP flows (a) and for 2 tunnels of 20 flows each (b).We have three limits: first, the router has a pps limit; secondly, the connection has a hard bandwidth limit, so these axes have a fixed limit, which will be reduced when considering background traffic. Finally, the "packet size" axe has only the MTU limit but, depending on the behavior of our router and network with

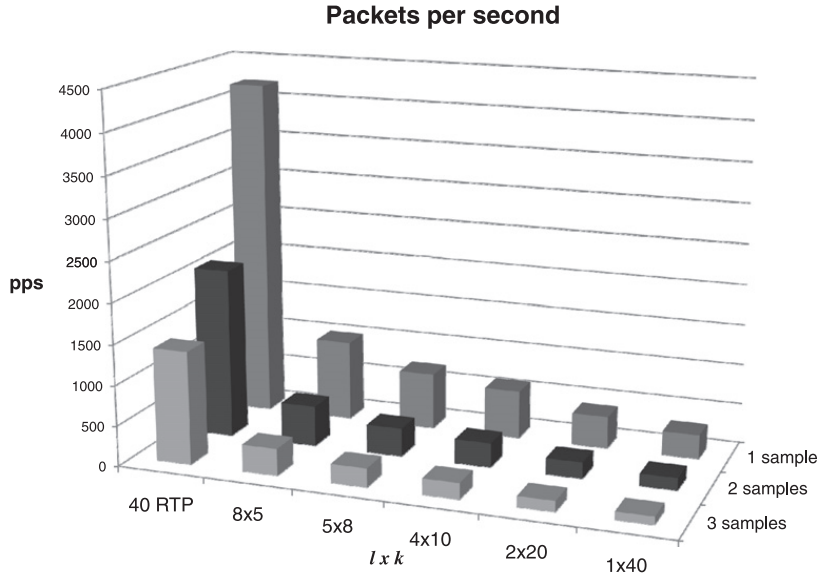

Fig. 7. RTP and RTCP packets per second as a function of the number of samples and the distribution of the tunnels.

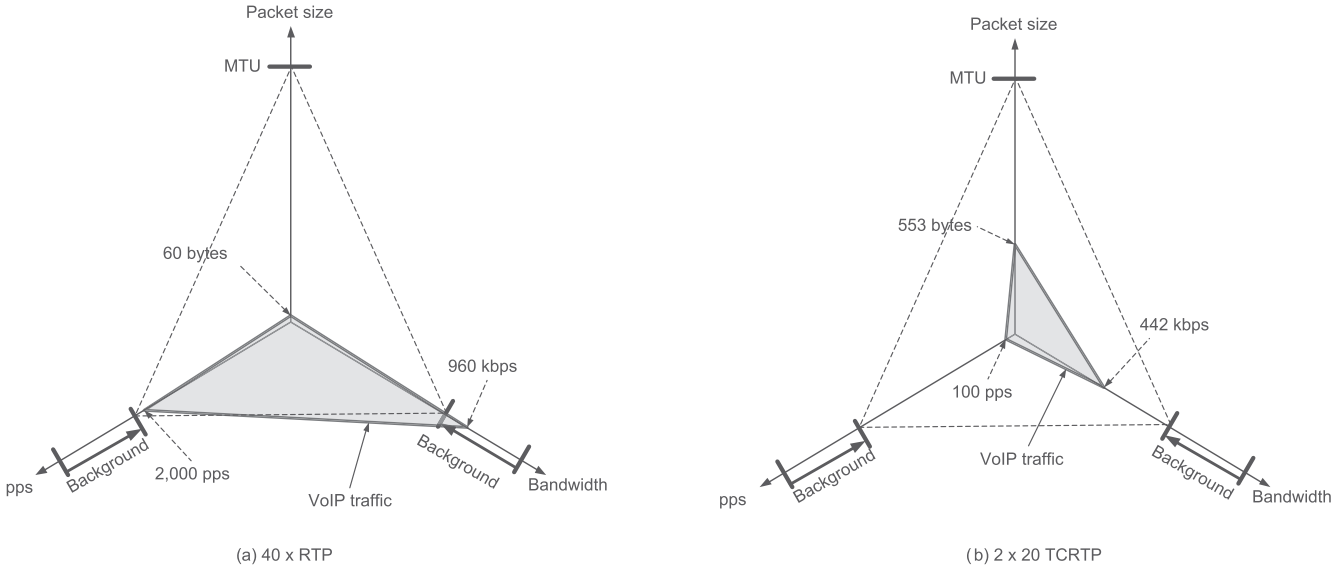

Fig. 8. Bandwidth, pps and packet size tradeoff for $k = 40$.

respect to packet size, we could be interested on small or next-to-MTU sizes. The discontinuous triangle represents the limit of our router and connection, taking into account background traffic, and the solid one represents the parameters of the used scheme.

The decision of the RTP multiplexing scheme to use allows us to adapt the VoIP traffic to our router and our access network, avoiding the values which are next or above the limits. If we are planning a network, we can also consider RTP multiplexing in order to save money by selecting a cheaper router, able to manage less pps, or contracting a connection with less bandwidth.

4. Test methodology

Measurements and tests have been carried out using a testbed [28] based on virtual machines. As said in [29], tests can be carried out using real hardware, simulators, testbeds or emulators. A testbed is defined as "a normal instance of the system under study, (...) but it might not be an all-aspects-scaled-down version of it". The characteristics of each test environment make it more adequate for different purposes, but in our case we have used a testbed in order to use real operating systems and protocol stacks: virtual machines run Linux CentOS distribution.

Three machines have been used: the generator, the router and the receiver, as it can be seen in Fig. 9. VoIP and background traffic are sent from the traffic generator, and then they go through the router which, by the use of Linux tool *Traffic Control* (*tc*), emulates different buffers. We have used the *pfifo* option in order to implement the buffer with a *fixed number* of packets, and the *tbf* (token bucket FIFO) option for the *dedicated bandwidth*, *high capacity* and *time-limited* ones. It permits to set up different buffer sizes with some parameters as latency limit, buffer limit or the size of bursts. *tc* takes into account level-2 headers (Ethernet in our case) to calculate bandwidth limit, so traffic amounts have to be properly corrected. Once the traffic is captured, network delays are added using a statistical model based on Internet measurements, as we will see. This latency is added offline because the testbed has a size limitation and it is not able to emulate realistic Internet network delays. Finally, the effect of de-jitter buffer in terms of delay and packet loss is added, obtaining the final results.

4.1. Traffic generation

Background traffic is generated using D-ITG [30]. We have used the next distribution: 50% of the packets are of 40 bytes, 10% of the packets are of 576 bytes, and the rest,

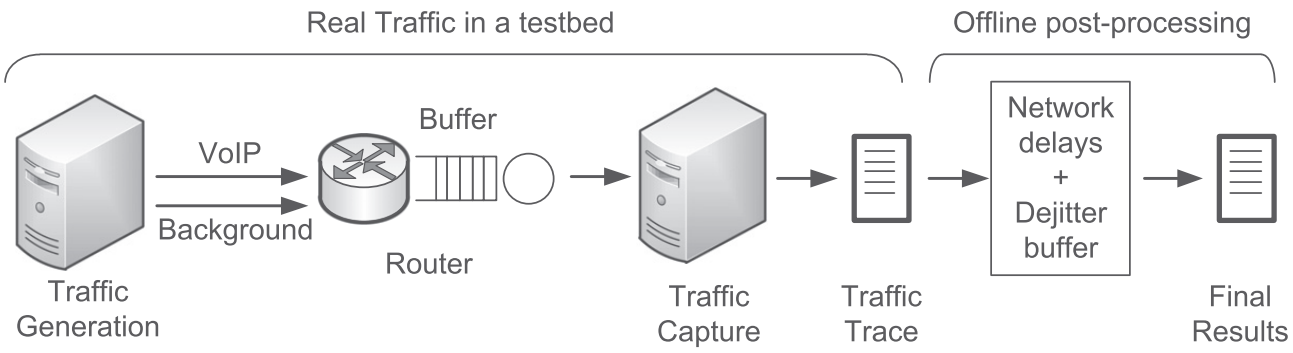

Fig. 9. Scheme of the tests.

40% are of 1500 bytes [27]. UDP has been used instead of TCP, in order to avoid flow control, thus sending always the same background traffic. VoIP traffic is also generated using D-ITG, which permits different statistics for both inter-packet time and packet size.

Multiplexed RTP traffic has to be characterized with statistical models in order to obtain a realistic behavior. Ref. [25] presents a comparison of CRTP and ECRTP for VoIP applications over satellite links. The obtained values show that, for ECRTP, 97.3% of the packets have a COMPRESSED_RTP header, while 2.6% have a COMPRESSED_UDP one. The percentage of FULL_HEADER packets is very small (0.0033%), and we will consider it negligible, so a binomial distribution will be used as a particular case of the multinomial one.

We have modeled TCRTP's behavior in terms of packet size, adding the correspondent number of extra bytes for each COMPRESSED_UDP packet, according to the binomial distribution depending on the number of multiplexed packets k. These extra bytes correspond to a time stamp and an absolute IPID, which have to be updated.

For each measurement, 400 s of real traffic have been sent in a scenario similar to the one shown in Fig. 9. Later, the first and last 20 s have been discarded in order to get a stationary behavior. No silence suppression has been used.

4.2. System delays

A multiplexer-demultiplexer system has to be transparent for the communication ends: the packet sent from the origin and the packet received at the end have to be exactly the same, so the demultiplexer needs information in order to rebuild the original packet and deliver it to its destination. Taking this into account, we will now summarize the different delays that have to be considered in our system. They are illustrated in Fig. 10.

- Packetization delay ($T_{\text{packetization}}$): It depends on the codec. In this work we always use G.729a, which delay is 10 ms for each sample plus 5 ms corresponding to look-ahead time. So, e.g. if we use two samples per packet, this delay will be 25 ms.
- Retention time ($T_{\text{retention}}$): The multiplexer has to wait in order to receive one packet from each RTP flow. In this study we will assume that RTP sources are connected to a high speed LAN, so retention time can be considered equivalent to the time between packets, as an upper bound (see Fig. 2). Of course, if flows were synchronized, this time could be significantly reduced.
- Processing time in the mux/demux (T_{process}): Ref. [12] built a software prototype of their multiplexing scheme, running under Linux. They observed that the processing mux/demux times caused by packet transmission and header manipulation were below 1 ms. As the packets are bigger when multiplexing, store & forward delay will be slightly increased. In this work we will add 5 ms in order to take into account processing time in the mux and demux, and also store & forward and propagation times in local networks.
- Queuing delay at the origin router's buffer (T_{queue}): The pass from a high speed LAN to the Internet access network constitutes a bottleneck that has to be taken into account. This delay will strongly depend on the buffer implemented at the router.
- Network delay (T_{network}): Packet arrival times are captured after the router, and then a different network delay is added to each packet, using a statistical distribution. We have used the model proposed in [31], which is based on the results of some global measurement projects [32]. It consists of a fixed minimum delay depending on geographical distance between the two nodes, plus a lognormal distributed delay that is applied to each packet. By default, we have considered an intra-region scenario, and we have used values extracted from [33]: 20 ms of minimum One Way Delay (OWD), and for the lognormal distribution, the average was 20 ms with a variance of 5. Some comparatives will use higher delays. We have not considered the network to increase packet loss.
- Queuing delay at the destination router's buffer: It is considered negligible, because we are passing from an Internet access to a high speed LAN.
- De-jitter buffer of the destination application (T_{dejitter}): It adds a new delay and it also increases packet loss, as every packet that does not arrive on time to be reproduced will be equivalent to a lost packet. As we want to avoid the use of a concrete implementation, de-jitter buffer losses have been calculated using an approximation suggested by Cole et al. in [34]:

$$Loss_{de-jitterbuffer} \sim P\{o > bg\} \tag{10}$$

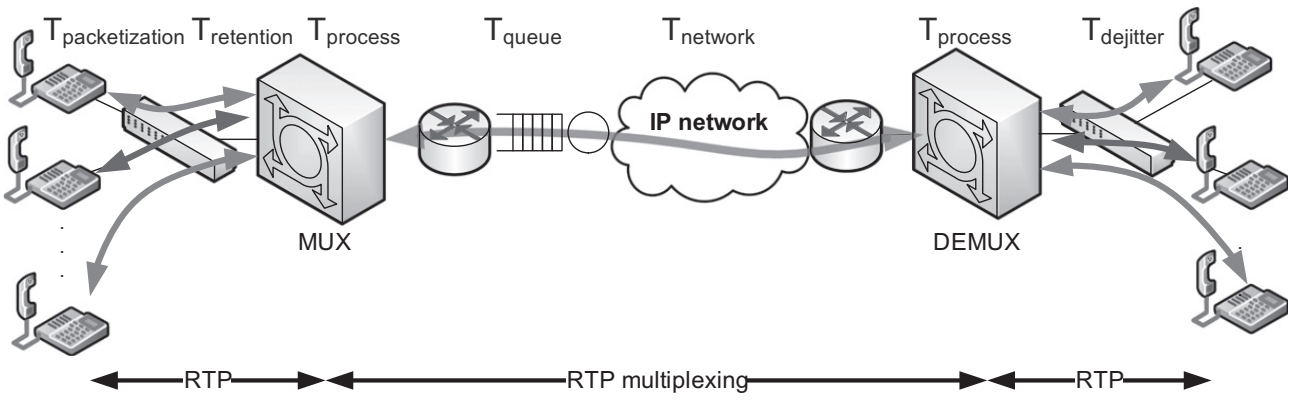

Fig. 10. Multiplexing scheme and delays.

where o is the difference between OWD of consecutive packets, b is half the buffer size, and g represents inter-packet generation time. This approximation supposes a static de-jitter buffer. R-factor ranges from 0 (bad quality) to 100 (high quality). Medium quality is normally considered from $R > 70$. By the use of adaptive schemes, instead of static ones, better results could be obtained. This is the reason why in some graphs we have included values of R-factor smaller than 70. De-jitter buffer size has been calculated in each case to maximize R-factor, obtained with the analytical expression also proposed in [34]:

$$R \sim 94.2 - 0.024d + 0.11(d - 177.3)H(d - 177.3) - I_{ef} \tag{11}$$

where

$$\begin{aligned} H(x) &= 0 \quad if\ x < 0,\ else \\ H(x) &= 1 \quad for\ x >= 0 \end{aligned} \tag{12}$$

I_{ef} represents the equipment impairment factor. This expression presents a knee when the delay is 177.3 ms: if OWD exceeds this value, the quality falls dramatically.

5. Results

In this section we present the results, mainly in terms of R-factor, OWD and packet loss. Different effects like multiplexing gains, the number of samples and the distribution of the flows are studied and compared, taking into account the four buffer implementations considered. Some effects are only interesting when using certain buffers. We will extract some conclusions but, for clarity, we will leave some discussion for the next section.

We have compared multiplexing schemes with native RTP, but not with CRTP or ECRTP, because these protocols operate link by link, so they are not adequate for our Internet scenario. The used codec is G.729a. By default, two samples per packet are used, but some comparatives also include some measurements using one or three samples.

5.1. Dedicated bandwidth

The first objective of TCRTP is bandwidth saving by means of multiplexing and header compression. So we have done some measurements in order to illustrate this. First, we have studied what happens if some bandwidth is reserved for VoIP packets. We have sent different numbers of RTP flows using a *tc* limit of 200 kbps of *dedicated bandwidth*. So we can expect the system to behave well while the VoIP bandwidth is smaller than the limit. Fig. 11 shows R-factor as a function of the number of flows k. It can be seen that, e.g. using native RTP with two samples per packet, only 6 flows are supported (7 flows × 29 kbps at Eth level is above the 200 kbps limit), while TCRTP using the same number of samples supports up to 17 flows. The overhead of the multiplexing scheme is shared by all the flows, but in the case of native RTP the required bandwidth simply increases by a factor of k (the number of flows).

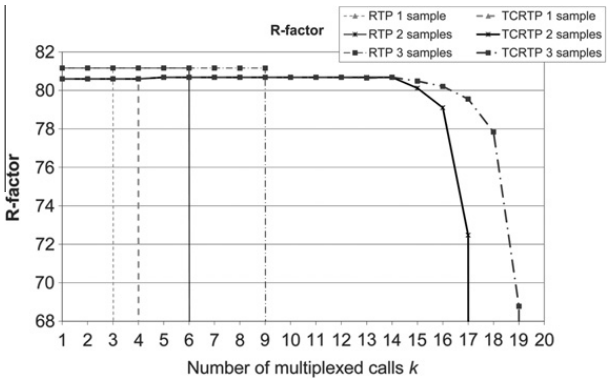

Fig. 11. Comparative using 200 kbps of *dedicated bandwidth*.

5.2. Buffer with a fixed number of packets

Other implementation that can be found in access routers is a buffer with a *fixed number* of packets, i.e. instead of having a buffer capacity measured in bytes, the limitation is in the number of packets which can be simultaneously stored. In the current study we will consider this buffer, but we will set the parameters in order to have a reduced number of maximum packets. This policy has been implemented using Linux *tc* with the *pfifo* option, setting the corresponding parameter to a maximum amount of 50 packets. Another queue has also been included in order to limit the bandwidth to 1 Mbps. We have selected this small number in order to clearly observe the effect of this implementation. If the number of packets was very big, the buffer will behave like a *high capacity* one, which will be studied in the next subsection.

Fig. 12 presents R-factor, when using native RTP (a) and TCRTP (b), as a function of background traffic, for 5, 10, 15 and 20 simultaneous VoIP flows. It can be seen that TCRTP achieves a slightly smaller quality for lower values of background traffic, due to retention time, but it is able to provide an acceptable service when higher amounts of background traffic are present. For example, when 20 flows are sent, if native RTP is used, only 450 kbps of background traffic are tolerated, but this fig. grows up to 750 kbps when using TCRTP.

If we want to get a clearer idea of this behavior, we have to observe the graphs of OWD (Fig. 13) and packet loss (Fig. 14). It is also interesting to calculate the average packet size of the traffic that fills the buffer in each case, taking into account both VoIP and background packets. Table 2 summarizes these values. As background packets are of 40, 576 and 1500 bytes, the use of native RTP will reduce the average packet size, but TCRTP will increase it. The differences can be significant, e.g. for 20 native RTP the average packet size is 101 bytes, while for TCRTP it is 626 bytes, when 400 kbps of background traffic are sent.

If we look at Fig. 13, it can be seen that the delay grows up, but not significantly, as only 50 packets can be stored at the buffer, and the average packet size is small (between 100 and 320 bytes). On the other hand, the delays added

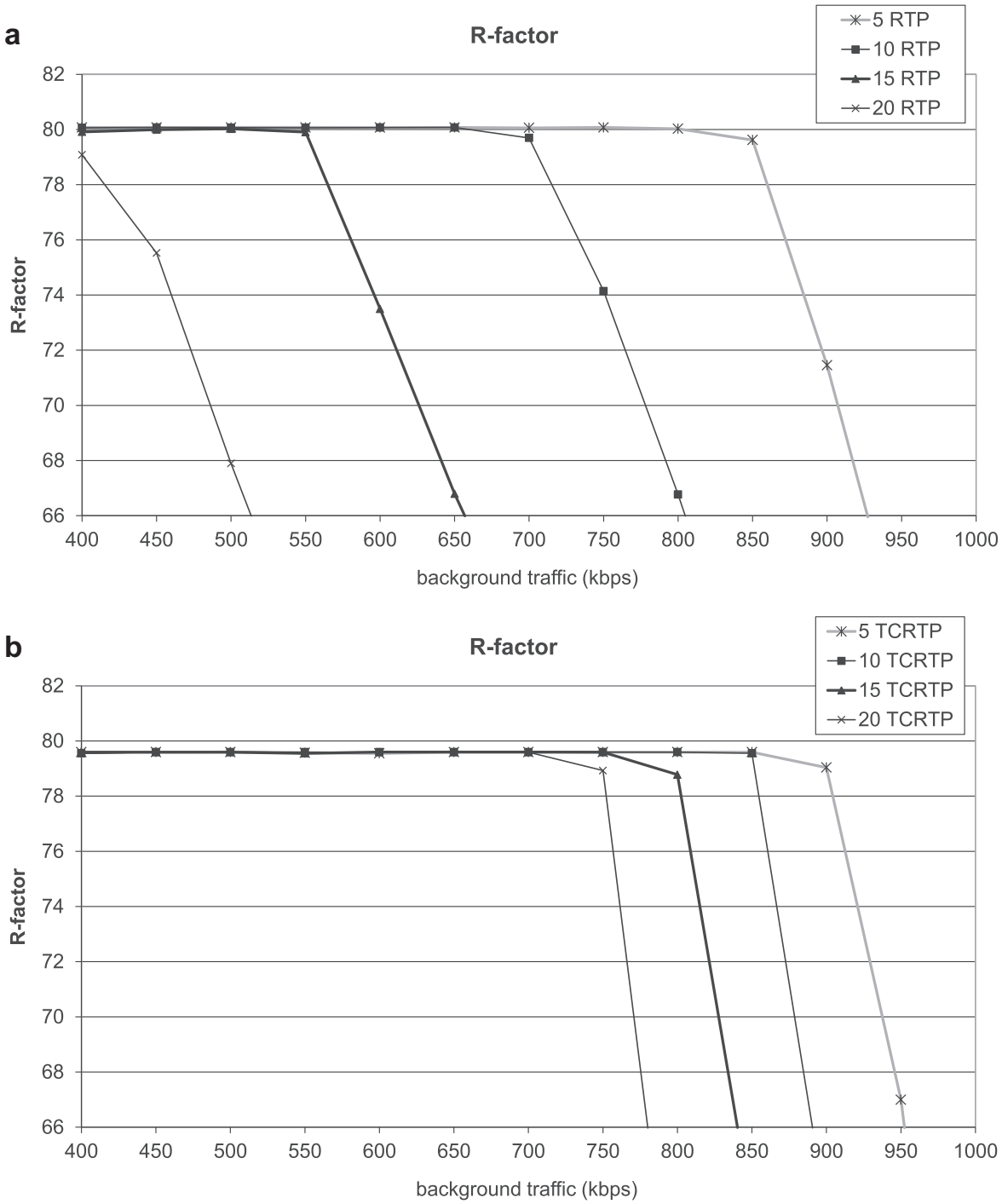

Fig. 12. *R*-factor when using (a) native RTP; (b) TCRTP for *fixed number* buffer.

to TCRTP packets are higher: although only 50 packets can be stored, they are bigger (between 465 and 650 bytes). We can conclude that RTP has an advantage regarding the delay when this buffer is used.

But if we represent packet loss (Fig. 14), we see that this parameter behaves significantly worse for native RTP: values of about 35% are reached, while they are always below 14% for TCRTP. The reason for this is that the amount of pps generated for TCRTP is significantly smaller, as we have seen in previous sections. A TCRTP packet is considered as a single IP packet, although it includes many compressed RTP ones, so if the router has this buffer implementation, multiplexing can be an important improvement. This is a case in which the reduction in terms of pps represents a clear advantage.

5.3. High-capacity buffer

Next, we will study multiplexing behavior when a big buffer is used. We will assume a single buffer with a very big size. In our testbed we have used an 800 ms limited queue in order to emulate this kind of buffer. The band-

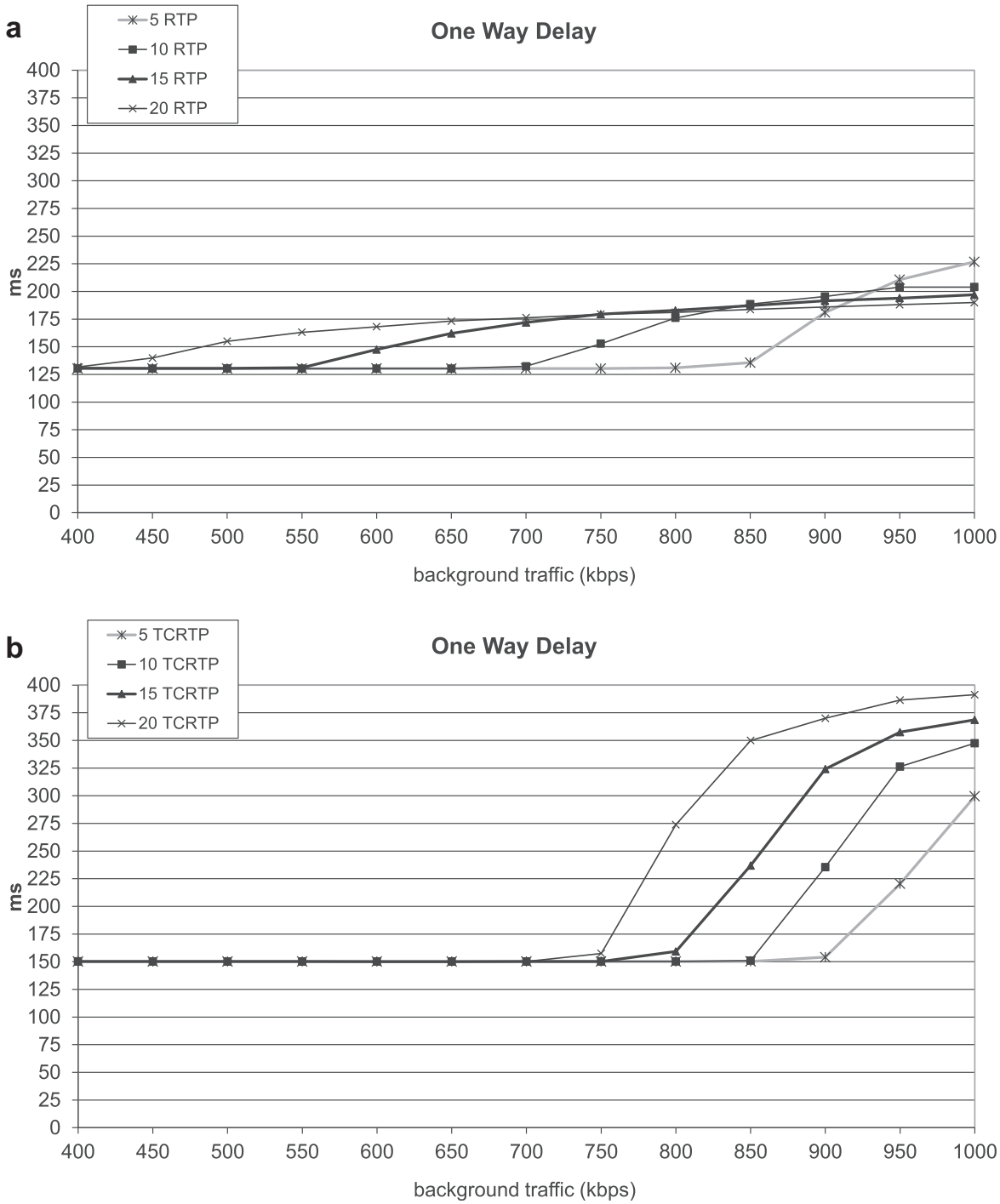

Fig. 13. OWD when using (a) native RTP; (b) TCRTP flows for *fixed number* buffer.

width limit in this case is 1 Mbps. With this buffer, if bandwidth limit is reached, delay grows dramatically, above the required limits for VoIP packets. Fig. 15 shows *R*-factor as a function of background traffic, with different numbers of VoIP flows. It can be seen that the behavior is similar to the one obtained with *dedicated bandwidth*: when the bandwidth limit is reached, *R*-factor gets unacceptable. In these tests the destination application uses a de-jitter buffer with a fixed size of $b = 3$ samples.

We will now study the influence of the number of samples included into each RTP packet. We find again the trade-off between network efficiency and delay. On the one hand, if we use native RTP with only one sample, we save 10 ms of packetization time, plus 10 ms of retention time, but at the price of sending twice the number of packets, thus spending more bandwidth due to the fixed header size of 40 bytes. On the other hand, if we increase the number of samples to three, the efficiency will be improved, but more delays are added due to packetization and retention delays.

In Fig. 16 we can observe two simultaneous effects: on one hand, when TCRTP is used, the added delays produce an impairment on *R*-factor for small amounts of band-

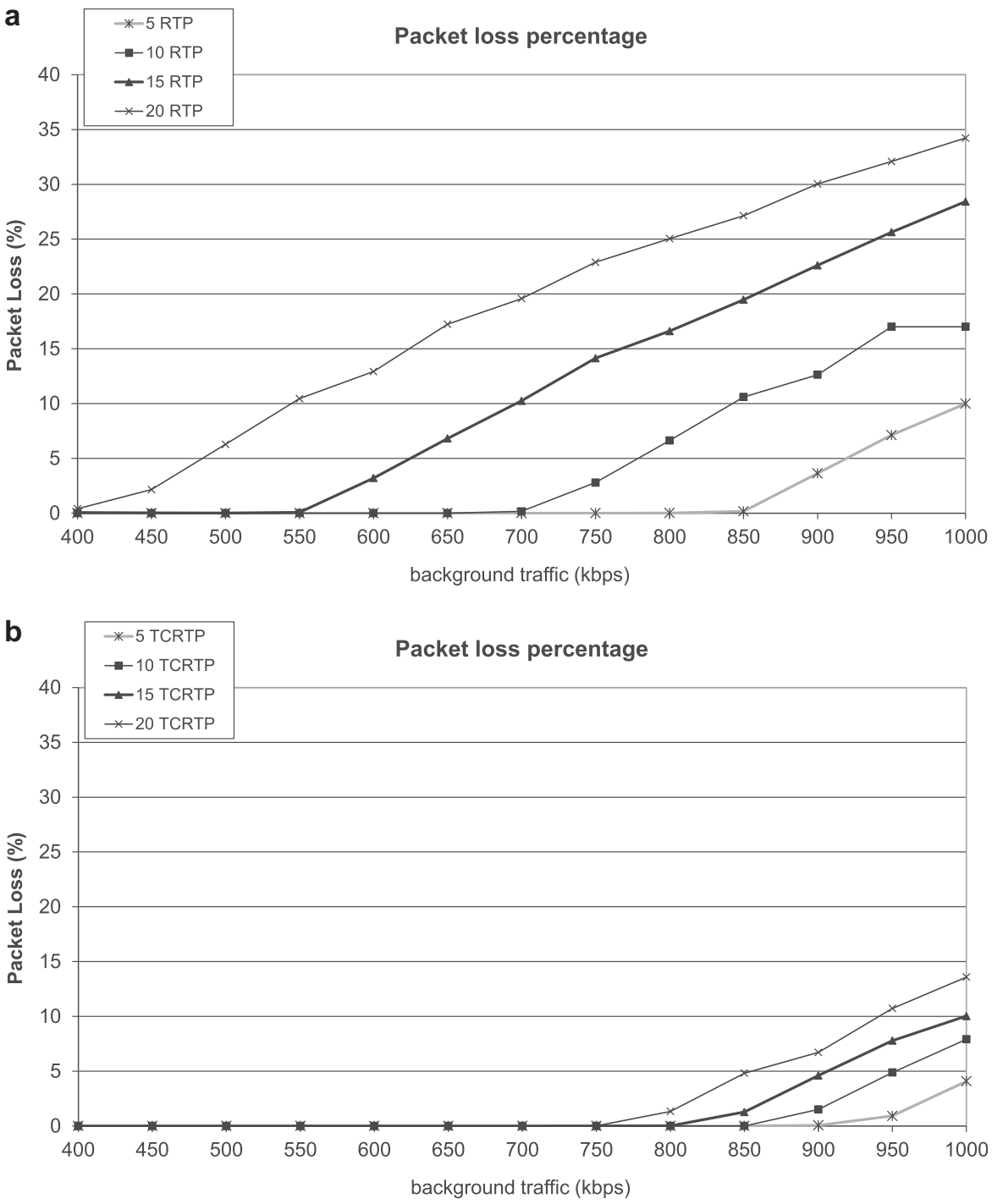

Fig. 14. Packet loss when using (a) native RTP; (b) TCRTP for *fixed number* buffer.

width. But on the other hand, multiplexing makes it possible to have a good conversation quality with bigger amounts of background traffic. As an example, if native RTP with one sample per packet is used, although the quality is good, the overhead will be big, so only 450 kbps of background traffic can be supported. If we multiplex, background traffic can grow up to 800 kbps while maintaining an acceptable quality for the calls.

As said in Section 4.2, the previous results have been obtained using an average network delay of 40 ms: 20 ms fixed plus a lognormal one of 20 ms with a variance of 5. But network delays depend on many factors, especially geographical distance [31]. So we have also studied this effect, as the extremes of communication can be in different geographical zones. Fig. 17 shows OWD for network delays ranging from 40 to 100 ms average. We see that the delay does not change with background traffic while there is bandwidth enough, but when the limit is reached, it grows to unacceptable values.

Fig. 18 shows *R*-factor, and it shall be noticed that if OWD gets above 177.3 ms, which occurs for network delays of 80 and 100 ms, *R* gets significantly worse.

If we want to combine the effect of network delays and the number of samples per packet, we have to take into ac-

Table 2
Average packet size at IP level including VoIP and background traffic (in bytes).

	400 kbps background	700 kbps background	1000 kbps background
5 RTP	198.84	267.92	319.58
5 TCRTP	465.54	530.73	565.27
10 RTP	138.21	185.00	224.32
10 TCRTP	519.04	567.79	593.61
15 RTP	114.44	149.36	180.21
15 TCRTP	572.55	604.84	621.95
20 RTP	101.75	129.54	154.77
20 TCRTP	626.02	641.87	650.27

count the fixed delays of our network. In Fig. 19 we have represented *R*-factor with 40 and 100 ms of network delay, for one, two and three samples per packet. We see that in the case of a high network delay, we are forced to avoid the use of three samples per packet (*R* becomes smaller than 70), due to fixed delays.

We will now present a study of how different grouping schemes for a number of TCRTP flows can modify QoS. In this case, 40 calls using G.729a codec with 2 samples per packet are sharing the same link between two extremes. We will study the variation of *R*-factor with different values of *k*, which is the number of multiplexed flows of each tunnel, and *l*, the number of tunnels, satisfying the equation $l \times k = 40$ calls.

In this case the testbed is used to send both desired and background traffic through a 2 Mbps link. Table 3 shows the required bandwidth for different values of *k*, and also the average packet size at IP level. Measurement parameters are the same as in previous subsections, but in this case de-jitter buffer size has been set to $b = 2$.

Fig. 20 shows the obtained *R*-factor for different values of *k*. Its behavior is good until bandwidth limit is reached, and then it falls dramatically. In this case, the best behavior is obtained for $k = 40$ and $k = 20$. The behavior is very similar, as the bandwidth used is roughly the same.

5.4. Time-limited buffer

Finally, a *time-limited* buffer has been tested. We will deeply study it as we consider it useful for real-time and interactive services, like VoIP, where delay has to be maintained under a limit in order to provide a service similar to traditional telephony. The connection bandwidth is 1 Mbps. The buffer has only one queue and discards every packet that spends more than 80 ms on it. The buffer, which has been implemented using the *tbf* queue of Linux *tc*, has a value for the number of tokens, and each packet needs to *take* a number of them, depending on its size, in order to be accepted. The tokens arrive at the queue using a fixed rate. Logically, this buffer makes big packets have a bigger probability of being dropped than small ones, which also happens with fixed size queues. This is an advantage for native RTP packets, as they are small, but it is a disadvantage for multiplexed packets, as they are bigger than non-multiplexed ones, so they will be dropped in a higher percentage. Using this buffer, *R*-factor is expected to go down more slowly than for big buffer, as voice packets have this advantage. There are two simultaneous effects: multiplexing saves bandwidth, but at the cost of generating bigger packets and thus increasing dropping probability. So it can be interesting to study in which cases one effect is more important than the other.

Fig. 21 shows a comparative for different values of *k*. If we compare it with Fig. 15, it shall be noticed that the obtained values of *R*-factor for small background traffic

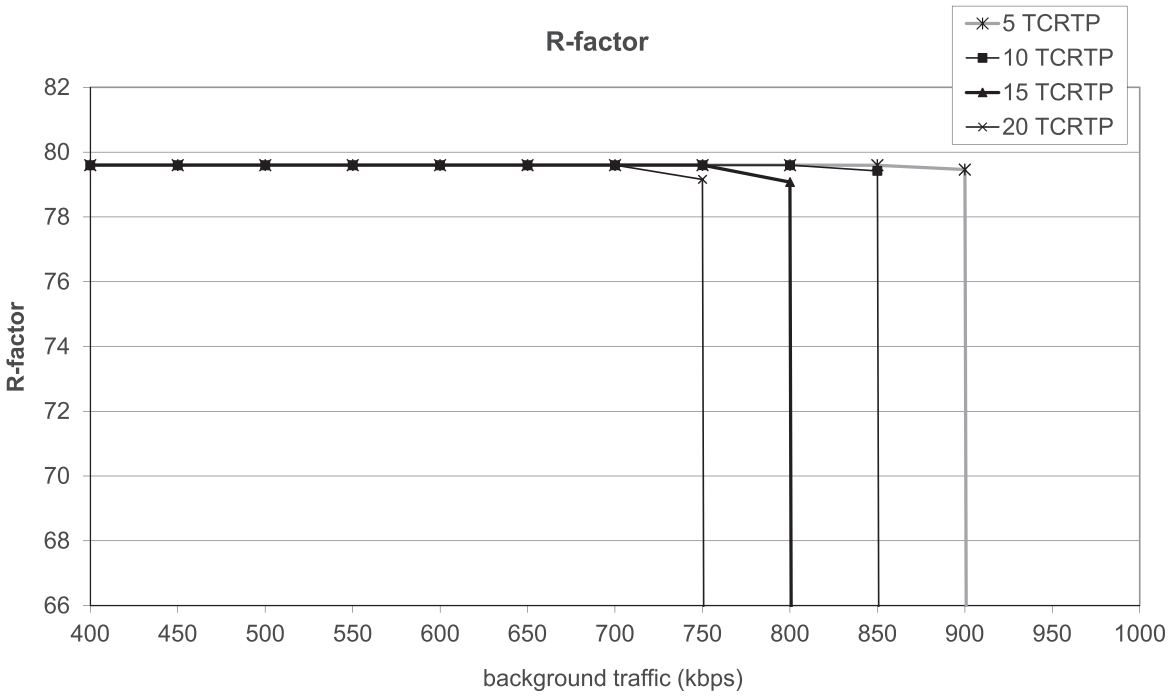

Fig. 15. *R*-factor for different values of *k* and *high capacity* buffer.

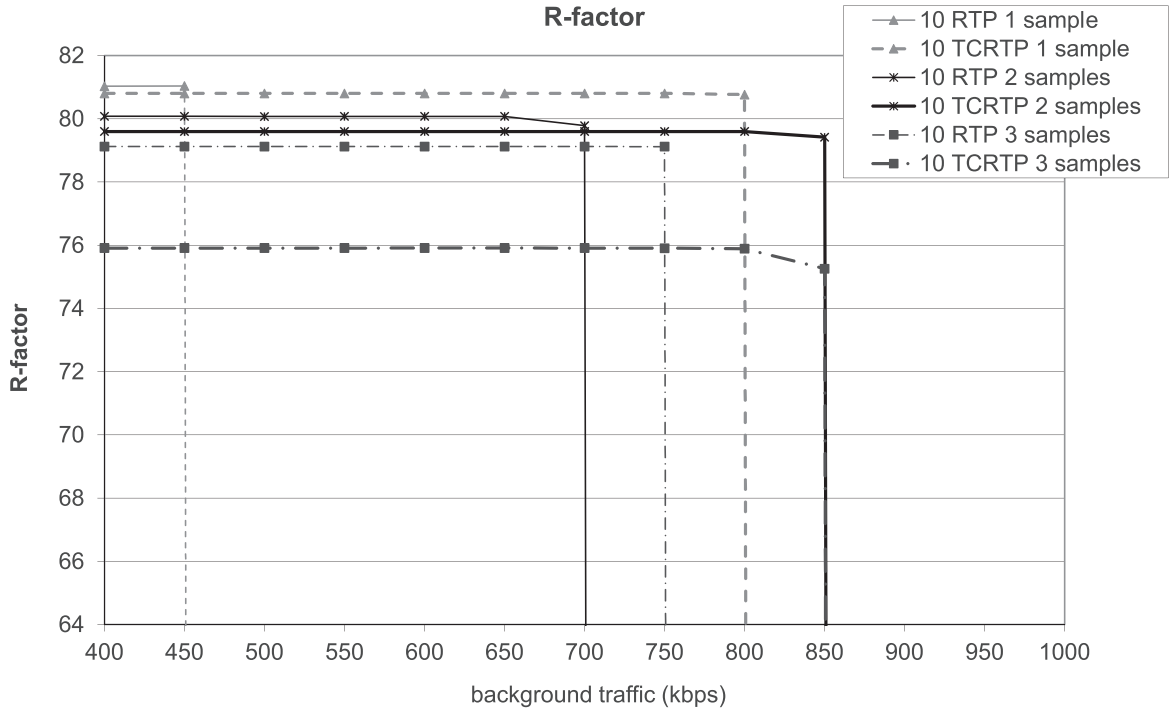

Fig. 16. *R*-factor with 10 multiplexed flows and different number of samples for *high capacity* buffer.

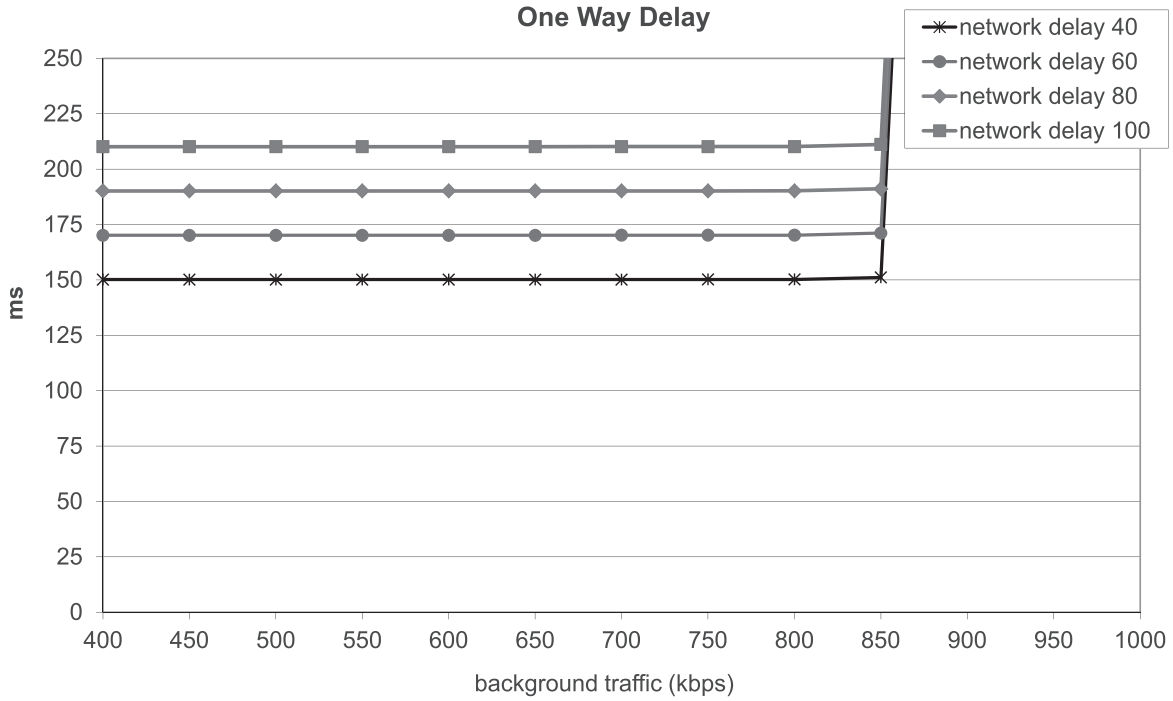

Fig. 17. OWD with 10 multiplexed calls and different network delays for *high capacity* buffer.

are the same as the ones obtained for *high-capacity* buffer, but the graphs have changed from a step-like shape to a lower slope ones, thus allowing a higher background traffic amount until they reach the value $R = 70$. The cause is that this buffer penalizes background packets of 1500 bytes, as they have a higher probability of being discarded.

Fig. 22 shows *R*-factor improvement when using TCRTP instead of native RTP. First, it must be noticed that for small background traffic, the impairment of multiplexing is below 1%. This is mainly caused by the additional delays introduced by multiplexing.

But when background traffic grows, we observe that above 5 multiplexed flows, the use of multiplexing represents a good improvement, gaining up to 21%. This effect is caused by the bandwidth saving of TCRTP with respect to RTP.

Finally, we see that when background traffic is 95% of the limit, native RTP again achieves a better result than

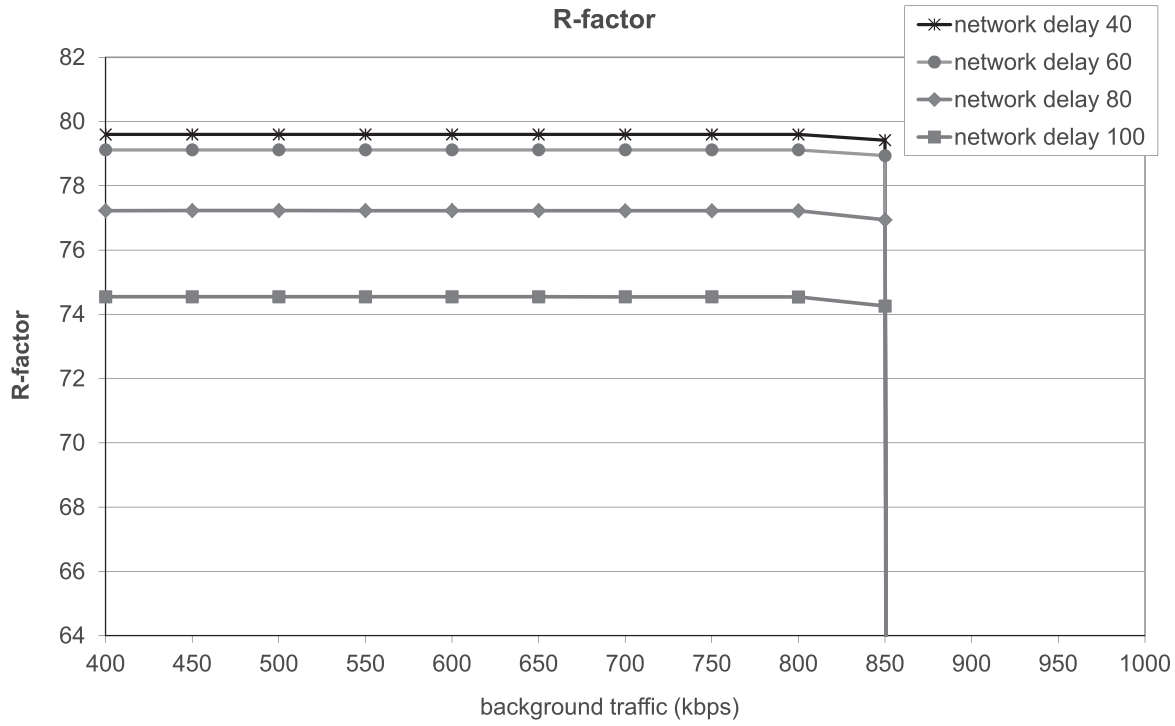

Fig. 18. *R*-factor with 10 multiplexed calls and different network delays for *high capacity* buffer.

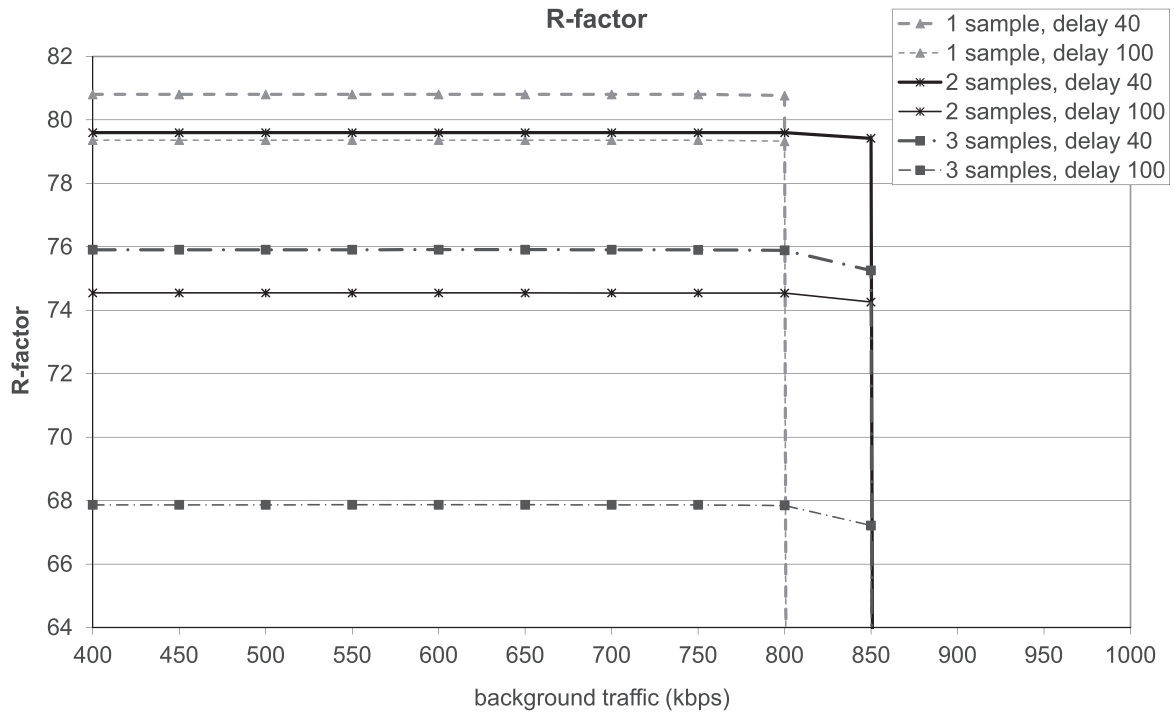

Fig. 19. *R*-factor with 10 multiplexed calls, different network delays and different number of samples per packet, for *high capacity* buffer.

Table 3
Average packet size at ip level (in bytes), and bandwidth (in kbps).

$l \times k$	1×40	2×20	4×10	5×8	8×5	No mux
Avg. packet size (bytes)	1081	553	289	236	157	60
Bandwidth (kbps)	432	442	462	472	502	960

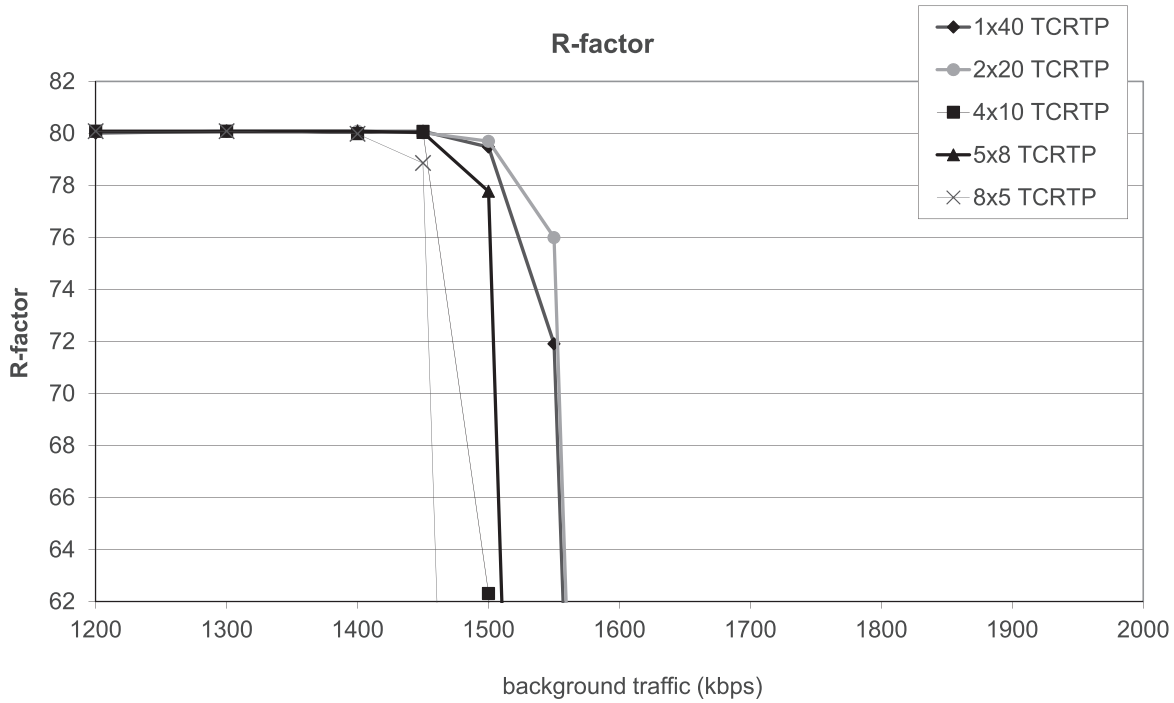

Fig. 20. *R*-factor for different values of *k* using *high capacity* buffer.

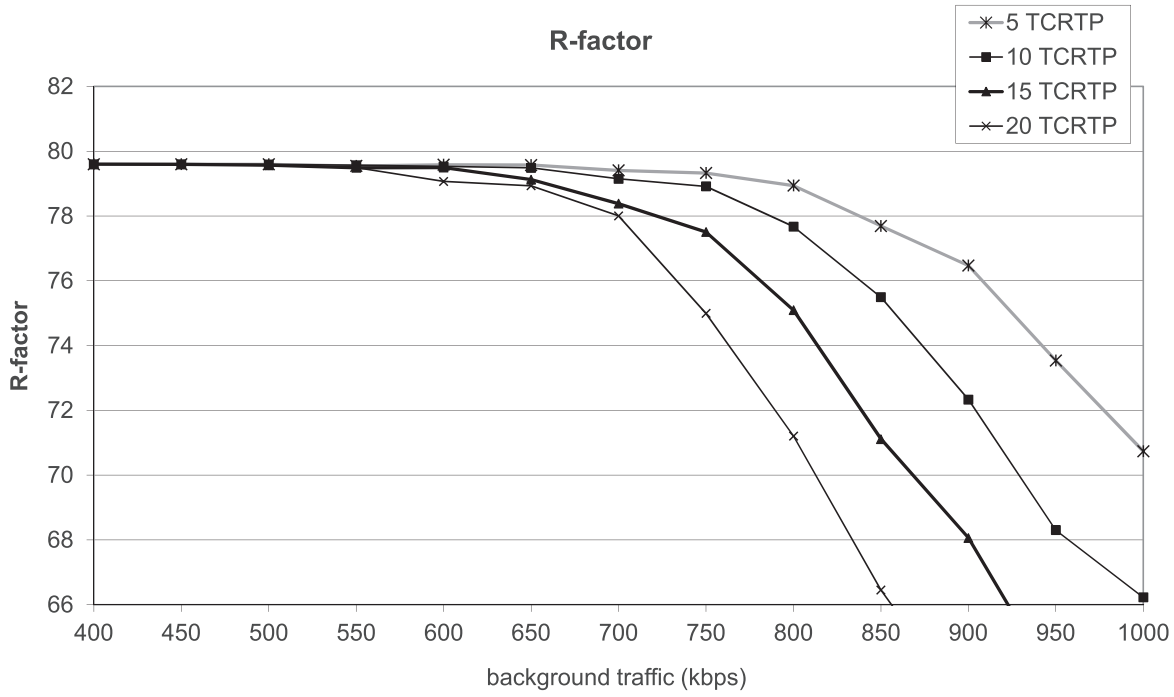

Fig. 21. *R*-factor for *time-limited* buffer.

multiplexing. The cause for this is that multiplexed packets are dropped in a higher percentage due to their size, as we can see in Fig. 23.

By the use of TCRTP we can merge *k* packets into a multiplexed one, so the size of multiplexed packets will also be modified by the number of samples per packet.

Although the bandwidth required is increased when the number of samples per packet is reduced, another effect of this reduction is that the packets get smaller, so this traffic may have an advantage depending on the buffer behavior.

Fig. 24 shows the effect of the number of samples per packet for *time-limited* buffer. For low background traffic, we have obtained the best result for TCRTP with one sample per packet, as it is the solution with the lowest delay. But, as it is not the solution that saves the most bandwidth,

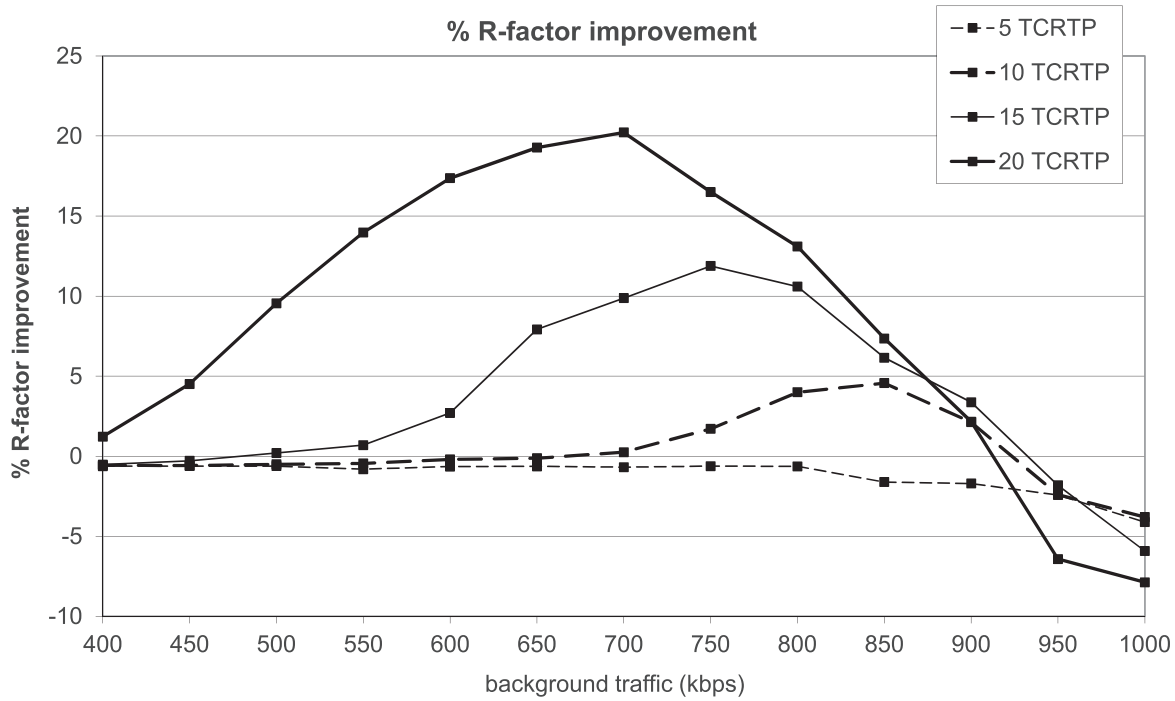

Fig. 22. *R*-factor improvement with $S = 20$ and different values of k.

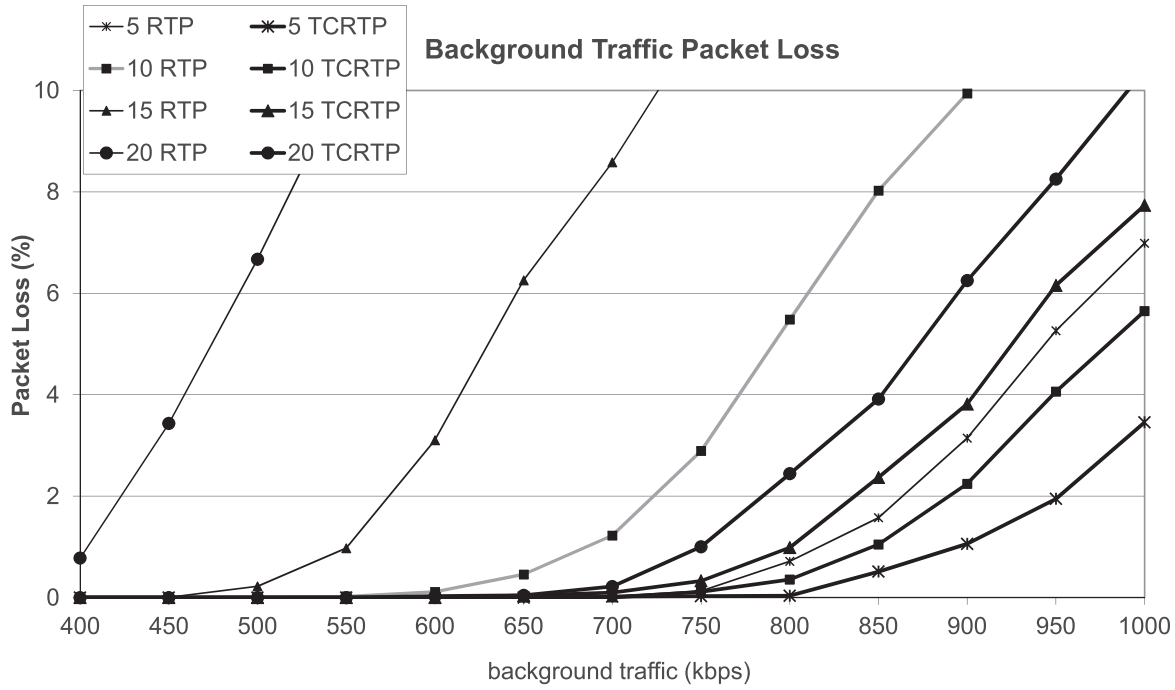

Fig. 23. Packet loss for background traffic with $S = 20$.

background traffic will have a higher loss percentage, as we see in Fig. 25. The option of two samples per packet is able to obtain a value of *R*-factor above 70 with 90% of background traffic. We also see that RTP has a significantly worse behavior than TCRTP for background traffic, due to the bandwidth it requires.

In Fig. 26 and 27 we illustrate the effect of network delay on perceived quality, with $k = 10$ multiplexed flows. When OWD achieves the limit of 177.3 ms, *R*-factor falls. This confirms that network delay is also an important parameter which has to be taken into account.

Fig. 28 shows the combined effect of network delay and the number of samples. We observe that when network delays are high, the use of three samples per packet should be avoided, as the combined effect of the packetization and retention delays makes OWD grow above 177.3 ms.

With regard to the problem of the distribution of many flows into a number of tunnels, we have built Fig. 29 in a

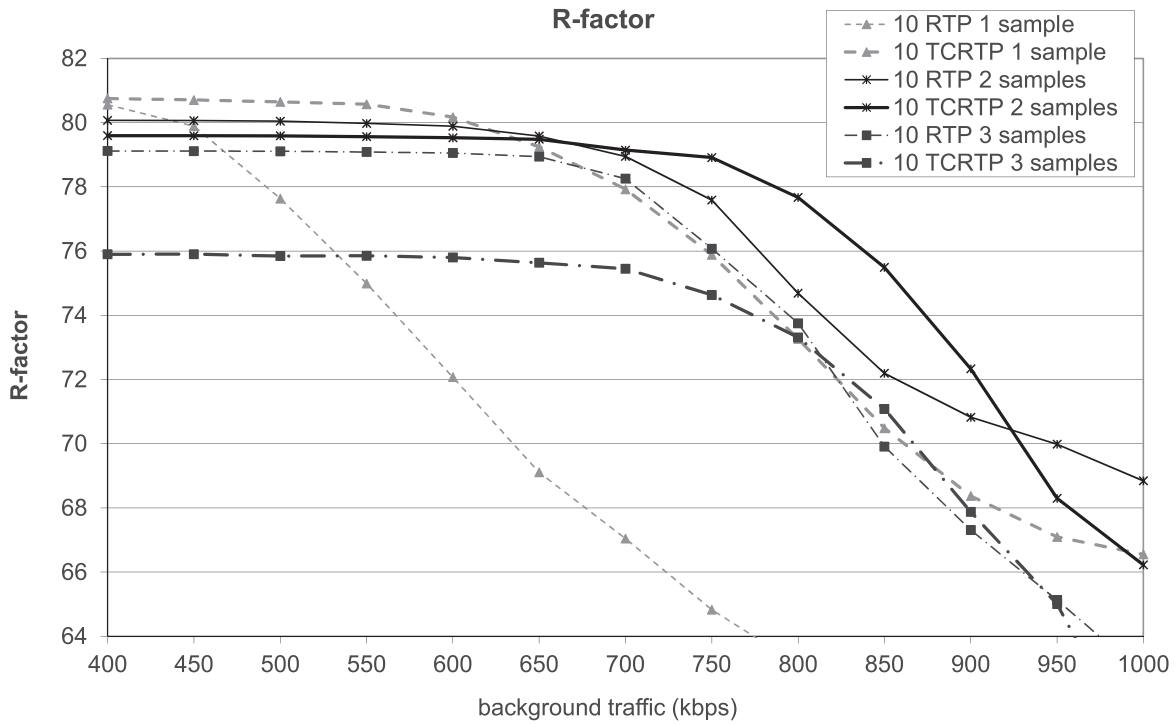

Fig. 24. *R*-factor with 10 multiplexed flows and different number of samples, for *time-limited* buffer.

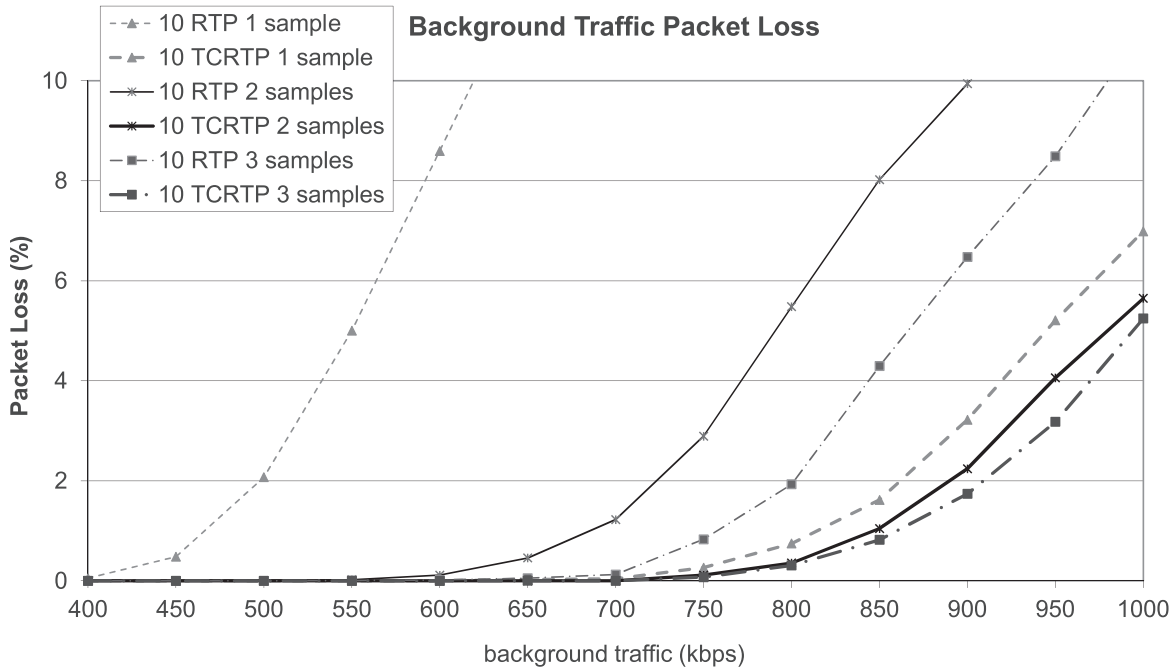

Fig. 25. Background traffic packet loss with 10 multiplexed flows and different number of samples, for *time-limited* buffer.

similar way to Fig. 20, but using a *time-limited* buffer of 80 ms, and 2 Mbps of bandwidth. Once again, the advantage of this buffer with respect to the *high capacity* one is that the slope of the curves is lower, so an acceptable conversation quality ($R > 70$) can be achieved for higher amounts of background traffic. The graph of 40 RTP flows (no multiplexing) has also been included, but it shows a very bad behavior for these values of background traffic, as total offered traffic is above the link capacity: the RTP traffic for 40 flows is about 1160 kbps, and background traffic values presented in the graph begin at 1200 kbps.

By the use of a *time-limited* buffer, we have obtained a different result when varying the distribution of the flows. We observe that the inclusion of all the flows into a single tunnel is not the best solution. There are three distributions which achieve better results.

On behalf of clarity, we have included the most interesting values of Fig. 29 in Table 4, i.e. the ones for 1500 and

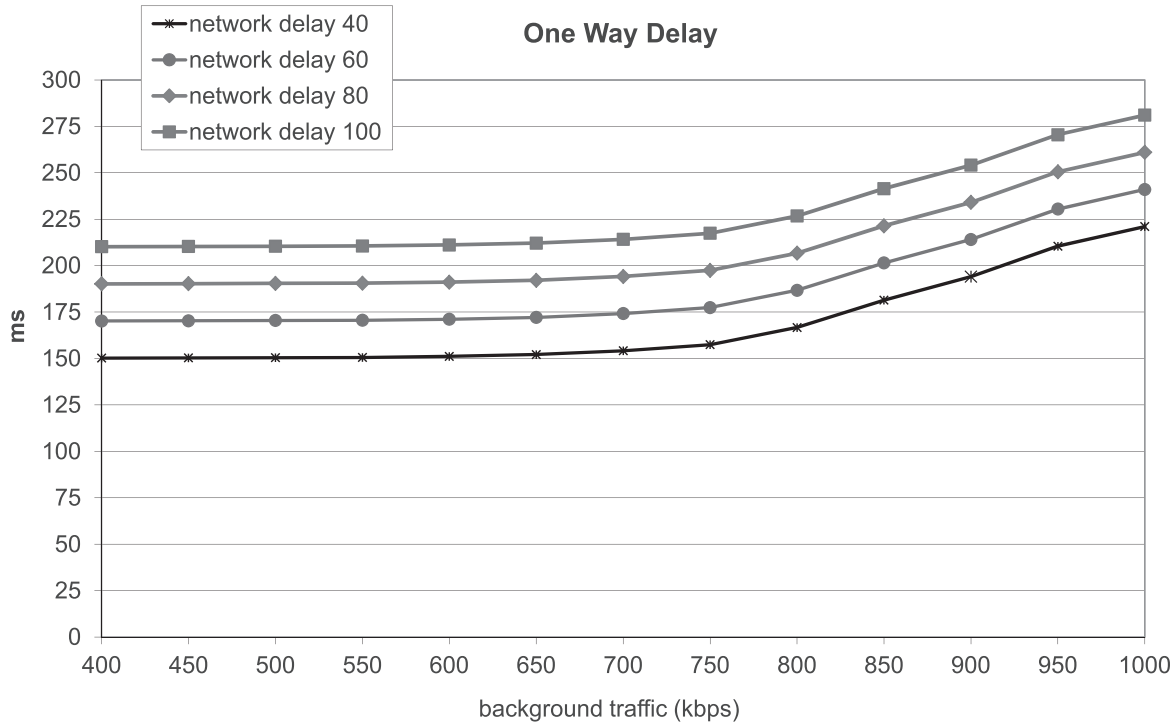

Fig. 26. OWD with 10 multiplexed flows and different network delays, for *time-limited* buffer.

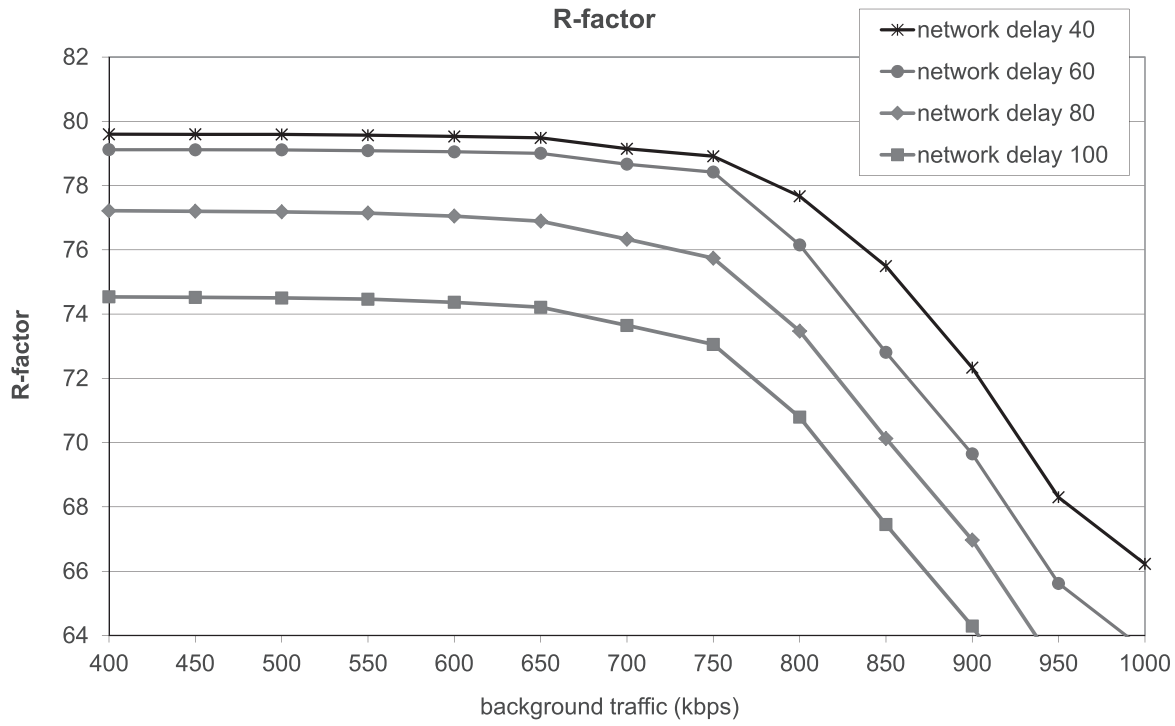

Fig. 27. *R*-factor with 10 multiplexed flows and different network delays, for *time-limited* buffer.

1600 kbps of background traffic: the best behavior is obtained using $l = 2$ tunnels of $k = 20$ multiplexed flows, which is very similar to $l = 4$ tunnels of $k = 10$ flows.

On one hand, multiplexing all the flows into a single tunnel ($k = 40$), will save more bandwidth than any other solution, although the difference is not very significant: passing from 2×20 to 1×40 only saves 10 kpbs (about 2% of the bandwidth), but it makes the packets almost double their size. As we saw in the analytical section, bandwidth savings have an asymptote, so above 20 flows the gain is negligible. But, on the other hand, packets will be bigger, so the probability of being discarded by the buffer is increased. So if the buffer penalizes big packets, as it occurs in this case, it will be more interesting to group the

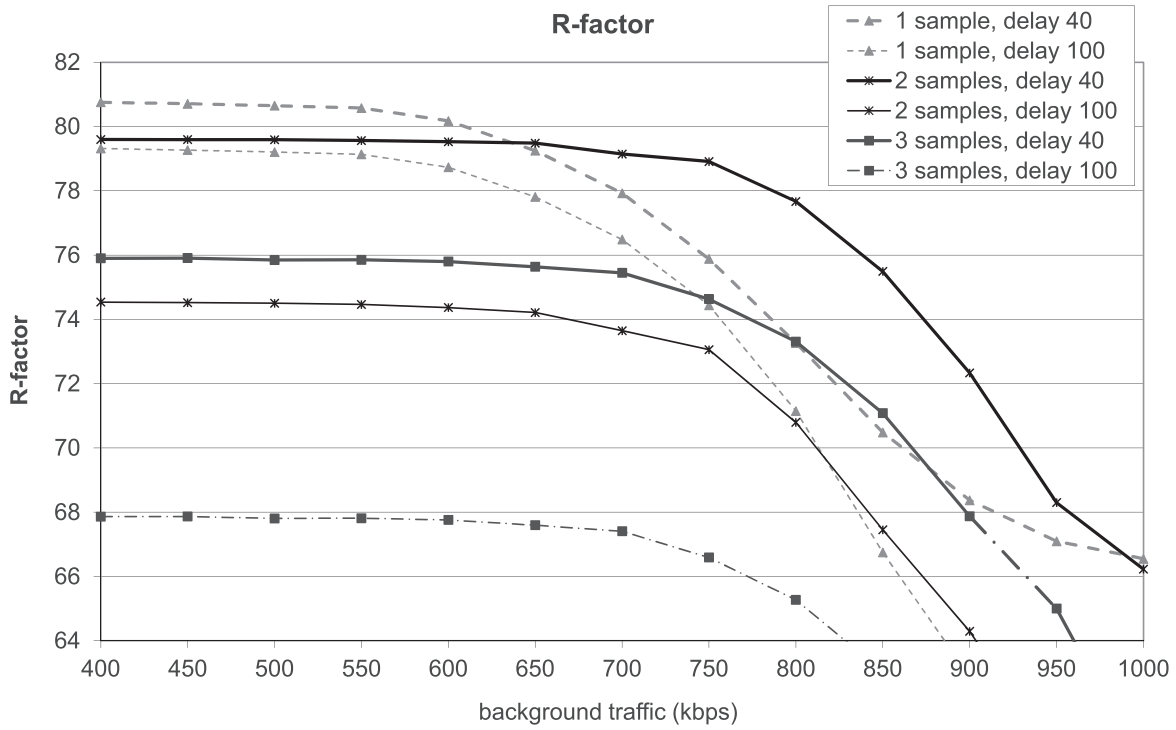

Fig. 28. *R*-factor 10 multiplexed flows, different network delays and different number of samples per packet, for *time-limited* buffer.

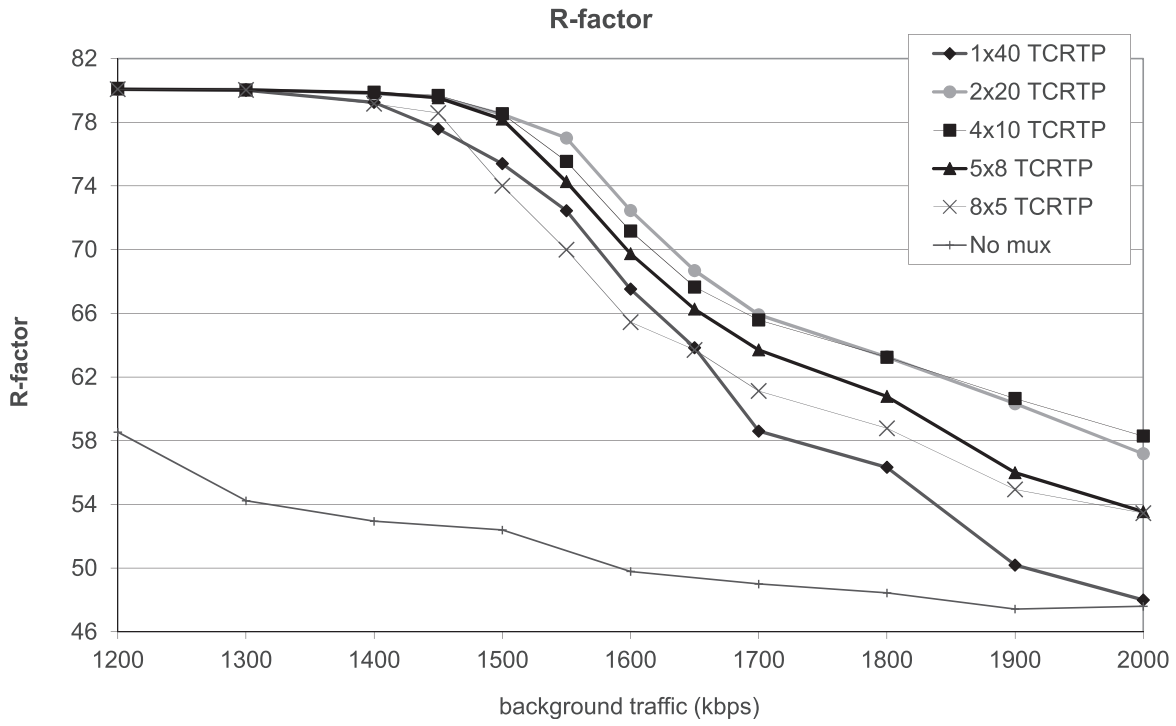

Fig. 29. *R*-factor for different values of *k* with *time-limited* buffer of 80 ms.

Table 4
Values of *R* for 40 flows multiplexed with different values of *k*.

$l \times k$	1×40	2×20	4×10	5×8	8×5	No mux
BG. 1500 kbps	72.44	77.01	75.54	74.26	69.99	52.39
BG. 1600 kbps	67.51	72.46	71.16	69.74	65.44	49.78

flows into a number of tunnels, which will generate smaller packets.

Fig. 30 shows the percentage of *R*-factor improvement that can be obtained by multiplexing, with respect to the values obtained for native RTP. This fig. illustrates the fact that multiplexing all the calls into a single tunnel is not al-

ways the best solution. The best results are obtained for 2×20 and 4×10. There is another fact which can be highlighted: although the curve of 8×5 is the first one that goes down, it does with a smaller slope than the other ones, as it uses smaller packets. So it achieves better results than 1×40 above 1700 kbps of background traffic.

It may also be noticed that the use of big values of k helps to save background traffic from being discarded. As it can be seen in Fig. 31, the bigger the number of multiplexed calls, the smaller the packet loss percentage for background traffic. This is happening because multiplexing all the calls into a single tunnel is the option that achieves the highest bandwidth saving, although it produces the biggest packets. On the other side, if we use 8 tunnels of 5 calls, we will generate smaller packets, but we will save less bandwidth.

To sum up, we see that there is a tradeoff: if we want to prioritize voice traffic, we will have to use the value of k that maximizes R-factor. But if we simply want to save bandwidth, we should multiplex all the calls into a single tunnel, achieving the best performance for background traffic.

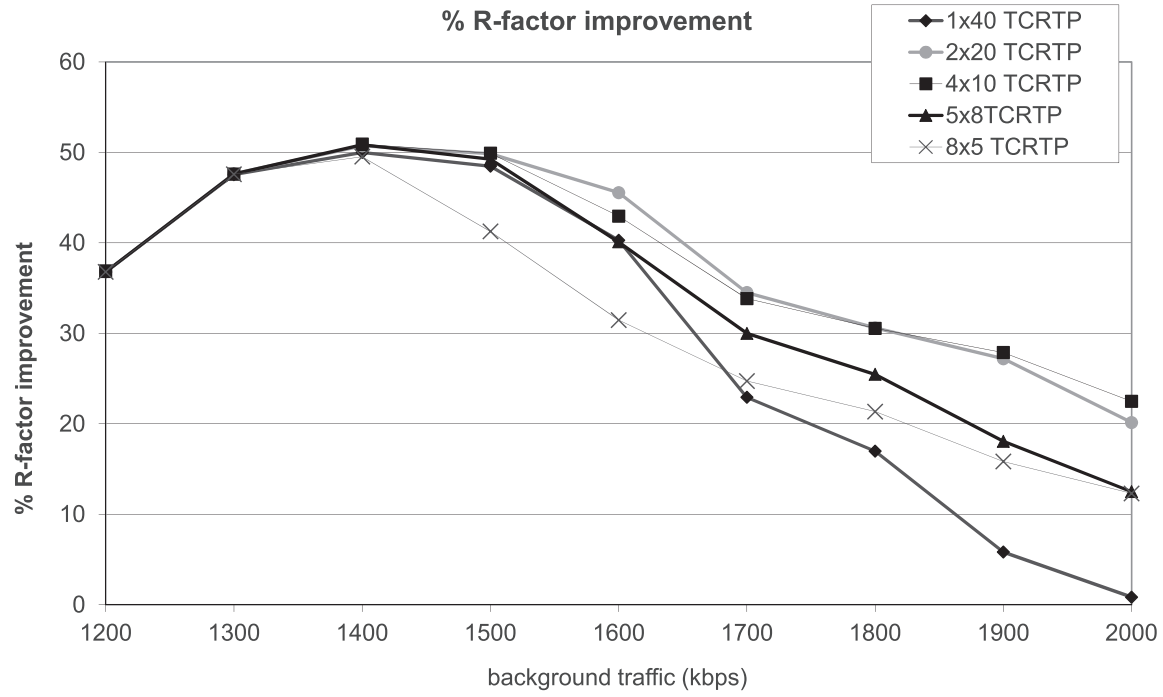

Fig. 30. Percentage of R-factor improvement with respect to 40 RTP flows for *time-limited* buffer.

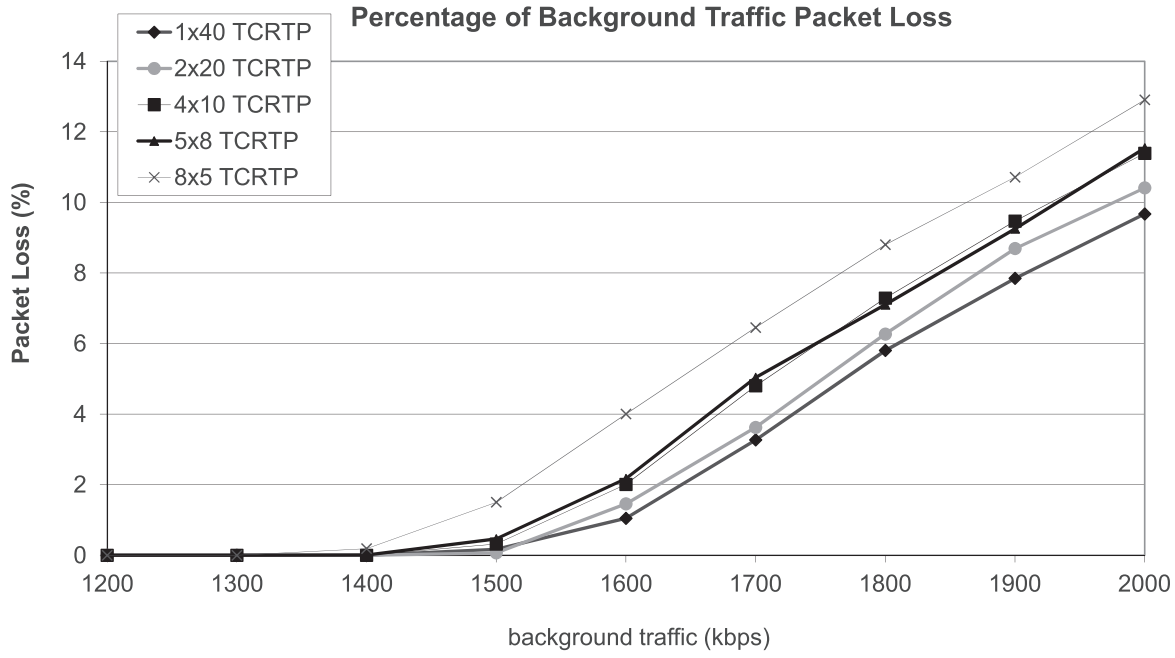

Fig. 31. Percentage of background packet loss with different values of k for *time-limited* buffer.

6. Discussion of the results

In this section we will analyze the results, trying to extract some conclusions about the best way to use RTP multiplexing in real scenarios.

As it has been shown in previous sections, there are some first decisions we can take in order to set the main parameters of our VoIP system, i.e. the number of samples per packet and whether multiplexing or not. These decisions will modify the total bandwidth, the packet size and pps of our VoIP traffic.

Native RTP always uses the same packet size, and the bandwidth increases linearly with the number of flows. The bandwidth used by TCRTP is also linear with respect to k (see Eq. (2)), but packet size grows with the number of flows. As we have seen in the results section, the packet size growth will not always be beneficial for the perceived quality, as the delays will grow, and packet loss may be increased depending on the implementation of the buffer. So there is a tradeoff, which has to be solved taking into account the perceived quality, i.e. R-factor.

6.1. R-factor depending on the number of flows

As we increase the number of multiplexed flows, we see that we can save more bandwidth, but we have to take into account the asymptotic behavior of the multiplexing gain, because above 15 or 20 flows the gain is negligible.

If the buffer has a *fixed number* of packets, the use of TCRTP makes it able to store a bigger number of bytes, as TCRTP packets are bigger. This is beneficial in order to avoid packet loss. Although the delay is slightly increased when using bigger packets, what really harms R-factor is the high packet loss percentage.

If a *high capacity* buffer is used, then the behavior of the system is simple: it works properly until the bandwidth limit is reached. So in this case, multiplexing is always interesting because of bandwidth saving.

But if a *time-limited* buffer is used, the behavior changes, as we have seen. Packet size has to be taken into account, as VoIP packets have to compete with background ones. Fig. 32 compares the maximum background traffic that can be tolerated in order to have a concrete value of R. We have depicted the graphs for $R = 70$, which is the normally accepted limit, and also for $R = 65$ and $R = 75$, using two samples per packet.

We see that there is no multiplexing gain for $k = 5$ and 10 (e.g. for $k = 10$ the result is almost the same for $R = 70$), but with $k = 15$ and 20, there is an interesting improvement.

We have also seen that packet loss for background traffic increases with the bandwidth used by VoIP flows. So the best choice in order to maximize R-factor may not be the one which minimizes packet loss for background traffic.

6.2. Distribution of the flows

In the previous section we have studied the influence of the distribution of a fixed number of RTP flows on the perceived quality. We have not studied this topic for the buffer with a *fixed number* of packets. As the results have shown, the increase of packet size is always beneficial, so the best choice will be grouping all the flows into a single tunnel. This also happens for *high capacity* buffer. In the case of *time-limited* buffer, the use of a single tunnel will not necessarily be the best solution: in some cases, better values of R-factor have been obtained by the division of the total number of flows into different tunnels.

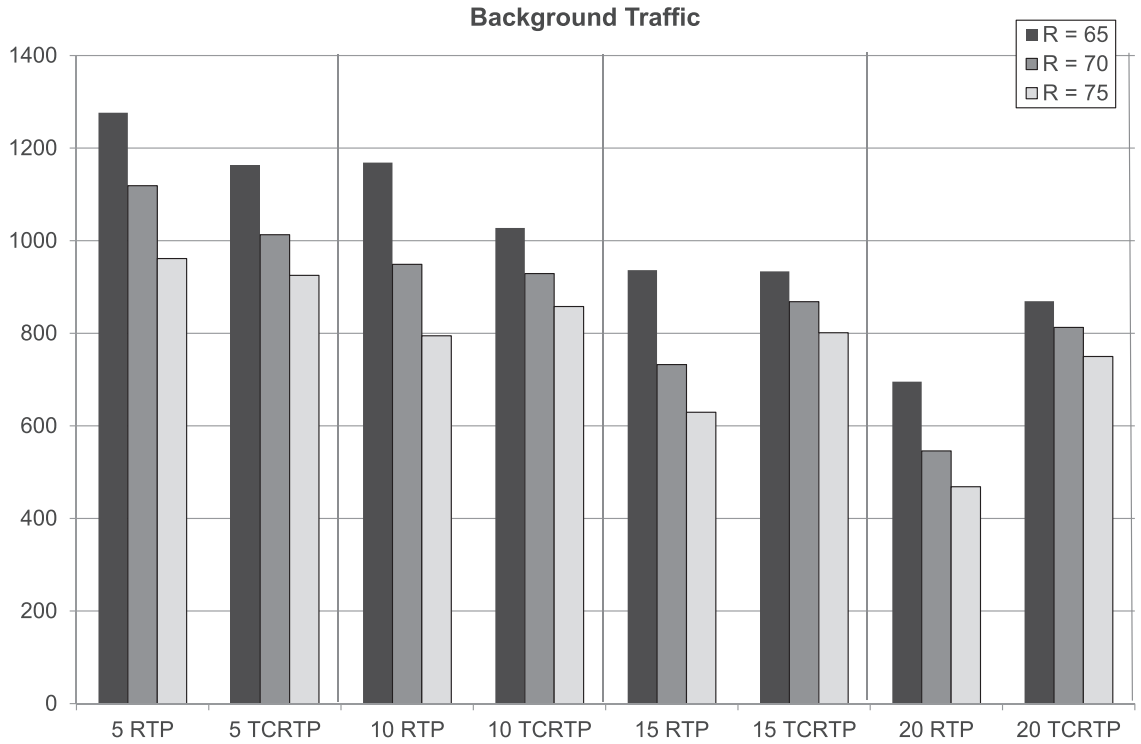

Fig. 32. Maximum background traffic for $R = 65$, 70 and 75, for *time-limited* buffer.

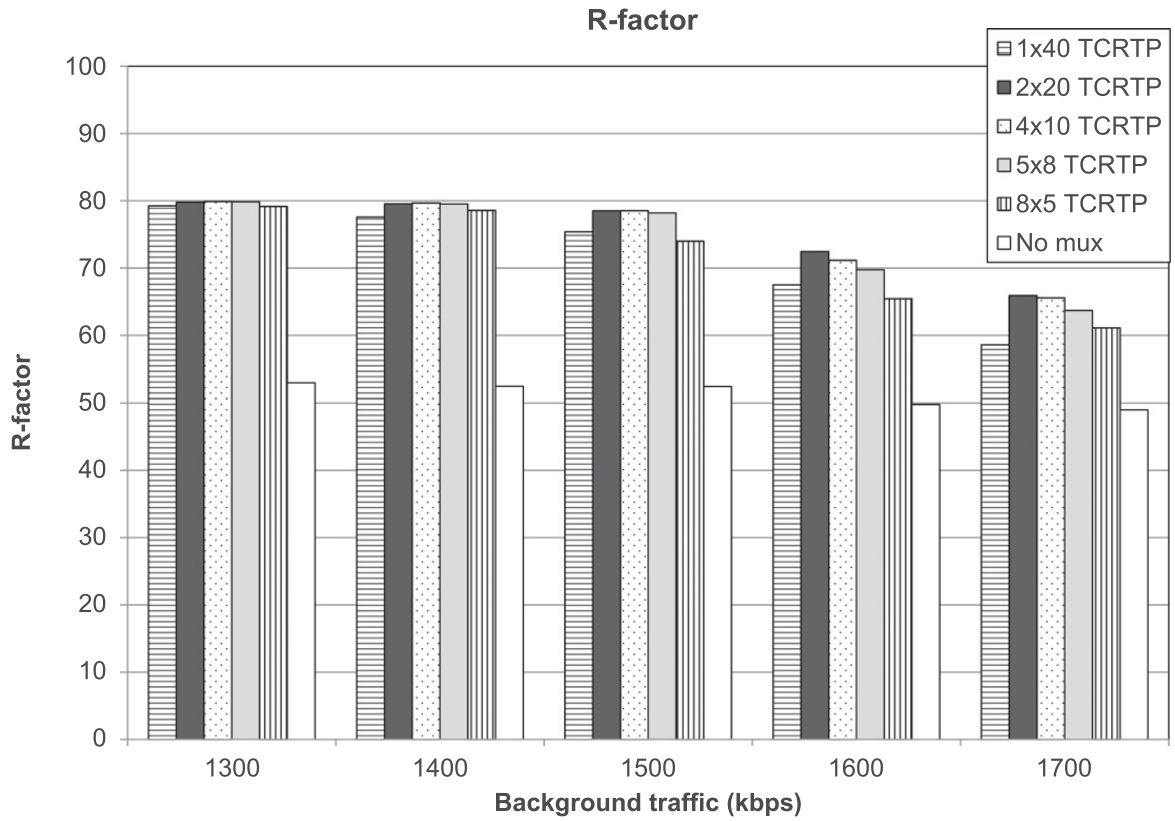

Fig. 33. *R*-factor for fixed background traffic with different values for *l*, for $k = 40$, for *time-limited* buffer.

Fig. 33 shows *R*-factor for fixed values of background traffic, for different possible distributions of $k = 40$ flows. As it can be seen, the values obtained for 2×20, 4×10 and 5×8 distributions are better than the ones obtained for a single tunnel. The reason is the small difference in terms of bandwidth, due to the proximity of the asymptote, and the big difference in terms of packet size.

We can also observe that by multiplexing we can obtain better *R*-factor values with respect to native RTP, mainly due to bandwidth saving. So we will have to take a decision depending on the behavior of our router and the bandwidth of our access network.

7. Conclusions

This work studies RTP multiplexing from the perspective of perceived quality, taking into account that the behavior of the router buffer has a big influence on it. The RTP multiplexing option selected is IETF's TCRTP scheme. First, an analytical study of this scheme in terms of packet size, bandwidth and packets per second has been presented.

Next, different multiplexing schemes have been tested and compared with native RTP carrying VoIP samples, as it is a significant and widely used real-time service. Some tests using ITU *R*-factor have been carried out in order to compare the effect of different buffer implementations. On one hand, the use of RTP multiplexing requires less bandwidth but, on the other hand, it introduces new delays, i.e. retention time and also small processing times in both sides of the communication. It also modifies packet size, as multiplexed packets are bigger than RTP ones, and this may increase their probability of being discarded, depending on the behavior of the buffer.

If the buffer is designed in order to store a *fixed number* of packets, multiplexing shows an advantage with respect to native RTP, as it avoids the discarding of a high percentage of the packets.

When using high capacity buffer, *R*-factor shows a simple behavior: it is good until bandwidth limit is reached. The same behavior can be observed for a dedicated bandwidth for VoIP traffic. But using a time-limited buffer, *R*-factor can be acceptable even if the total traffic amount is above bandwidth limit, as voice packets have the advantage of their small size with respect to background ones.

It has been found that in certain conditions multiplexed RTP can obtain better results than native RTP. The use of a tunnel in case of TCRTP does not harm conversation quality, although it implies some bandwidth cost, and it also adds some delays.

The number of samples per packet has also been studied, and it has been found that the election of this parameter may affect the experienced quality. If the delays of the network are big, we will be forced to use a small number of samples per packet.

We have studied the influence of the distribution of the number of flows into different numbers of TCRTP tunnels, and the conclusion is that grouping all the flows into a single tunnel will not always be the best solution, as the increase of the number of flows does not improve bandwidth efficiency indefinitely. If the buffer penalizes big packets, it will be better to group the flows into a num-

ber of tunnels. Finally, the router processing capacity has to be taken into account, as the limit of packets per second it can manage must not be exceeded.

The buffer implementation and the traffic we want to prioritize will be the parameters used to decide the best distribution of the RTP flows in a different number of tunnels. The number of pps has to be taken into account too, in order to avoid the discarding of packets by the router.

The obtained results show that multiplexing is a good way to improve customer experience of VoIP in scenarios where many RTP flows share the same path, but we have to know the behavior of the router in order to take the correct decision.

Acknowledgment

This work has been partially financed by CPUFLIPI Project (MICINN TIN2010-17298), MBACToIP Project, of Aragon I+D Agency and Ibercaja Obra Social, and NDCIPI-QQoE Project of the Catedra Telefonica of the Univ. of Zaragoza.

References

[1] J. Saldana, J. Murillo, J. Fernández-Navajas, J. Ruiz-Mas, E. Viruete, J. I. Aznar, Evaluation of Multiplexing and Buffer Policies Influence on VoIP Conversation Quality, Consumer Communications and Networking Conference (CCNC), 2011 IEEE, Las Vegas, 9–12 Jan 2011, pp. 378–382.
[2] J. Saldana, J. Murillo, J. Fernández-Navajas, J. Ruiz-Mas, E. Viruete, J. I. Aznar, Influence of the Distribution of TCRTP Multiplexed Flows on VoIP Conversation Quality, Consumer Communications and Networking Conference (CCNC), 2011 IEEE, Las Vegas, 9–12 Jan 2011, pp. 689–690.
[3] C. Perkins, Rtp: Audio and Video for the Internet, Addison-Wesley Professional, 2003.
[4] The E-model, a computational model for use in transmission planning, ITU-T Recommendation G.107, March 2003.
[5] V. Jacobson, RFC 1144: Compressing TCP/IP Headers for Low-Speed Serial Links, February 1990.
[6] M. Degermark, B. Nordgren, D. Pink, RFC 2507: IP Header Compression, February 1999.
[7] S. Casner, et al., RFC 2508: Compressing IP/UDP/RTP Headers for Low-Speed Serial Links, February 1999.
[8] T. Koren, et al., RFC 3545: Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering, July 2003.
[9] C. Bormann (Ed.), RFC 3095: Robust Header Compression (ROHC), June 2001.
[10] E. Ertekin, C. Christou, Internet protocol header compression, robust header compression, and their applicability in the global information grid, IEEE Communications Magazine 42 (2004) 106–116.
[11] B. Thompson, T. Koren, D. Wing. RFC 4170: "Tunneling Multiplexed Compressed RTP (TCRTP)", November 2005.
[12] H.P. Sze, S.C. Liew, J.Y.B. Lee, D.C.S. Yip. A Multiplexing Scheme for H.323 Voice-Over-IP Applications, IEEE J. Select. Areas Commun, Vol. 20, September 2002, pp. 1360–1368.
[13] T. Hoshi, K. Tanigawa, K. Tsukada, Proposal of a method of voice stream multiplexing for IP telephony systems, in Proc. IWS '99, February 1999, pp. 182–188.
[14] A. Trad, H. Afifi, Adaptive Multiplexing Scheme for Voice Flow Transmission Across Best-Effort IP Networks, INRIA Research, Report 4929, September 2003.
[15] B. Subbiah, S. Sengodan, draft-ietf-avt-mux-rtp-00.txt, User Multiplexing in RTP payload between IP, Telephony Gateways, August 1998.
[16] J. Yu, I. Al-Ajarmeh, Call Admission Control and Traffic Engineering of VoIP, in: Proc. Second International Conference on Digital Telecommunications, IEEE ICDT, 2007.
[17] W. Feng, F. Chang, W. Feng, J. Walpole, A Traffic Characterization of Popular On-Line Games, IEEE/ACM Trans. Netw., 2005, pp. 488–500.
[18] R. M. Pereira, L.M. Tarouco: Adaptive Multiplexing Based on E-model for Reducing Network Overhead in Voice over IP Security Ensuring Conversation Quality, in: Proc. Fourth international Conference on Digital Telecommunications, Washington, DC, pp. 53–58, July 2009.
[19] K. Pentikousis, E. Piri, J. Pinola, F. Fitzek, T. Nissilä, I. Harjula, Empirical evaluation of VoIP aggregation over a fixed WiMAX testbed, in: Proc. 4th international Conference on Testbeds and Research infrastructures For the Development of Networks & Communities. Innsbruck, Austria, March 2008.
[20] A. Vishwanath, V. Sivaraman, M. Thottan, Perspectives on router buffer sizing: recent results and open problems, SIGCOMM Comput. Commun. Rev. 39 (2) (2009) 34–39.
[21] C. Villamizar, C. Song, High performance TCP in ANSNET, ACM Computer Communication Review, October 1994.
[22] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers, in: SIGCOMM '04, ACM Press, New York, USA, 2004, pp. 281–292.
[23] A. Dhamdhere, C. Dovrolis, Open issues in router buffer sizing, Comput. Commun. Rev. 36 (1) (2006) 87–92.
[24] One-way transmission time, ITU-T recommendation G.114. February 1996.
[25] G. Dimitriadis, S. Karapantazis, F.-N. Pavlidou, Comparison of header compression schemes over satellite links, in: Proc. International Workshop on IP Networking over Next-generation Satellite Systems (INNSS'07), Budapest, Hungary, July 2007.
[26] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RFC 3550: RTP: A Transport Protocol for Real-time Applications, July 2003.
[27] Cooperative Association for Internet Data Analysis "NASA Ames Internet Exchange Packet Length Distributions".
[28] J. Saldaña, E. Viruete, J. Fernández-Navajas, J. Ruiz-Mas, J. I. Aznar, Hybrid Testbed for Network Scenarios, SIMUTools 2010, the Third International Conference on Simulation Tools and Techniques, Torremolinos, Spain, March 2010.
[29] E. Göktürk, A stance on emulation and testbeds, in: Proc. 21st European Conference on Modelling and Simulation ECMS, 2007.
[30] A. Botta, A. Dainotti, A. Pescapè, Multi-protocol and multi-platform traffic generation and measurement, INFOCOM 2007 DEMO Session, Anchorage, Alaska, USA, May 2007.
[31] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson, R. Steinmetz, Modelling the internet delay space based on geographical locations, in: 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009), February 2009.
[32] The PingER Project, <http://www-iepm.slac.stanford.edu/pinger>.
[33] AT&T Global IP Network, <http://ipnetwork.bgtmo.ip.att.net/pws/>.
[34] R.G. Cole, J.H. Rosenbluth, Voice over IP performance monitoring, SIGCOMM Comput. Commun. Rev. 31 (2) (2001) 9–24.

Jose Saldana received his B.S. and M.S. in Telecommunications Engineering from University of Zaragoza, in 1998 and 2008, respectively. He received his PhD in Information Technologies in 2011. He is currently a research fellow in the Department of Engineering and Communications of the same University. His research interests focus on Quality of Service in Real-time Multimedia Services, as VoIP and networked online games.

Julián Fernández-Navajas received the Telecommunications Engineering degree from the Polytechnic University of Valencia, in 1993, and the Ph.D. degree from the University of Zaragoza, Spain, in 2000. He is currently an Associate Professor in the Centro Politécnico Superior, Universidad de Zaragoza. His professional research interests are in Quality of Service (QoS), Network Management, Telephony over IP, Mobile Networks, online gaming and other related topics.

José Ruiz-Mas received the Engineering of Telecommunications degree from the Universitat Politècnica de Catalunya (UPC), Spain, in 1991 and the Ph.D. degree from the University of Zaragoza in 2001. He worked as a software engineer at the company TAO Open Systems from 1992 to 1994. In 1994 he joined the Centro Politécnico Superior as an Assistant Professor until 2003, when he became an Associate Professor. At present he is member of the Aragón Institute of Engineering Research (I3A) and his research activity lies in the area of Quality of Service in Multimedia Services with special emphasis on the provision of methodologies and tools to assess the perception of the end-user (Quality of Experience, QoE).

Jenifer Murillo received her B.S. in Telecommunications Engineering from University of Zaragoza in 2010. She is currently working as a researcher at the same University, and she is interested in issues related to IP telephony and Quality of Service.

Eduardo Viruete Navarro received his B.S. and M.S. in Telecommunications Engineering from University of Zaragoza, in 2003 and 2005, respectively. He is currently pursuing the Ph. D. degree in the Department of Electronic Engineering and Communications of the same University. His research interests focus on Quality of Service in Multimedia Services and Ambient Intelligence.

José Ignacio Aznar received his B.S. Telecommunications Engineering in 2008 and M.S. in 2010 from University of Zaragoza. His main research interests have been focused on the optimization and enhancement of IMS-based architectures and currently, he is involved in several research lines, related to estimation, control and monitoring of Quality of Service (QoS) parameters.