Online Games Traffic Multiplexing: Analysis and Effect in Access Networks

Jose Saldana, Julián Fernández-Navajas, José Ruiz-Mas and Luis Casadesus
Communications Technologies Group (GTC) – Aragon Inst. of Engineering Research (I3A)
Dpt. IEC. Ada Byron Building. EINA Univ. Zaragoza
50018 Zaragoza, Spain
[e-mail: {jsaldana, navajas, jruiz, luis.casadesus}@unizar.es]
*Corresponding author: Jose Saldana

Received August 13, 2012; revised October 24, 2012; revised November 13, 2012; accepted November 13, 2012;published November 30, 2012

Abstract

Enterprises that develop online games have to deploy supporting infrastructures, including hardware and bandwidth resources, in order to provide a good service to users. First Person Shooter games generate high rates of small UDP packets from the client to the server, so the overhead is significant. This work analyzes a method that saves bandwidth, by the addition of a local agent which queues packets, compresses headers and uses a tunnel to send a number of packets within a multiplexed packet. The behavior of the system has been studied, showing that significant bandwidth savings can be achieved. For certain titles, up to 38% of the bandwidth can be saved for IPv4. This percentage increases to 54% for IPv6, as this protocol has a bigger overhead. The cost of these bandwidth savings is the addition of a new delay, which has an upper bound that can be modified. So there is a tradeoff: the greater the added delays, the greater the bandwidth savings. Significant reductions in the amounts of packets per second generated can also be obtained. Tests have been deployed in an emulated scenario matching an access network, showing that if the number of players is big enough, the added delays can be acceptable in terms of user experience.

Keywords: gaming, multiplexing, compressing, network games, quality of experience, first person shooter

A preliminary version of this work was published in "Bandwidth Efficiency Improvement for Online Games by the use of Tunneling, Compressing and Multiplexing Techniques," Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems SPECTS 2011, pp.227-234, The Hague, Netherlands, June 2011. Some other parts were published in the article "Influence of the Router Buffer on Online Games Traffic Multiplexing," presented at the same conference, pp.253-258. The study has been extended to another FPS game. The study of the saving in packets per second has been expanded through analytical calculations and inclusion in the comparisons between analytical and simulation results. The exponential distribution has been added and studied. This work has been partially financed by the CPUFLIPI Project (MICINN TIN2010-17298), the European Social Fund, the MBACToIP Project, of the Aragon I+D Agency and Ibercaja Obra Social, and the NDCIPI-QQoE Project of the Catedra Telefonica, Univ. of Zaragoza.

http://dx.doi.org/10.3837/tiis.2012.10.010

1. Introduction

In recent years, online gaming via Internet has become more and more popular. Some titles have millions of users [1] so the enterprises that develop these games have to face a difficult problem whenever a new title is released: they need hardware and bandwidth resources in order to avoid saturation of their infrastructure. As the success of a new title is not easily predicted, they may have to over-provision these resources to ensure that users who buy the game receive a good service [2]. In [3] a study of the behavior of *gamers* was presented. It was shown that they are very difficult to satisfy. If they have connection problems, they usually leave and never return, and they tend not to be loyal to a specific server.

Two of the most popular genres of online games are MMORPGs (Massively Multiplayer Online Role Playing Games) and FPSs (First Person Shooters). Reference [4] studied the traffic of MMORPGs, concluding that they share some characteristics such as periodicity, locality, and self-similarity. Another conclusion is that they have less bandwidth and real-time requirements than FPSs.

In FPSs, the actions of the players have to be propagated to the server and to the rest of the players in a very short time, so network delays are very critical. These games produce a high rate of small UDP packets (some tens of bytes) from the client to the server, so the overhead caused by IP/UDP headers is significant. Server-to-client packets are typically bigger.

The design of these games takes into account that the access network is usually the narrowest bottleneck. In fact, in [5] it was shown that some titles were designed "to saturate the narrowest last-mile link," taking into account the maximum available bandwidth in these networks at the moment the game was released. Most access technologies, as in the case with cable or DSL, present an asymmetrical bandwidth, and the uplink is normally the most stringent one.

In order to reduce the workload on the central game server, and to ensure a provision of a good service, some studies [6], [7] have proposed the inclusion of network elements (proxies) next to the access network, which could deploy different tasks, and thus alleviate the central server processing workload. On the other hand, techniques for multiplexing packets and compressing headers [8] have also been defined and extensively used so as to reduce overhead in other real-time services, such as VoIP.

Merging these two ideas, we can think about a proxy or a local agent which queues packets, compresses headers and sends multiple packets into a bigger packet, at the cost of adding new delays, mainly caused by the retention time in the queue. Two main benefits can be obtained: a reduction of the number of packets per second the router has to manage, and bandwidth savings given that small packets have a significant overhead. This proxy could be distributed with the game application in the same way as local servers are distributed with certain titles.

Regarding the possible scenarios where this technique can be applied, network aggregation points where a number of flows is present are the most interesting. A small multiplexing delay may be enough in order to merge a large number of packets, thus obtaining significant bandwidth reduction. The first scenario could be the infrastructure of a game provider (**Fig. 1a**), where proxies can be used in order to transfer workload to network borders. The same technique could be used in LAN parties, where large numbers of players share the same path. On the other hand, Internet cafés, which have given many users the opportunity of accessing Internet services since the middle of the 1990s, are also an interesting scenario. Nowadays, they still represent an important means of connection for users in some countries [9]. The

profile of their users has been studied [10], and gaming has been reported as an important activity. Internet cafés are present all over the world, but they have a special significance in developing countries [11]. This scenario, where many computers share the same Internet connection, is subject to substantial variability: different network technologies depend on the telecommunications infrastructure of the country in question, different routing equipment and network topologies, etc. Bandwidth is considered a scarce resource, which has to be well administrated. It is common for a group of people go to a café to play a game, so the traffic of these groups of players could be compressed and multiplexed in order to save bandwidth, taking into account that the access network is usually the narrowest bottleneck. The local agent can be placed in different locations in the scenario: it can be included in a local machine (**Fig. 1b**) or into the computer of one of the players (**Fig. 1c**). It could even be embedded in the router (**Fig. 1d**). Thus, it could be able to measure the current traffic distribution of the access network and to use that information to properly tune multiplexing parameters.

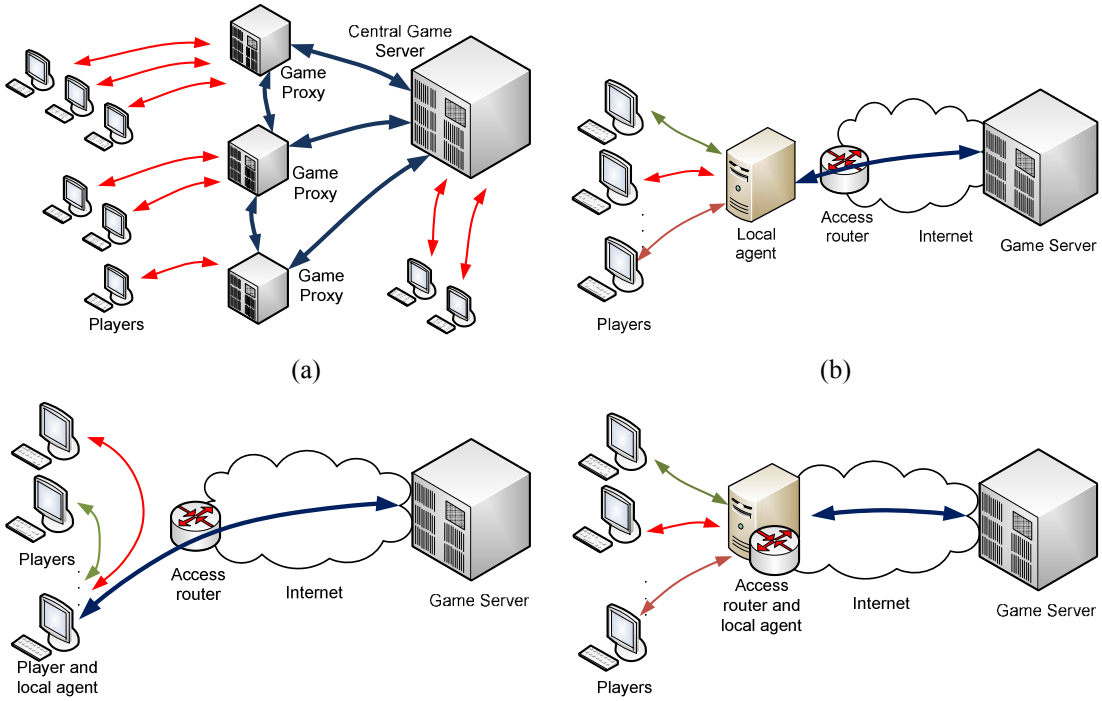


Fig. 1. Scenarios where many players share the same path: a) traffic between proxy servers of the same game; b) players sharing an access network, using a local agent; c) players using the computer of another player in order to create the tunnel; d) local agent embedded in the router. Thick lines represent the traffic of a number of players

At the other end of the communication, the server would have to implement the demultiplexer and decompressor, which would imply some processing capacity, and some space for the *context* of each flow, i.e. the information necessary to rebuild the compressed headers, (some tens of bytes, as we will see [12]). This does not involve a scalability problem, as the server already stores the state of the game for each player. On the other hand, the savings in terms of bandwidth and packets per second may be beneficial for the server.

If the number of players is sufficiently large, at the expense of adding small delays, a large

number of packets could be multiplexed into a larger packet. Bandwidth saving not only affects the gaming traffic, but it can also be beneficial for the background traffic which shares the access with it. Furthermore, multiplexing has another advantage: the number of packets per second the router has to manage will be reduced. There are other applications and scenarios where many real-time flows share the same path, e.g. VoIP trunking, and the use of multiplexing and compressing techniques has been proposed and even standardized [8] for these. Many online games have similar traffic patterns, generating a high rate of tiny packets, and thus presenting a big overhead.

Only client-to-server traffic will be considered here, as bigger savings can be obtained and in many scenarios (e.g. DSL) this traffic is transmitted via the most restrictive link. We have to measure the network impairments that determine the quality experienced by users, in order to properly tune the parameters that define the tradeoff between bandwidth saving and quality.

Although this technique could be applied to other game genres, we will consider FPS traffic in this work as FPS games have very stringent real-time requirements. The subjective quality mainly depends on delay and packet loss [13]. The System Response Time (SRT: the time the system needs to detect a user event, to process it and to send the updated game state to the local output device) has to be maintained under a certain limit.

A technique named TCM (Tunneling, Compressing and Multiplexing) was presented in [14] and tested by means of simulation. This technique compresses headers, multiplexes packets from different flows and sends them using a tunnel. The traffic of eight FPS games was generated using real traces or synthetic models, and the bandwidth savings were presented. By the addition of small delays, it is possible to obtain bandwidth savings of 30% for client-to-server traffic of many games, and up to 50% for some others. Another effect of the technique is that packet size increases depending on the number of merged packets.

The current paper presents a theoretical analysis of the savings and compares this with results obtained using simulation. Taking into account that many scenarios are access networks, a further step has been taken. The study has been extended not only to bandwidth savings, but also to the way these savings can be translated into QoS parameter improvements. The traffic has been sent in an emulated access network scenario, using different buffer policies in the access router. A big buffer is tested, and also a time-limited one, showing its capability for limiting the latency. The results in terms of delay and packet loss for the FPS traffic and also for background traffic are presented and analyzed.

The rest of the work is organized as follows. The following section describes related work. Section 3 explains and analyzes the tunneling, compressing and multiplexing method. Section 4 details the results. The paper ends with the conclusions.

2. Related Work

Various related topics are considered in this study. First, we consider online gaming traffic. We then study the problem of the access router. The supporting infrastructures of online games are subsequently summarized and, finally, different optimization algorithms are described.

2.1 Online Gaming Traffic

There is a significant amount of literature regarding the traffic of online games. We only consider active traffic, i.e. traffic generated once the game has started. This traffic has two different characteristics. First, the client application is responisble for communicating the actions of the players to the server, using small packets with a small period. Second, the server

calculates the new state of the game and broadcasts it to all the players, using bigger packets whose size depends on the number of players. Ref. [15] presented a method to extrapolate server to client traffic, obtained from empirical measurements. Size distributions were obtained for an N-player game from the measured traffic of 2 and 3 players. The client to server size distribution was found to be independent of the number of players.

In [5] a 500 million packet trace of *Counter Strike* was analyzed. It was concluded that the game is designed to saturate the bottleneck, which is the last-mile link. In [16] the characteristics of many online games were analyzed in terms of packet size and inter-packet time. In [17] a survey of different traffic models for 17 popular games can be found. These studies show that such games generate a high rate of small packets. This produces a big overhead, so bandwidth savings can be achieved by means of header compression and multiplexing. In [5] it is also said that the main bottleneck is frequently the number of packets per second that a router can manage, and not only the bandwidth of the access line. The reason for this is that routers are usually designed for big packets, and can experience problems when managing bursts of small ones.

2.2 Access Networks and Size of the Router Buffer

The scenario we are considering may have very different technologies, and the access router is no exception: there is a big variety of them. The problem of sizing the buffer of a router has been studied in depth. In the review presented in [18], Dhamdere and Drovolis explained that some years ago, the "rule of the thumb" of using the bandwidth-delay product was contested by the so-called "Stanford Model", which uses smaller buffers. In the same work, the authors also proposed a time-limited buffer which discards the packets that spend more than a certain time in the queue. This buffer penalizes big packets, but is interesting for real-time multimedia flows, as it maintains the delay under an upper bound. In this paper we compare this approach with the use of bigger buffers.

In [19] a large number of residential accesses were measured, and it was found that many of them had big queues that may add delays of hundreds of milliseconds. In the event of having traffic amounts that fill the queue, the delay is sufficient to prevent interactive applications from having an acceptable quality for the user. This can happen if peer to peer applications, which tend to saturate the access link, are used at the same time as interactive games. As a consequence, the size of the router buffer has to be studied as an important element when considering the possibility of playing interactive games.

2.3 Infrastructure for Supporting Online Games

The problem of the infrastructure that supports games has also been studied in some works. From the point of view of the user, Ref. [20] presented an algorithm to allow the client to adaptively select the best server for a certain online game. This could allow a group of users to play in the same server, and use multiplexing techniques.

From the point of view of the server, there are two architectures to support the service: centralized and distributed. In the first, there is a server that maintains the state of the game and distributes it to the players. The problem is that the server represents a bottleneck. In distributed architectures [21] there is no need for a central server, as the players exchange the information. But this architecture is generally not used in commercial games. A recent simulation study using the traffic of a popular MMORPG [22] has concluded that P2P propagation schemes are not suitable for this kind of game, taking into account the current status of access networks.

The problem of the scalability of the required infrastructure for these games was also studied by Mauve *et al* [6]. They proposed the use of proxies in order to provide robustness and congestion control, to reduce delays and to avoid cheating. Some proxies could be located close to the players, avoiding workload on the central server. Ref. [7] also proposed the use of *booster-boxes*, which would be next to the router and could be aware of the state of the network, providing network support to the applications. As said in the introduction, the solution proposed in the current work could even run in the machine of a player.

2.4 Traffic Optimization Methods

Some IETF protocols for header compression were developed many years ago. First, VJHC [23] presented a method to compress IP/TCP headers. Some years later, IPHC [12] was also able to compress UDP and IPv6 headers. At the same time, CRTP was developed in order to compress IP/UDP/RTP headers. Some years later it was enhanced and named ECRTP. However, these are not suitable for compressing gaming traffic, as games do not use RTP.

These algorithms compress headers in a hop-by-hop way, using the high redundancy of IP, TCP and UDP header fields in order to avoid the sending of some of them. A *context* is defined, which is first transmitted from the sender to the destination with the first headers. The different header fields are classified into *non-change*, *random*, *delta* and *inferred*. The first are only sent in full headers. *Random* headers are sent without compression, while *delta* are codified using fewer bytes than the original size of the field. Finally, *inferred* headers can be obtained from the fields of other layers, e.g. the length of the packet can be obtained from the corresponding level 2 field.

ROHC [24] is a more recent standard, able to compress both IP/UDP/RTP headers and IP/UDP headers. It reduces the impact of context desynchronization by providing a feedback mechanism from the decompressor to the compressor. It uses three different compression levels, and the header can be compressed to one byte [25]. The use of these techniques makes the implementation more difficult [26], and may add higher processing delays.

Real-time services, as VoIP, video conference or online gaming, have very strict delay requirements. This means that the applications generate high rates of small packets, representing a substantial overhead. Multiplexing solutions can be combined with header compression in order to optimize traffic. Different options have been proposed [8], [27], of which one was standardized [8] for scenarios where many VoIP real-time flows share the same path, as occurs in VoIP trunking. A large number of samples can be included in a single packet while only adding the retention delay corresponding to inter-packet time. Thus, the greater the number of flows the better the bandwidth efficiency. This technique is currently being adapted [14], [28] for use with the traffic of FPS games and other interactive services. An adaptation was necessary since the games do not generate RTP packets. The effect of the additional delay and jitter in terms of a subjective quality estimator, was studied in [29], showing that these techniques can be used without harming user experience.

The advantages of merging packets of interactive services have been shown in other studies. [22] concluded that message aggregation before transmission, adding a small delay, can reduce both bandwidth and global latency in both client-server and P2P schemes. In addition, [30] considered the possibility of multiplexing TCP packets of an MMORPG. These studies tested the results when multiplexing a number of packets from a single user. However, the method proposed in this paper has to be deployed in network aggregation points where a high number of flows may be present. This may allow substantial bandwidth reduction while adding small delays.

3. Compressing and Multiplexing Method

In this section we analyze the proposed compressing and multiplexing method in terms of packets per second and bandwidth efficiency. Some graphical results are included. The section ends with a description of the different delays affecting game traffic.

3.1 Tunneling, Compressing and Multiplexing Algorithm

In RFC 4170 [8], the IETF described Tunneled Compressed RTP (TCRTP) for the compression and multiplexing of RTP flows. First, ECRTP header compression is applied, and many packets are combined into one larger packet using PPPMux. An L2TP tunnel is then used in order to send the whole multiplexed packet end to end.

The proposed solution uses a similar scheme, but in this case the traffic is not RTP. We can therefore only compress IP/UDP headers using IPHC or ROHC. We will call this method *Tunnel-Compress-Multiplex* (TCM hereafter). **Fig. 2** shows the protocol stack and the structure of a TCM packet. This can be divided into the following parts:

- Common Header *(CH)*: corresponding to the IP, L2TP and the PPP headers.
- PPPMux header *(MH)*: included at the beginning of each compressed packet.
- Reduced header *(RH)*: corresponding to the IP/UDP compressed header of each original packet.
- Payload *(P)*: the UDP payload of the original packets generated by the application.

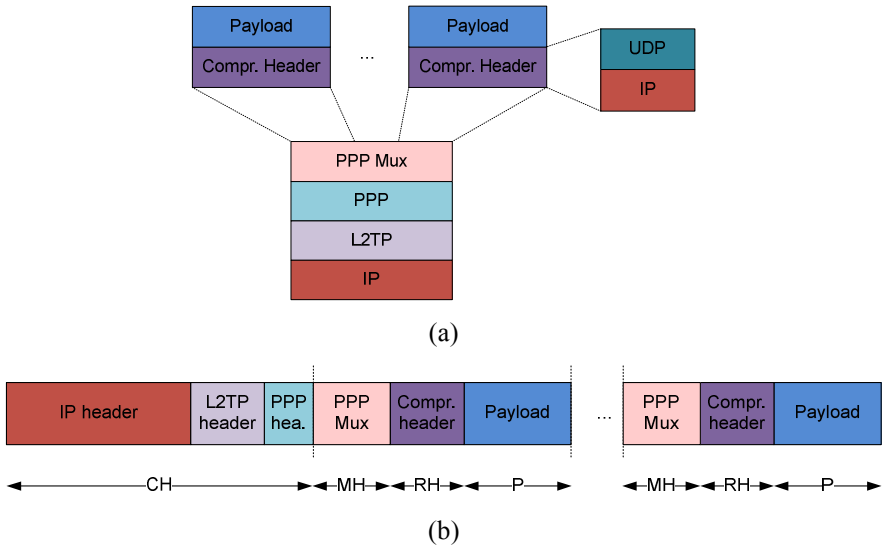


Fig. 2. TCM a) protocol stack and b) scheme of a multiplexed packet

3.2 Theoretical Analysis Of The Proposed Method

We now present an analysis of the savings which can be achieved by the use of this technique. As stated in the introduction, packet delay is of considerable importance for this service. We have therefore used a multiplexing policy that maintains packet delay under an upper bound. Two policies were compared in [31]. We will use the policy based on a period, since it adds less delay and jitter. A period, named T, is defined in the multiplexer. A packet including all the arrived packets is sent at the end of each period (**Fig. 3**). There are two exceptions: if there

is no packet to multiplex, nothing will be sent; and if there is only one packet, it will be sent in its native form, as the use of a tunnel would make it bigger.

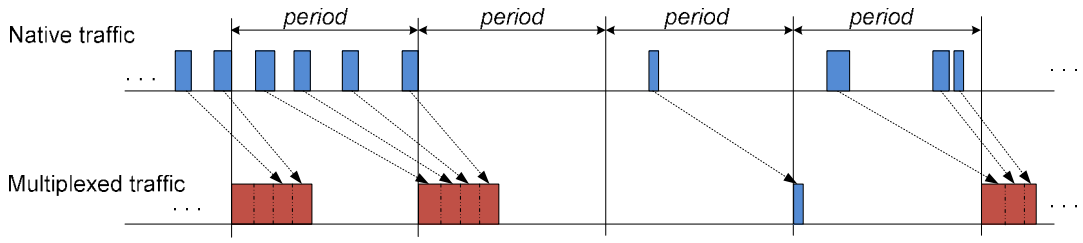


Fig. 3. Behavior of the multiplexing policy.

We will refer to the packets generated by the application as *native*, in contrast to multiplexed (*mux*) packets. With this policy, the average retention delay will be $T/2$, and its upper bound will be T.

As we have seen in the Related Work section, the reduction in terms of packets per second is an interesting advantage of multiplexing. We will first calculate the multiplexed packet rate, taking into account that a packet will be sent every period in which a native packet has arrived at the multiplexer. The expression tends to be the inverse of the period as the number of players grows:

$$pps_{mux} = \frac{\Pr(k > 0)}{T} \tag{1}$$

Next, in order to calculate the bandwidth reduction, we have first to obtain the expressions for the number of bytes sent in a period if the traffic is *native* or multiplexed. We will denote the number of arrived packets in a period as k. NH denotes the size of an IP/UDP header. The expected value of the sum of the sizes of all the $E[k]$ *native* packets arrived will be:

$$S_{native} = E[k]\ (\ NH + E\ [P]\ ) \tag{2}$$

In order to calculate the number of bytes sent in a period when multiplexing is applied, we have to distinguish the case of having one packet, in which the size will be the same as in (2), and the case of having more than one, in which the multiplexing scheme will be applied. We will have a common header CH plus the size of a number of compressed packets (MH+$E[RH]$+$E[P]$). So we can obtain an expression for the size of a multiplexed packet:

$$S_{mux} = \Pr\ (\ k{=}1\ )\ (\ NH + E\ [P]\ ) + \Pr\ (\ k{>}1\ )\ (\ CH + E[k|k{>}1]\ (\ MH + E[RH] + E[P]\ )) \tag{3}$$

In order to illustrate the bandwidth reduction, the results are presented in terms of bandwidth relationship BWR, which is the division of *mux* and *native* bandwidths. Thus, taking into account that the period is the same for *native* and *mux* packets, we can obtain BWR dividing the bandwidths (BW_{mux} and BW_{native}), and using (3) and (2):

$$BWR = \frac{BW_{mux}}{BW_{native}} = \frac{S_{mux}/T}{S_{native}/T} = \frac{S_{mux}}{S_{native}} =$$

$$= \frac{\Pr(k=1)}{E[k]} + \Pr\ (\ k{>}1\ )\ \frac{CH}{E[k](NH + E[P])} + \Pr\ (\ k{>}1\ )\ \frac{E[k \mid k > 1]}{E[k]}\ \frac{MH + E[RH] + E[P]}{NH + E[P]} \tag{4}$$

The first term is a result of the decision of not multiplexing when having only one packet.

The second term expresses how the common header is shared by the whole packet, and it becomes smaller as the number of multiplexed packets increases. The third term depends on the compressing algorithm, and on the average packet size generated by the application.

So, if we have a big number of users, or a long period, the number of multiplexed packets will be big, and the first and second terms will become negligible. Regarding the third term, $\Pr(k > 1) \approx 1$, and also $E[k|k>1] / E[k] \approx 1$. We can thus obtain the expression for an asymptote for *BWR:*

$$BWR_a = \frac{MH + E[RH] + E[P]}{NH + E[P]} \tag{5}$$

We observe that the smaller the value of $E[P]$, the smaller the value of the asymptote. The technique should provide a good performance in applications that generate a high rate of small packets, as FPS games do. Logically, it is expected that the greater the number of players, the better the performance, as the same number of packets can be multiplexed with fewer added delays. The increase of T will also be beneficial for *BWR,* but we cannot increase it indefinitely as players are very sensitive to delay.

In order to obtain some preliminary numerical results, we now use the real parameters of some commercial games and those used in the proposed protocols:

- *NH*: 28 bytes for IPv4/UDP and 48 bytes for IPv6/UDP.
- *CH*: 25 bytes for IPv4: 20 correspond to IP, 4 to L2TP and 1 to PPP header. For IPv6, *CH*=45 bytes.
- *MH*: 2 bytes, corresponding to PPPMux.
- $E[P]$*:* The value of the UDP payload depends on the application used.
- $E[k]$*:* The number of packets per second generated by the N players of the game.
- $E[RH]$*:* In this example, as a worst case scenario, we have considered IPHC compressing UDP headers to 2 bytes, by using only 8 bits for the CID field, and avoiding the optional checksum. IPv4 and IPv6 headers can also be compressed to 2 bytes. So we will consider an average of 4 bytes for compressed headers and 28 or 48 bytes for full headers, which are sent every 5 seconds (the default F_MAX_TIME parameter of IPHC [12]).

Using these values, we obtain the percentages shown in **Table 1**. These are the values of the asymptote, i.e. the best *BWR* that can be achieved if the number of users and the period are sufficiently large. They are obtained for IPv4 and IPv6. We have selected some popular games, and the specific values have been obtained from [16] and [17]. The values for *Halo 2* refer to a console with only one user [32]. The values obtained are significant. All the games allow bandwidth savings above 30% for IPv4, and this saving can increase to 54% if IPv6 is used in some titles.

Table 1. *BWR* Asymptote Values for Different Games

Game	***Engine***	***E[P]***	***BWR_a IPv4***	***BWR_a IPv6***
Unreal T 2003	Unreal 2.0	29.5	62%	46%
Quake III	Id Tech 3	36.15	65%	50%
Quake II	Id Tech 2	37	66%	51%
Counter Strike 1	GoldSrc	41.09	68%	53%
Halo 2	Halo2	43.2	69%	54%

3.3 Analysis For Different Traffic Distributions

Once we have obtained the value for the asymptote, we must calculate the values of $E[k|k>1]$, $\Pr(k=0)$, $\Pr(k=1)$ and $\Pr(k>1)$. As they may vary depending on the behavior of each game, in this subsection we will calculate them for different traffic distributions. In order to obtain an expression for $E[k|k>1]$, we can first express $E[k]$ as:

$$E[k]=\Pr(k=0)\,E[k|k=0]+\Pr(k=1)\,E[k|k=1]+\Pr(k>1)\,E[k|k>1] \tag{6}$$

And taking into account that $E[k|k=0]=0$ and $E[k|k=1]=1$, we obtain:

$$E[k|k>1]=\frac{E[k]-\Pr(k=1)}{\Pr(k>1)} \tag{7}$$

In the previous analysis, we have defined k as the total number of packets arrived at the multiplexer, i.e. the sum of the packets from each player. Now, we define l as the number of packets arrived from a single player. We consider that the N different players' packet arrivals are independent, so $E[k] = N\,E[l] = N\,\lambda\,T$.

The most common statistical distributions used in the literature for modeling inter-packet times are exponential, deterministic, normal, lognormal and extreme [32]. We will obtain the expressions for exponential and deterministic distributions. For the latter, the special case of having two possible values for inter-packet times will also be considered.

3.3.1. Exponential packet arrival

In this case, we can obtain $\Pr(k=0)$ and $\Pr(k=1)$ as:

$$\Pr(k=0)=e^{-N\lambda T} \qquad \Pr(k=1)=N\lambda T e^{-N\lambda T} \tag{8}$$

As a consequence, $\Pr(k>1)$ can be obtained as:

$$\Pr(k>1)=1-\Pr(k=0)-\Pr(k=1)=1-e^{-N\lambda T}(1+N\lambda T) \tag{9}$$

So we can use (7) to find $E[k|k>1]$:

$$E[k|k>1]=\frac{E[k]-\Pr(k=1)}{\Pr(k>1)}=\frac{N\lambda T-N\lambda T e^{-N\lambda T}}{1-e^{-N\lambda T}(1+N\lambda T)} \tag{10}$$

3.3.2. Deterministic packet arrival

In this subsection, we will consider a constant packet rate, as occurs in many games [17]. Let t be inter-packet time. We will consider $T < 2t$, in order to avoid big added delays, so the maximum value of l is 2, and consequently:

$$E[l]=\lambda T=\Pr(l=1)+2\Pr(l=2) \tag{11}$$

If we have $T \le t$, then $\Pr(l=2)=0$, so:

$$\Pr(l=1)=E[l]=\lambda T \qquad \Pr(l=0)=1-\Pr(l=1)=1-\lambda T \tag{12}$$

And if we have $T > t$, then $\Pr(l=0)=0$, so knowing that the sum of the probabilities is 1, and using (11), we obtain:

$$\Pr(l=1)=2-E[l]=2-\lambda T \qquad \Pr(l=2)=1-\Pr(l=1)=\lambda T-1 \tag{13}$$

$$\Pr(k=0)=[\Pr(l=0)]^N \tag{14}$$

If there is more than one player and $T > t$, then $\Pr(k=1)$ will be null, as every player will have sent at least one packet during the period. And if $T \le t$, then $\Pr(k=1)$ will be:

$$\Pr(k=1)|_{T\le t} = \binom{N}{1} \Pr(l=1)\,[\Pr(l=0)]^{N-1} \tag{15}$$

Once $\Pr(k=0)$ and $\Pr(k=1)$ have been obtained, $\Pr(k>1)$ can be calculated in the same way as in (9), and $E[k|k>1]$ can be obtained using (7).

3.3.3. Deterministic arrival with two possible values

In this subsection we consider the case of a game that generates packets using two different inter-packet times, as occurs in some games [17]. Let t_1 be the smallest time and t_2 the biggest, and p_1 and p_2 the respective probabilities of having t_1 and t_2. In this case, we have the following value of λ:

$$\lambda = \frac{1}{p_1 t_1 + p_2 t_2} \tag{16}$$

We will now distinguish two cases. First, if we have $T < t_1$, then we have the same case as (12), since $\Pr(l=2)=0$. In the case that $t_1 \le T < t_2$, the probability of having no packets in a period T will be the probability of the period beginning during the first t_2-T seconds of an inter-packet time of duration t_2. We have considered that $T < 2t_1$, and $t_2 < 2t_1$, and that consecutive inter-packet times are independent:

$$\Pr(l=0) = p_2 \frac{t_2 - T}{p_1 t_1 + p_2 t_2} = p_2 \lambda (t_2 - T) \tag{17}$$

And now, using (11) and knowing that the sum of the probabilities has to be 1, we are able to obtain:

$$\Pr(l=1) = \lambda [T - 2p_1(T - t_1)] \qquad \Pr(l=2) = p_1 \lambda (T - t_1) \tag{18}$$

Finally, (14) and (15) can be used to obtain the probabilities of the different values of k.

3.4 Analytical Results

In order to depict some graphs that illustrate the savings in terms of packets per second and bandwidth, a specific game has to be selected. We chose *Half Life Counter Strike 1*, due to its popularity and the availability of many studies of its behavior [5], [33]. In OpenGL mode, it presents two possible deterministic values for inter-packet time: 33 ms and 50 ms, each with the same probability. So we can calculate λ using equation (16), obtaining an average inter-packet time of 41 ms.

First, **Fig. 4a** presents the theoretical amount of packets per second, as a function of the period and the number of players. We can see that it tends to be the inverse of the period. In **Fig. 4b** we have depicted *BWR* for IPv4 as a function of the number of players and the period. If we fix the number of players, we obtain **Fig. 4c** and if we fix the value of the period, we obtain **Fig 4d**. The asymptotic behavior can be observed for both parameters, so the most interesting zone is shere the bandwidth relationship is between 0.70 and 0.75. As an example, if we look at the *20 players* graph in **Fig. 4c**, once the value 0.75 is reached, the increase in the delays in order to improve the bandwidth saving will only achieve a small benefit.

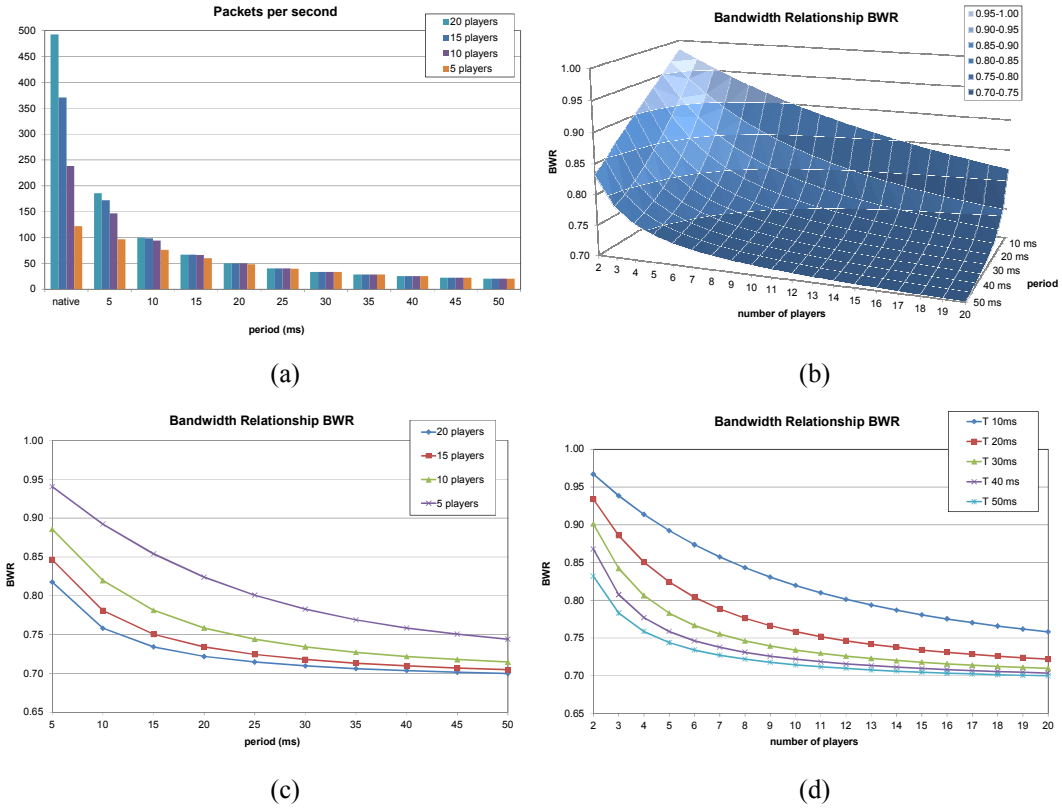


Fig. 4. a) Packets per second; b) *BWR* as a function of the number of players and the period; c) *BWR* as a function of the period; d) *BWR* as a function of the number of players.

It can also be seen that the increase in the number of players has an influence. Logically, if there are more players, the same value of $E[k]$ can be achieved using smaller values of T. So we confirm that the increase in the number of players is always beneficial. In fact, if there are only 5 players, perhaps it would be better to maintain the value of *BWR* at around 0.80. Logically, the value of the network delay will have an influence on the decision of the value of T. If the network is fast, we can add a bigger delay, thus increasing bandwidth saving.

3.5 System Delays

In this subsection we study the impact of the proposed method on the SRT, and we describe the different delays affecting the traffic of the game. In **Fig. 5** we show a scheme of the system with the delays that are added, in order to obtain SRT.

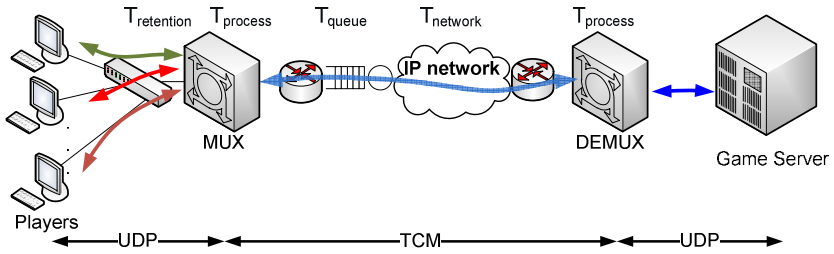


Fig. 5. Delays of the system

- $T_{retention}$ is the time a packet is retained in the queue of the multiplexer.
- $T_{process}$ represents the time spent in both the multiplexer and demultiplexer. In [27] an RTP traffic multiplexer was implemented, and the processing time was less than 1 ms.
- T_{queue} is the time spent in the queue of the access router.
- $T_{network}$ is the network delay, which is not affected.

The only significant delay which is added is $T_{retention}$ (an average of $T/2$). In [13] it is said that latency tolerance is between 150 and 180 ms for *Quake III*, and above 200 ms for *Counter Strike*, so this retention delay can be easily assimilated.

In this paper we have not considered the possibility of modifying the application. However, if this could be done, a first synchronization phase could be implemented in order to make all the computers in the same game generate the packets at the same moment. The delay could then be significantly reduced for the games that use a fixed inter-packet time.

4. Tests and Results

In this section we show some results obtained with real game traces. First, we describe the method used in order to generate the traffic for the tests. Next, a comparison is drawn between the analytical and simulation results. Finally, we describe some emulation tests deployed so as to study the influence of the router buffer.

4.1 Simulation of Traffic Multiplexing

In order to compare the simulation with the theoretical results, we use two different popular FPS games: *Half Life Counter Strike 1* in OpenGL mode, and *Quake III*. Traffic traces have been obtained from the CAIA project (e.g. the trace for 5 players for *Counter Strike 1* appears in [34]). There are available traces from 2 to 9 players. There is an offset of the first 10,000 packets, and only the next 5,000*number_of_players packets are included, to ensure that all the packets correspond to active game traffic, which is what we are studying.

Counter Strike 1 is an example of a game with two possible values for inter-packet times: we use 33 and 50 ms for the theoretical model, with a 50% probability for each one. *Quake III* can be considered as an example of deterministic inter-packet time, with a value of 11.6 ms.

In order to obtain traces for different numbers of players, we have added some of them together. For example, we have obtained a trace of 20 players by the addition of the traces of 9, 6 and 5 players in the same scenario. This can be done due to a property of client to server traffic, whose distribution is independent of the number of players [15], [33]: the client to server traffic of a 20-player game will be similar to the addition of three games of 9, 6 and 5 players. Logically, we have cut the time of the traces to the shortest one. Generation time, user identifier and the size of each packet are extracted from the real traces, and used as input for the tests.

Simulations using Matlab have been conducted, in order to obtain the compressed and multiplexed traffic traces, as illustrated in **Fig. 6**. First, the original trace is split into the individual traces of the different players, and server-to-client traffic is eliminated, since the simulations only use client-to-server traffic. Next, IP/UDP compressing is applied to each flow. Finally, using the period T, the sizes and times of the multiplexed packets are calculated. Thus, we can compare the native and multiplexed traces, in terms of bandwidth and packets per second. These results are presented in the following subsection.

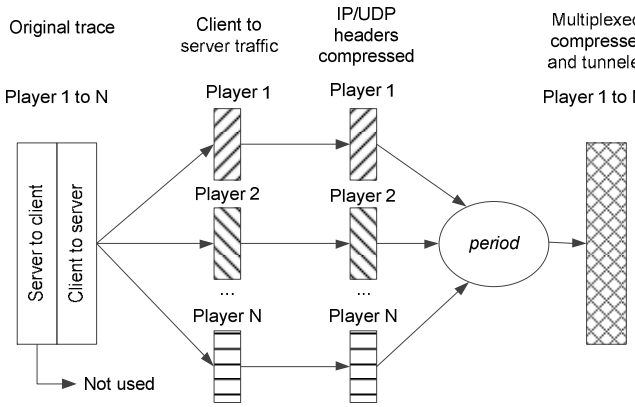
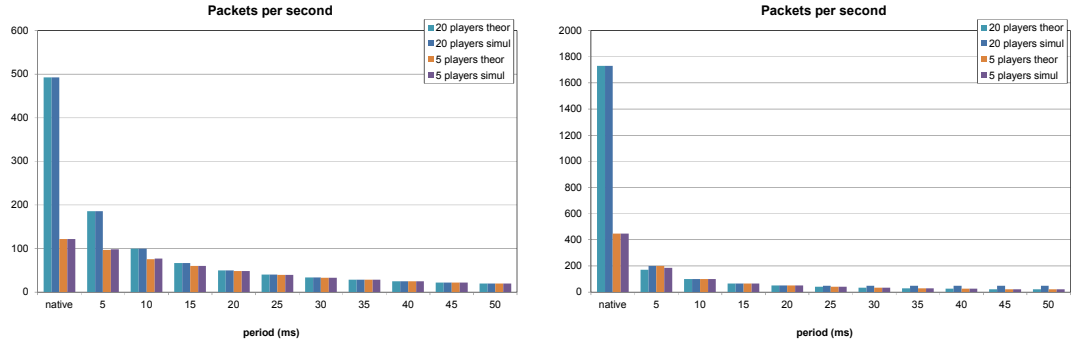


Fig. 6. Method used to build the traces.

4.2. Comparison of Analytical and Simulation results

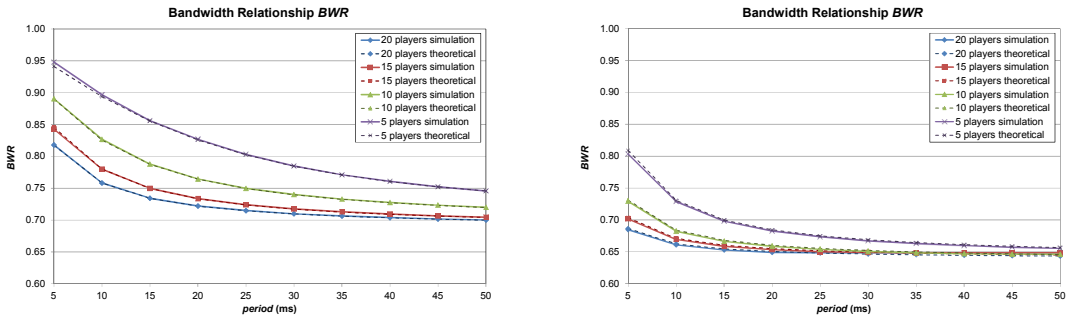
The amount of packets per second obtained in the simulations is compared to the theoretical value in **Fig. 7**. We observe that as the period increases, the amount of packets per second gets reduced. The packets per second amount tends to be the inverse of the period, irrespective of the number of players. It can be seen that *Quake III* generates a very large amount of packets per second and, consequently, the saving can be greater than that obtained for *Half Life Counter Strike 1*. It can also be observed that the theoretical and simulation values fit well.



(a) (b)

Fig. 7. Packets per second managed by the access router: a) *Half Life Counter Strike 1*; b) *Quake III*.

Fig. 8 compares the theoretical values of *BWR* with those obtained in the simulations. It can be observed that the values obtained are similar, except for minor differences when the period and the number of players are small. The cause of this is that there is a slight difference between real inter-packet times and those of the statistical model. First, we can observe that the values for the *BWR* asymptote are the expected ones (**Table 1**). *Quake III* obtains better results because its average packet size is smaller than that of *Half Life Counter Strike 1*. Another difference is that *Quake III* has values close to the asymptote for smaller values of the period. The reason for this is that the amount of packets per second is larger, so the number of packets arriving at the multiplexer will be greater for the same values of the period. If we have 20 players, a period greater than 10 or 15 ms will have no sense for this game, since the bandwidth saving will only slightly increase at the cost of more added delay. This fact is interesting, since the added delays will be small, not harming subjective quality.



(a) (b)

Fig. 8. Comparison of the theoretical and simulation results of *BWR*: a) *Counter Strike 1*; b) *Quake III*.

4.3. Influence of the Router Buffer

Access networks are one of the most common scenarios in which FPS games are played. The players' traffic is first sent via the access router. This is the most stringent bottleneck of the path to the game server, as described in [19], which shows that the buffer of the access router is of primary importance for real-time services. Thus, in this subsection we study the mutual influence of the buffer policy and TCM. Although the buffer does not appear to have a direct influence on TCM, as packets are first multiplexed and then sent to the router, there is a relationship: the longer the period, the greater the size of the packets sent to the buffer, and the bigger the bandwidth saving. The behavior of the packets in the router will depend on the buffer implementation. The added delays and losses may vary depending on the packet size.

In this subsection, the traffic traces of *Half Life Counter Strike 1* with 20 players, obtained in previous sections, will be used as the desired traffic. The scenario used for the tests (**Fig. 9**), emulates an access network, and includes three machines. First, the traffic of the game and the background traffic are sent from a host. Next, a router with different implementations sends the traffic to the Internet, and the traffic is captured at the end. This part is carried out in a testbed [35]. In order to include network and processing delays, the traffic trace is processed offline to obtain the final results.

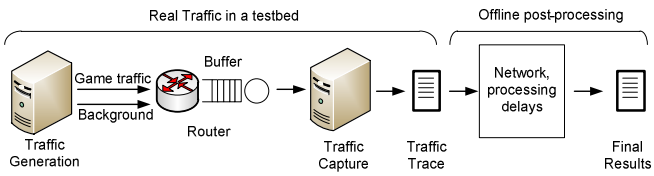


Fig. 9. Access network measurement scheme.

The traffic is sent using JTG [36] generator, which is able to send the traces exactly as they were originally as it reads packet sizes and inter-departure times from a file. A traffic model was therefore not necessary. The size distribution of background traffic is as follows [37]: 50% of the packets are of 40 bytes, 10% of 576 bytes, and 40% of 1,500 bytes. For each point of the graphs, 810 seconds of traffic have been sent. The traffic of the game and the background traffic share the same access link, which is emulated by a machine running Linux *tc* (Traffic Control). This tool limits the bandwidth at Ethernet level and implements different buffer policies. The bandwidth limit has been set to 1Mbps. The *burst* parameter of *tc* has been

set to 5,000 bytes. The buffer size is defined by limiting the maximum delay of a packet, which is the same as limiting the buffer size, as the two parameters are related by the link speed.

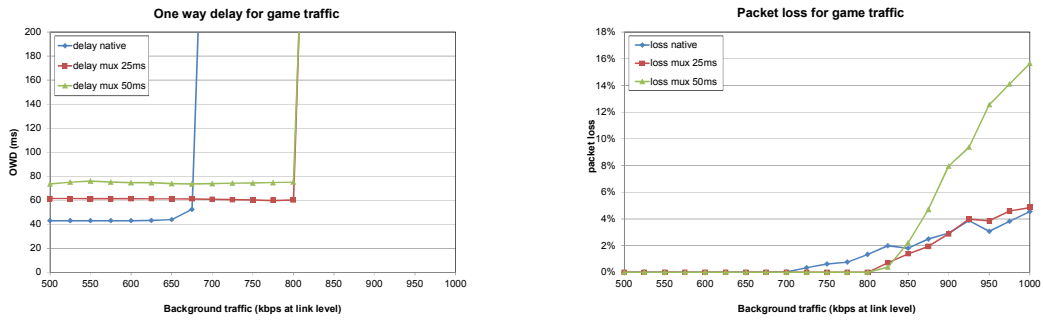
As mentioned above, in [19] an extensive study of a large number of residential Internet accesses was conducted. One of the conclusions was that many ISPs add big queuing delays, which can be of hundreds of milliseconds. As a consequence, we use two different buffer sizes: a high capacity buffer with a maximum delay of 500 ms and a time-limited buffer with a maximum delay of 50 ms.

The network delay, which is added offline, is the sum of a fixed delay of 20 ms corresponding to the geographical distance, and a lognormal delay with an average of 20 ms and a variance of 5 [38]. A processing delay is also added. As previously stated, in [27] a multiplexing scheme for VoIP was implemented and its processing delay was about 1 ms. In order to include the effect of multiplex and demultiplex, we have added a fixed delay of 5 ms.

In [39] a study of *Half Life* concluded that players would not play when latencies were above 225-250 ms. More recent studies [13] have concluded that acceptable quality can be perceived with 200 ms of delay for certain titles. So the delays added by TCM can be accepted by players. Regarding packet loss, the behavior depends on the game: while some of them stop working with packet loss of about 4%, others can work properly with this parameter at about 35% [13].

Next, we present some graphs of One Way Delay (OWD) and packet loss for both buffers, using different amounts of background traffic in order to saturate the access router. Logically, multiplexing will only be interesting when the traffic of the game has to compete with large amounts of background traffic. For each buffer we have used three traffic types: the *native* one, in which no multiplexing is applied, another using T=25 ms, and a third with T=50 ms.

Fig. 10 shows the results for the high capacity buffer. It should be noted that a small increase of the delay is introduced when multiplexing, due to the retention (half the period) and processing time in the multiplexer. The native bandwidth is 319 kbps at Ethernet level. When the total traffic exceeds the limit, the delays increase dramatically. It can also be seen that the bandwidth saving (about 120 kbps) is translated into a greater amount of background traffic that can be supported while maintaining acceptable delays.



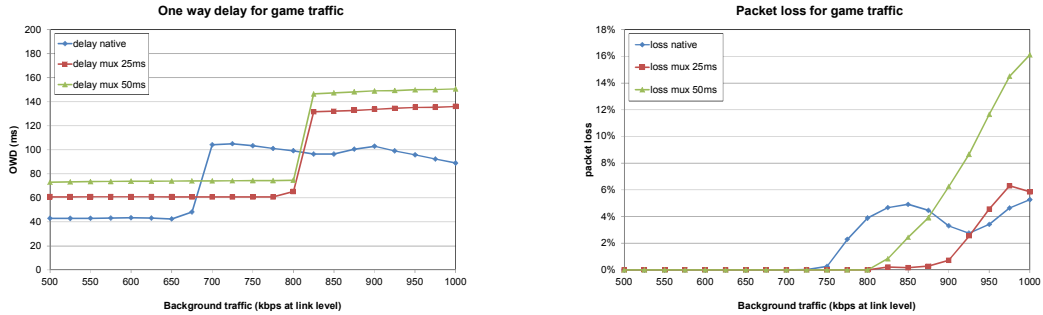
(a) (b)

Fig. 10. High capacity buffer: a) One Way Delay b) packet loss

Fig. 11 shows the results using the time-limited buffer. If we compare it with **Fig. 10**, we can see that the effects on the delay are the same as those observed for the high capacity buffer. However, the use of the time-limited buffer has the advantage of maintaining the delay below

160 ms irrespective of the amount of background traffic. There is a zone in the graph where the delay obtained when multiplexing is smaller than the native delay. This is an interesting result taking into account the hard real-time constraints of FPS games.

Another interesting phenomenon is that for the native traffic, the packet loss rate decreases as the background traffic grows from 850 to 925 kbps. This happens because the bandwidth limit is reached, so the first packets to be discarded are the big ones (1,500 bytes) as they have a greater probability of not having a place in the queue. This represents a benefit for native packets, as they are very small. The multiplexed traffic does not show this effect, because the packets are bigger.



(a) (b)

Fig. 11 Time-limited buffer: a) One Way Delay b) packet loss

There is a further remarkable effect regarding packet loss. While the use of T=25 ms achieves better results than native traffic, due to bandwidth saving, it also achieves better results than the use of T=50 ms. We can discuss this surprising result looking at the *20 players* graph in **Fig. 4c**. The values of *BWR* for 25 and 50 ms are very similar, as they are near the asymptote. In fact, the difference in terms of bandwidth is smaller than 6 kbps. But if we calculate the average packet size, we can see that in the first case it is 608 bytes and in the second it is 1,192 bytes. So, as the buffer policy penalizes big packets, it will be better not to use a long period.

Above 925 kbps of background traffic, it can be seen that native traffic suffers less packet loss than multiplexed traffic for both buffers. This is because smaller packets have less probability of being discarded. Therefore, there are some situations in which multiplexing can increase packet loss. Multiplexing will affect delay and packet loss of the game in a different manner depending on the router buffer policy.

5. Conclusions

This work has analyzed a tunneling, compressing and multiplexing method which can be used to achieve bandwidth savings by compressing headers and grouping packets into bigger ones. It can be useful for reducing the overhead of online gaming traffic, as these applications usually generate a high rate of small packets. The method uses an IP/UDP header compression protocol, PPPMux multiplexing and L2TP tunneling in order to work end-to-end.

Enterprises which develop games could be interested in reducing the bandwidth and also the number of packets per second they have to manage. The bandwidth savings can also be of interest in order to obtain better performance in access networks with limited bandwidth. The method has been tested with the traffic of FPS games, because these applications have very stringent temporal constraints, as players demand a high level of interactivity. Simulations

have been conducted in order to study the bandwidth savings, and the results show that the bandwidth can be reduced up to 38% for IPv4 and over 50% for IPv6. The added multiplexing delays can remain small if the number of players sharing the same path is sufficiently large.

Taking into account that access networks are a scenario where these games are frequently used, a comparison of the performance of native and TCM multiplexed flows of FPS has to be carried out depending on the router buffer size. It is shown that the best multiplexing solution is not necessarily the one that achieves the best bandwidth saving. Packet size also has to be considered, as some buffer policies penalize big packets. The TCM technique can help us to adapt traffic to the network conditions. If the network is better prepared for low rates of big packets, we can modify the traffic in order to adapt it to the underlying technology.

References

[1] P. Svoboda, W. Karner and M. Rupp, "Traffic Analysis and Modeling for World of Warcraft," *IEEE Int. Conference on Communications*, pp.1612-1617, 2007. Article (CrossRef Link).

[2] G. Huang, M. Ye and L. Cheng, "Modeling system performance in MMORPG," *IEEE Global Telecommunications Conference Workshops*, pp. 512- 518. 2004. Article (CrossRef Link).

[3] C. Chambers, W. Feng, S. Sahu and D. Saha, "Measurement-based Characterization of a Collection of On-line Games," Proc. 5th *ACM SIGCOM conference on Internet Measurement,* USENIX Association, Berkeley, 2005. Article (CrossRef Link).

[4] K. Chen, P. Huang and C. Lei, "Game traffic analysis: An MMORPG perspective," In *Proc. international workshop on Network and operating systems support for digital audio and video*, pp. 19-24. ACM, New York, 2005. Article (CrossRef Link).

[5] W. Feng W, F. Chang, W. Feng and J. Walpole, "Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server," *SIGCOMM Comput. Commun*. Rev. 32, p. 18, 2002. Article (CrossRef Link).

[6] M. Mauve, S. Fischer and J. Widmer, "A Generic Proxy System for Networked Computer Games," In *Proc. of 1st workshop Network and system support for games*, pp. 25-28. ACM, New York, 2002. Article (CrossRef Link).

[7] D. Bauer, S. Rooney and P. Scotton, "Network Infrastructure for Massively Distributed Games," In *Proc.* of 1st *workshop on Network and system support for games*, pp. 36-43. ACM, New York, 2002. Article (CrossRef Link).

[8] B. Thompson, T. Koren and D. Wing, RFC 4170: "Tunneling Multiplexed Compressed RTP" (TCRTP), 2005.

[9] S.H. Batool and K. Mahmood, "Entertainment, communication or academic use? A survey of Internet cafe users in Lahore, Pakistan," *Information Development* vol. 26. pp 141-147, 2010. Article (CrossRef Link).

[10] M. Gurol, T. Sevindik, "Profile of Internet Cafe users in Turkey," *Telematics and Informatics*, Vol. 24, Issue 1, pp 59-68, 2007. Article (CrossRef Link).

[11] B. Furuholt, S. Kristiansen and F. Wahid, "Gaming or gaining? Comparing the use of Internet cafes in Indonesia and Tanzania," *The Intern. Inform. & Library Review*, Vol. 40, Issue 2, pp 129-139, 2008. Article (CrossRef Link).

[12] M. Degermark, B. Nordgren and D. Pink, RFC 2507: "IP Header Compression," 1999.

[13] S. Zander and G. Armitage, "Empirically Measuring the QoS Sensitivity of Interactive Online Game Players" *Australian Telecommunications Networks & Applications Conf.*, Sydney, 2004.

[14] J. Saldana, J Fernández-Navajas, J. Ruiz-Mas, J.I. Aznar, E. Viruete and L. Casadesus "First Person Shooters: Can a Smarter Network Save Bandwidth without Annoying the Players?," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 190-198, 2011. Article (CrossRef Link).

[15] P. Branch and G. Armitage, "Extrapolating Server To Client IP traffic From Empirical Measurements of First Person Shooter games," *Proc. 5th ACM SIGCOMM workshop on Network and system support for games* (NetGames '06). ACM, NY, USA, 2006. Article (CrossRef Link).

[16] W. Feng, F. Chang, W. Feng W and J. Walpole, "A Traffic Characterization of Popular On-Line

Games," *IEEE/ACM Trans. Networking*, pp. 488-500, 2005. Article (CrossRef Link).
[17] S. Ratti, B. Hariri and S. Shirmohammadi, "A Survey of First-Person Shooter Gaming Traffic on the Internet," *IEEE Internet Computing*, vol 14, no. 5, pp. 60-69, 2010. Article (CrossRef Link).
[18] A. Dhamdhere and C. Dovrolis "Open issues in router buffer sizing", *Comput. Commun*. Rev., vol. 36, no. 1, pp. 87-92, 2006. Article (CrossRef Link).
[19] M. Dischinger, A. Haeberlen, K.P. Gummadi and S. Saroiu, "Characterizing residential broadband networks," In *Proc. 7th ACM SIGCOMM conference on Internet measurement*, ACM, New York, NY, USA, pp 43-56,- 2007. Article (CrossRef Link).
[20] K. Lee, B. Ko and S. Calo, "Adaptive Server Selection for Large Scale Interactive Online Games," In *Proc. 14th International Workshop on Network and operating systems support for digital audio and video*, pp. 152-157. ACM, New York, 2004. Article (CrossRef Link).
[21] G. Reina, E. Biersack and C. Diot, "Quiver: a Middleware for Distributed Gaming," *22nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, Toronto, Canada, Jun. 2012.
[22] J. L. Miller and J. Crowcroft, "The near-term feasibility of P2P MMOG's," In *Proc. 9th Annual Workshop on Network and Systems Support for Games,* Piscataway, NJ, USA, Art. 5, 6 pag. 2010.
[23] V. Jacobson, RFC 1144: Compressing TCP/IP Headers for Low-Speed Serial Links, 1990.
[24] L-E Jonsson, G. Pelletier and K. Sandlund, RFC 4995: The RObust Header Compression (ROHC) Framework, 2007.
[25] A. Couvreur A, L.M. Le-Ny, A. Minaburo, G. Rubino, B. Sericola and L. Toutain, "Performance analysis of a header compression protocol: The ROHC unidirectional mode," *Telecommunication Systems*, vol. 31, no. 6, pp. 85-98, 2006. Article (CrossRef Link).
[26] E. Ertekin and C. Christou, "Internet protocol header compression, robust header compression, and their applicability in the global information grid," *IEEE Communications Magazine*, vol. 42, pp. 106-116. 2004. Article (CrossRef Link).
[27] H. Sze, C. Liew, J. Lee and D. Yip, "A Multiplexing Scheme for H.323 Voice-Over-IP Applications," *IEEE Journal Select. Areas Commun*. vol. 20, pp.1360-1368, 2002. Article (CrossRef Link).
[28] J. Saldana, D. Wing, J. Fernandez-Navajas, M.A.M. Perumal and F. Pascual, "draft-saldana-tsvwg-tcmtf-03, Tunneling Compressed Multiplexed Traffic Flows," July 2012.
[29] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, E. Viruete Navarro and L. Casadesus, "Influence of Online Games Traffic Multiplexing and Router Buffer on Subjective Quality," in Proc. *CCNC 2012- IEEE Workshop DENVECT*, pp. 482-486, Las Vegas. Jan. 2012. Article (CrossRef Link).
[30] C. Griwodz and P. Halvorsen. 2006, "The fun of using TCP for an MMORPG," In Proc. Int. workshop on Network and operating systems support for digital audio and video (NOSSDAV '06). ACM, New York, NY, USA, Article 1, pp. 7. Article (CrossRef Link).
[31] J. Saldana, J. Fernández-Navajas, J. Ruiz-Mas, J.I. Aznar, L. Casadesus and E. Viruete, "Comparative of Multiplexing Policies for Online Gaming in terms of QoS Parameters," *IEEE Communications Letters*, vol. 15, no. 10, pp. 1132-1135, 2011. Article (CrossRef Link).
[32] S. Zander and G. Armitage, "A traffic model for the Xbox game Halo 2," In *Proc. of International Workshop on Network and operating systems support for digital audio and video*, ACM, New York, NY, USA, pp 13-18. 2005. Article (CrossRef Link).
[33] T. Lang, G. Armitage, P. Branch and H. Choo, "A Synthetic Traffic Model for Half-Life," *Australian Telecom, Networks and Applications Conference,* Melbourne, Australia, 2003.
[34] L. Stewart and P. Branch, HLCS, Map: dedust, 5 players, 13Jan2006. Centre for Advanced Internet Architectures SONG Database, http://caia.swin.edu.au/sitcrc/hlcs_130106_1_dedust_5_ fragment.tar.gz, 2006. Accessed 5 April 2011.
[35] J. Saldana, E. Viruete, J. Fernández-Navajas, J. Ruiz-Mas and J.I. Aznar, "Hybrid Testbed for Network Scenarios," *SIMUTools, 3rd International Conference on Simulation Tools and Techniques*. Torremolinos, Málaga, Spain, 2010. Article (CrossRef Link).
[36] J. Manner, JTG, http://www.cs.helsinki.fi/u/jmanner/software/jtg/, Accessed 11 August 2012.
[37] Cooperative Association for Internet Data Analysis: *NASA Ames Internet Exchange Packet Length Distributions*.

[38] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson and R. Steinmetz, "Modeling the internet delay space based on geographical locations," In *Proc. of 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2009. Article (CrossRef Link).
[39] T. Henderson, "Latency and User Behaviour on a Multiplayer Game Server," *Lect. Notes Comp. Science*, Springer Berlin/ Heidelberg, pp 1-13, vol 2233, 2001. Article (CrossRef Link).

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).

Luis Casadesus received his B.S in Computer Science from the University of La Habana in 2001, and his M.S. in Mobile Networks from the University of Zaragoza, in 2009. He is currently a Ph.D. candidate at the Communications Technologies Group of the University of Zaragoza. His research interests focus on Quality of Service and Quality of Experience in Multimedia Services, like video streaming, videoconferencing and networked online games.