First Person Shooters: Can a Smarter Network Save Bandwidth without Annoying the Players?

Jose Saldana, Julián Fernández-Navajas, José Ruiz-Mas, José I. Aznar, Eduardo Viruete, and Luis Casadesus, University of Zaragoza

ABSTRACT

Real-time services are very challenging for current network infrastructures. One of these services is online gaming, which has acquired more importance in the last years. The quality experienced by users could be improved by the use of a smarter network, which includes some proxies near the players in order to transfer intelligence from the game server to network borders. Proxies have been largely used for other services, such as web browsing and VoIP. First Person Shooters are a popular genre of online games that generate high rates of small packets. This issue becomes important in some scenarios where many flows of a game share the same path, as would happen between a proxy and the central server of a game. These flows could be multiplexed to improve efficiency by reducing packet overhead, thus allowing a bigger number of players to share the same link. A method named Tunneling, Compressing, and Multiplexing (TCM) has been proposed to multiplex these flows. In this article, this method has been tested using the traffic of eight popular First Person Shooters. The method has shown its ability to achieve bandwidth savings of about 30 percent for IPv4 and above 50 percent for IPv6 for all the games. The added delay and jitter are small enough to not annoy players.

INTRODUCTION

During the last years, many real-time services using the Internet have been deployed. They are very challenging for current network infrastructures, as they have to deliver information with low latency from the origin to the destination. Although they do not frequently use big amounts of bandwidth, they usually generate high rates of small packets.

Two of the most popular genres are the Massive Multiplayer Online Role Playing Games (MMORPG) and First Person Shooters (FPS). The former ones create a virtual world in which thousands of people can simultaneously play. Every single player manages a character, which can obtain new abilities and powers. This game genre requires reliability, but not very high interactivity, since the fights are not based on shooting but on the powers of each character. This is why these games mainly use TCP. Session times are usually long.

FPSs are typically played by some tens of gamers sharing a virtual scenario where they have to kill enemies or accomplish a mission. Each user has a weapon, which can be improved according to the results of the game. The time of a match can be short, but users play a number of matches in a session. The real-time requirements are strict, as the movements and shots are very fast and frequent, so they use UDP. These games are typically developed for high-end desktop PCs and consoles, as smart graphic cards are required.

FPSs use client-server architectures. Some reasons for this are the convenience of maintaining the consistency of the game, synchronization, and cheating prevention. Other reasons are simply commercial: game providers can charge for use or sell the server software.

Every time a new title is released, game providers have to be prepared to support it, which means servers with high processing capacity and high-bandwidth networks. Thus, the server can be a bottleneck introducing a limitation on the number of simultaneous players unless the developer has previously overprovisioned the resources.

The gamers are very difficult to satisfy: in [1] a study of their behavior was presented, and the conclusions are that they do not tend to be loyal to a server, and they have very little patience: if a server does not work properly, they go to another and never return. Another problem is the unfairness that can be caused when the delays of the players are different. Some games solve it by increasing the delay of the rest of the users.

All in all, there is a need for new architec-

0163-6804/11/$25.00 © 2011 IEEE

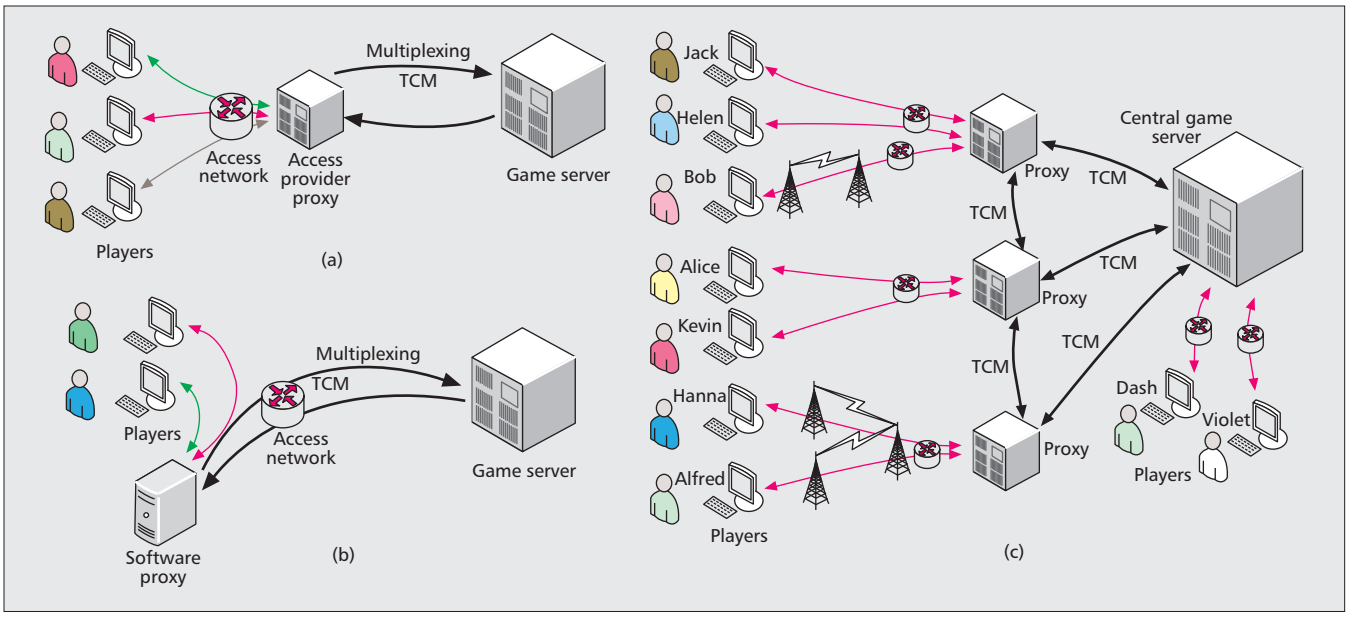

Figure 1. *Network infrastructure: a) proxy managed by the access provider; b) software proxy; c) infrastructure of the game provider.*

tures and techniques to translate some intelligence from the server to the borders of the network. This tendency can be observed in next-generation networks, and also in new emerging services. Traditional ones, like web browsing and voice over IP (VoIP), have largely used the concept of a proxy as a key component that transfers workload from the core to the borders. This concept can also be applied to online gaming service. In fact, many years ago this architecture was proposed by Mauve *et al.* [2], who studied the advantages of using proxies for supporting online gaming. Bauer *et al.* [3] also proposed the use of *booster boxes* which could be placed next to the access router, enabling some application functions to be assumed by the network. If the network is smarter, some quality of service (QoS) mechanisms can be implemented in order to improve the users' experience. A proxy can automatically detect and multiplex flows with the same origin and destination in a transparent way.

One of the big issues regarding FPS traffic deals with scalability: server-to-client traffic amount roughly increases with the square of the number of users, since the application has to know the actions of the rest of the players involved. The packets generated by the server may increase their size as the number of players grows, while the size of the ones generated by the client remains constant. Consequently, if a number of players are connected through the proxy, the game server can send a single packet to it, and the proxy will forward this information to all the players. This can be thought as the first utility of a proxy.

Other problems relative to client-to-server traffic are, on one hand, that it consists of high rates of small UDP packets, which entails low efficiency; on the other hand, there is the high number of packets per second the router has to manage when many players share the same link. In [4] the issue of the infrastructure for supporting games was studied, concluding that the most stringent bottleneck can be the capacity of the router for managing a big number of packets per second, more than the bandwidth limit.

The use of proxies also allows us to use a multiplexing technique when a certain number of players share the same connection. Multiplexing is a well-known idea that has already been used for other services like VoIP, and can also be applied in this scenario, simultaneously alleviating both bandwidth and packets per second problems.

In this article we present a multiplexing method for online gaming traffic that can be used when many players share the same route in the context of a game provider using proxies. It will be tested for client-to-server traffic of FPS games, as this traffic represents a good example which comprises three main characteristics: very stringent delay requirements, high overhead, and big amounts of packets per second.

The remainder of the article is organized as follows. The next section presents different network infrastructures where proxies can be applied. We then summarize the method proposed to optimize the traffic of FPSs. After that we present details, and discuss the tests and results, and the article ends with the conclusions.

Network Infrastructures

There are different infrastructures where proxies can be applied: they can be managed by the access provider, by the final users, or by the game provider. Logically, they can be combined in different ways.

Access Provider and Final Users

Figure 1a shows a proxy next to the access router, managed by the access provider, which obtains information on the network state and makes decisions in order to improve quality. It could implement some standard functions capable of working for many different games, bringing more opportunities to generate revenues.

The gamers are very difficult to satisfy; they do not tend to be loyal to a server, and they have very little patience: if a server does not work properly, they go to another and never return.

The access provider can establish an agreement with the user in order to optimize his gaming traffic.

As an example of a real scenario with many players sharing the same access, we can consider an Internet café. Nowadays, they are a very popular way of accessing Internet services in many countries. In [5], a study of the behavior of these businesses' clients was presented, and it concluded that online gaming is one of their most important activities. Internet cafés are present all over the world, but they have a special significance in developing countries. The telecommunication infrastructures vary from one place to another, and sometimes they may not have the latest access technologies, so techniques for bandwidth saving are of great interest.

Another interesting question regards the asymmetry of access technologies: while they may provide an acceptable bandwidth in the downlink, it is usual that the uplink has very low speed. Thus, bandwidth savings for client-to-server traffic may even be of greater interest.

Another option is the use of a software proxy, which a group of users or the owner of the café can install in their local networks (Fig. 1b). It could be distributed with the game, as local servers are.

Game Provider

As pointed out in the introduction, proxies can be seen as an extension of the server, which delegates some tasks to them (Fig. 1c). The game provider can use them to alleviate the work load of the central server, avoiding bottlenecks by sharing information and intelligence around the network.

A proxy should be able to interact with the others, so the traffic of many players will share the same path, and an algorithm that achieves bandwidth savings can also be very useful for them.

These proxies could also achieve some other advantages. If we look at Fig. 1c, perhaps Bob, who has a wireless connection with a big delay, will not be able to play against Alice or Dash, but maybe his nearest proxy could manage a suitable game with Helen and Jack. So some proxies can manage their own games with the players who are near them.

Traffic Management

In this section we first summarize the characteristics of FPS's traffic, and then the multiplexing method is presented.

FPS Traffic Characteristics

FPSs generate different traffic depending on the moment. For instance, some TCP traffic can be generated while the game is starting: map downloads and uploads, personalization of some images of the characters, and so on. Nevertheless, in this article we only focus on active game traffic (i.e., generated while the fight is going on).

The traffic has two different behaviors: the clients generate small UDP packets (typically some tens of bytes) with different patterns for interpacket delays. These flows carry the information of the actions of the players to the server. The amount of traffic generated by a player does not depend on the number of players of the game. Only some consoles permit up to four players to play the same game (e.g., Halo2 for Xbox), so in this case the traffic generated by a single machine can vary with the number of players.

The server has to keep the state of the game and communicate the movements of one player to the rest, so the traffic generated by the server grows with the square of the number of players. These packets are typically bigger than the ones generated by the clients.

Ratti *et al.* [6] presented a survey of the traffic patterns of 17 different FPS games collected from the literature. Each game is characterized by four different statistical distributions: interpacket times and sizes for both client and server. These statistics are interesting in order to conduct research studies, so their authors frequently also perform comparisons of the proposed models and the real traffic traces, using Q-Q plot, K-S test, or other statistical validations.

Traffic Optimization: Tunneling, Compressing, and Multiplexing

In this subsection we summarize the proposed method, which can save bandwidth and packets per second. A more technical study of the theoretical savings can be found in [7]. The method is based on TCM. It has been adapted from the Internet Engineering Task Force (IETF) standard RFC 4170, which is capable of obtaining bandwidth savings for Real-Time Transport Protocol (RTP) flows. Logically, there must be a counterpart: small delays will be added to the packets, so we have to test whether they are acceptable or impair the user experience.

Figure 2a shows the protocol stack. First, IP/UDP headers are compressed. A survey of the different IETF header compression standards can be found in [8]. As a summary, we can say that they use the repeatability of the headers and the fact that many fields increase by one from one packet to the next in order to compress them. The two parts share a context, that is, the information necessary to rebuild the headers at the destination. This context is transmitted with the first headers and refreshed whenever some field changes significantly.

Some standards compress IP/UDP/RTP headers, while others only work with IP/UDP or IP/TCP ones. Since our traffic is not RTP, the two standards that are suitable for this work are IPHC and ROHCv2. The first can compress IP and UDP headers to only 2 bytes each. The second is more recent and uses more sophisticated compressing algorithms, but requires more processing capacity.

Next, PPPMux is used in order to multiplex packets, inserting a small header before each compressed one. Finally, all the compressed packets are encapsulated into a PPP packet and sent using a Layer 2 Tunneling Protocol (L2TP) tunnel.

TCM defines a period at the multiplexer, and sends a packet including all the received ones at

the end of each period (Fig 2b). There are two exceptions: when no packet has arrived, nothing is sent; if there is only one packet, it is sent in its native form, as the tunnel would only make it bigger.

The added delay will be half the period on average, plus processing delays. In the next sections we study whether these delays are acceptable for players or not.

In this article we test TCM for client-to-server traffic for two reasons: first, since the packets are very small, the overhead is expected to be significantly reduced; and second, the existence of many access networks with asymmetric bandwidths makes it more interesting to reduce bandwidth and packets per second for the uplink.

Figure 3a illustrates the overhead problem of real-time flows. In the figure, which is real scale, we have first represented an IP/TCP packet of 1500 bytes, which is the maximum size in many wired networks. This kind of packet can be typically used for non-real-time services like email, web, FTP, and peer-to-peer (P2P) file sharing applications. It can be observed that its overhead is very small. η represents the efficiency (i.e., the relationship between the payload and the total size of the packet at IP level). As an example of the bad efficiency (33 percent) of many real-time services, a VoIP packet is presented.

In the same figure, if we look at the FPS (Counter Strike 1) server-to-client packet, we can observe that the efficiency is worse (85 percent), but it can still be acceptable. Real-time constraints make the application send information at a high frequency, so the payload is small, while standard Internet IP/UDP headers need 28 bytes. Finally, if we look at the client-to-server packets, we notice that the efficiency has descended to 68 percent. So if TCM is applied, we can pull it up to 83 percent. The figure shows the savings achieved for only four packets, but that number can be significantly bigger.

Nowadays, IPv4 is slowly being replaced by its newer version, IPv6, so we think it also has to be considered. In [9] a report about the adoption of this new version was presented. One of its drawbacks is the introduction of more overhead, as its header is 20 bytes bigger than that of its predecessor. Figure 3b also illustrates the efficiency problem, but using IPv6. It can be seen that while the added overhead is not significant for big TCP packets, its influence makes the efficiency of client-to-server packets of the FPS game fall to 56 percent. So in the scenario we will find in a few years, the importance of bandwidth saving techniques will necessarily grow due to the full integration of IPv6.

To close this section, we show the theoretical bandwidth savings TCM can achieve. We measure bandwidth saving (BS) as

$$BS = \frac{BW_{native} - BW_{multiplexed}}{BW_{native}}. \quad (1)$$

If the period or the number of players grows, the number of multiplexed packets will also grow, and the common header will be shared by a larger number of packets, making its effect become smaller. The behavior of this saving is asymptotic, as there is a limit we can never achieve, given by the next expression,

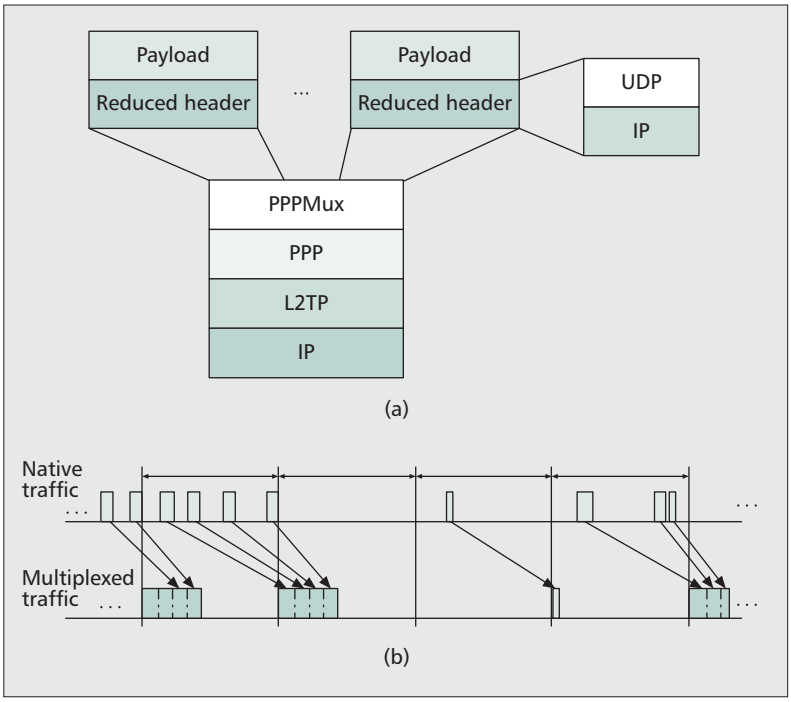

Figure 2. *TCM behavior: a) protocol stack; b) multiplexing policy.*

$$BS_{asymptote} = \frac{NH - MH - CH}{PS}, \quad (2)$$

where *NH* represents the normal header of a UDP packet (28 bytes for IPv4 and 48 for IPv6), *MH* is the multiplexed header (2 bytes), *CH* is the expected value of the compressed header, the size of which depends on the compressing algorithm, and *PS* is the expected value of the packet size.

We represent in Fig. 4 the theoretical values of bandwidth saving for Counter Strike 1, using IPv4, as obtained in [7]. The asymptotic behavior can be observed for both parameters: number of players and period. This must be taken into account while setting the value for the period. If we use a very big one, the bandwidth saving can increase very little while the added delays can significantly grow. As an example, if we look at the *20 players* graph of Fig. 4 b, we can see that increasing the period from 25 to 50 ms will only increase the bandwidth saving by 2 percent, so such a big value for the period will not be interesting.

Regarding the reduction in packets per second, it can be seen that TCM sends a packet every period, so the amount of packets per second is roughly the inverse of the period, despite the number of packets the game generates.

Tests and Results

This section presents an analysis of the use of TCM for eight different FPS games. They have been selected in order to have a variety of inter-packet delays and packet size behaviors. Simulations have been conducted using both real and synthetic traces.

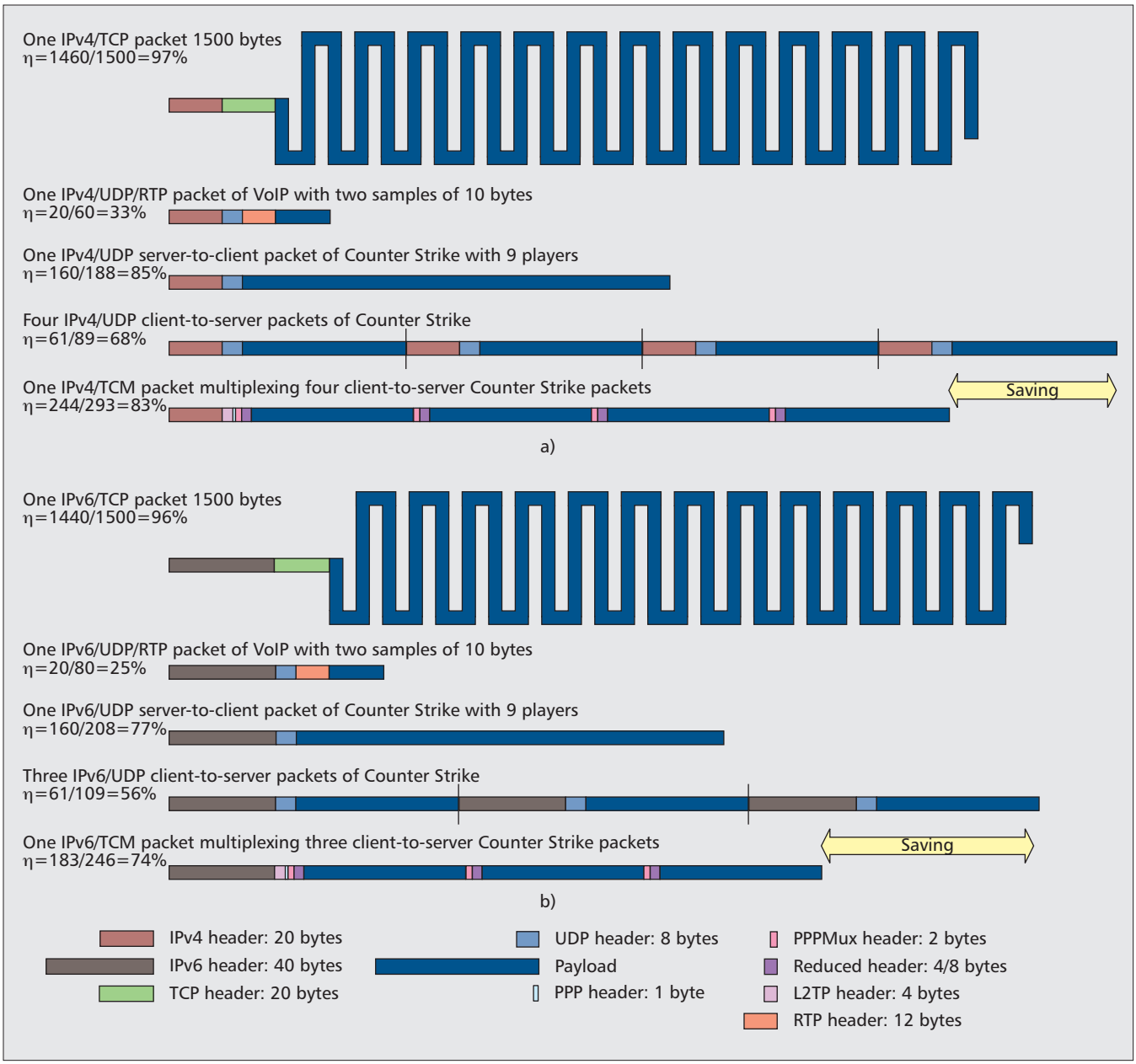

Figure 3. *Examples of packet efficiency for a) IPv4; b) IPv6. Packet sizes are real scale.*

Test Methodology

As previously mentioned, there are in the literature many studies related to the behavior of different FPS games. In our research, we have tried to use real traces in order to achieve better performance and realism. Traces of five of the selected games (Counter Strike 1, Counter Strike 2, Quake 3, Quake 4, and Wolfenstein: Enemy Territory) are available at the web site of the CAIA project (http://caia.swin.edu.au), well documented with information about the configuration of each computer and the conditions in which they were collected. In these five cases, we have downloaded the original game traces, separated the client-to-server traffic, and combined them in order to obtain traces for 5, 10, 15, and 20 players. This can be done because of the independence of client-to-server traffic with respect to the number of players of the game. As an example, the *20 players* trace is the addition of three traces: 9, 6, and 5 players.

For the other three games (Halo 2, Quake 2, and Unreal Tournament 1.0) the traces have been generated using theoretical models referenced in the survey of Ratti *et al.* [6]. For Halo 2 we have used the traffic of one player in a console.

Next, the traffic of each player is separated, and the header compressing algorithm is applied. Finally, simulation is used to multiplex all the compressed flows using different values for the period. For IPv6 tests, the sizes of the headers have been properly modified in order to obtain the correct packet size.

Results and Discussion

In Fig. 5 we present four graphs for each game: first, the histograms of packet size for IPv4 (including IP and UDP headers) and interpacket

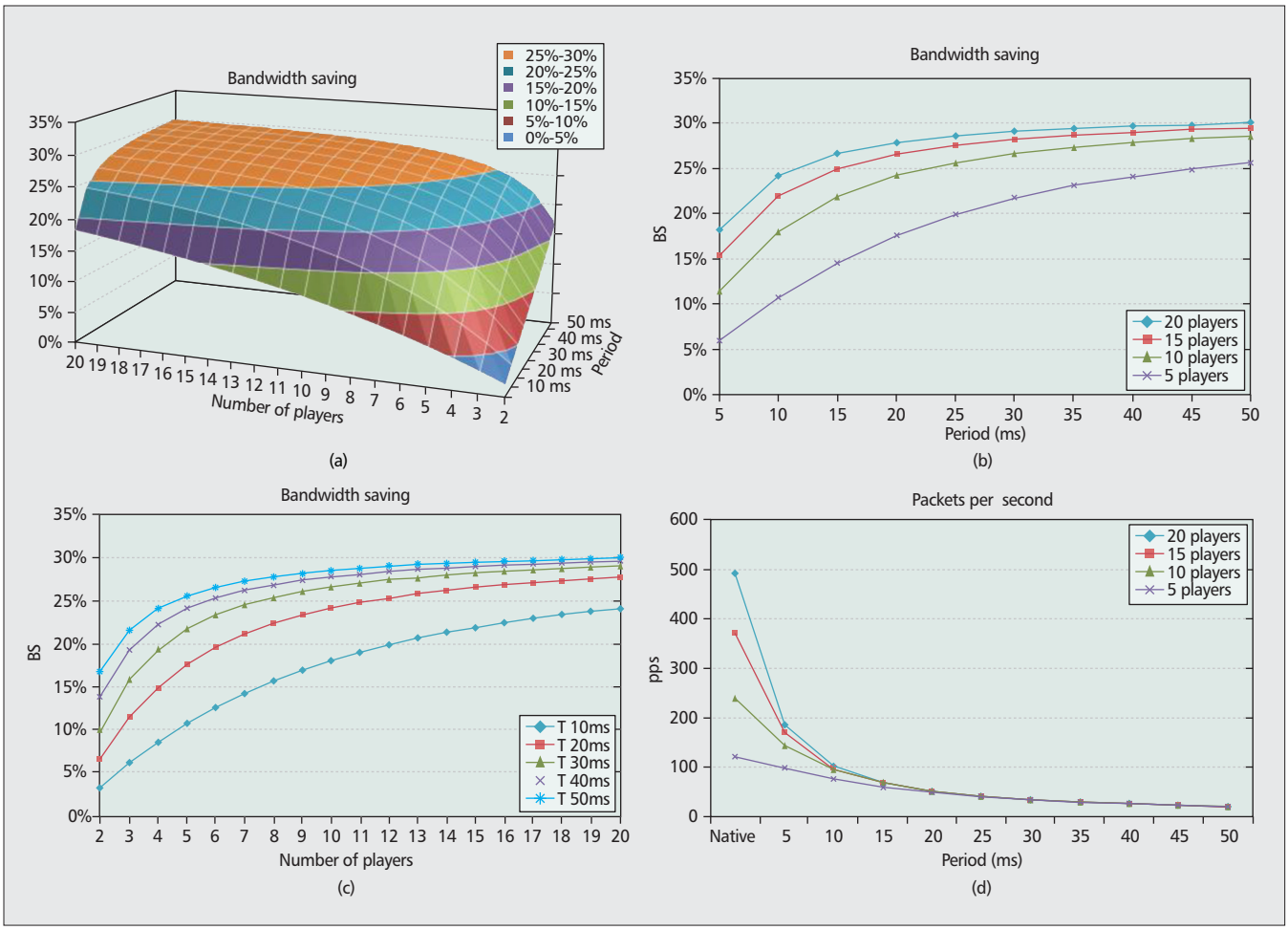

Figure 4. *a) Theoretical bandwidth saving for IPv4 as a function of the period and number of players; b) bandwidth saving as a function of the period; c) bandwidth saving as a function of the number of players; d) packets per second as a function of the period.*

time are presented. Next, bandwidth savings for IPv4 and IPv6 are depicted as a function of the period. For each game, we also present the year and the engine it uses, and also the average packet size for IPv4 and the number of packets per second it generates.

A first observation that can be made is that the saving graphs are the same as the ones expected theoretically, like the ones in Fig. 4. As an example, for Counter Strike 1 the biggest difference is about 1 percent.

Regarding packet size histograms, it can be seen that there is a big variety: some of them, like Halo2, only generate packets of a number of different sizes; others, like Unreal Tournament or Quake3, have a small range of sizes. But all of them generate small packets: the averages range from 57 to 79 bytes.

If we look at interpacket time histograms, we can observe that some titles have very regular patterns, like Unreal Tournament, which sends a packet every 25 ms, or Counter Strike 1, which in OpenGL graphic mode has two possible interpacket times: 33 and 50 ms. Others, like Counter Strike 2 and Quake 4, have a bigger range of variation.

With respect to bandwidth saving graphs, we will make two observations: first, the titles which achieve the biggest savings are the ones with smaller packet sizes, like Unreal Tournament, due to the important overhead they present. Second, the games that generate the highest amounts of packets per second are the ones that achieve good results faster (i.e., with small values for the period). For instance, if we look at the IPv6 graph of Quake3, we see that for 5 players and 5 ms the bandwidth saving is around 30 percent.

To summarize the results we have included in Fig. 6 four different values for each title: first, the obtained saving with a period of 10 ms for 5 and 20 players; next, the maximum saving achieved in the simulations; and finally, the asymptotic saving. Now, let us discuss the results presented in Fig. 6.

A first question which has to be explained is the difference between the third and fourth columns. The differences are small (about 2 percent), and are mainly caused by the asymptotic behavior, but in some cases, especially for the games which generate a high number of packets per second, there is another limitation: if the maximum size of a packet (1500 bytes) is reached, a packet is sent and a new period is started. This only occurs in Wolfenstein, Quake 3 and Quake 4.

Regarding the second column, which corresponds to a small value of the period, it can be

Figure 5. *Packet size histogram at IP level for IPv4; interpacket time histogram; bandwidth saving for IPv4; bandwidth saving for IPv6.*

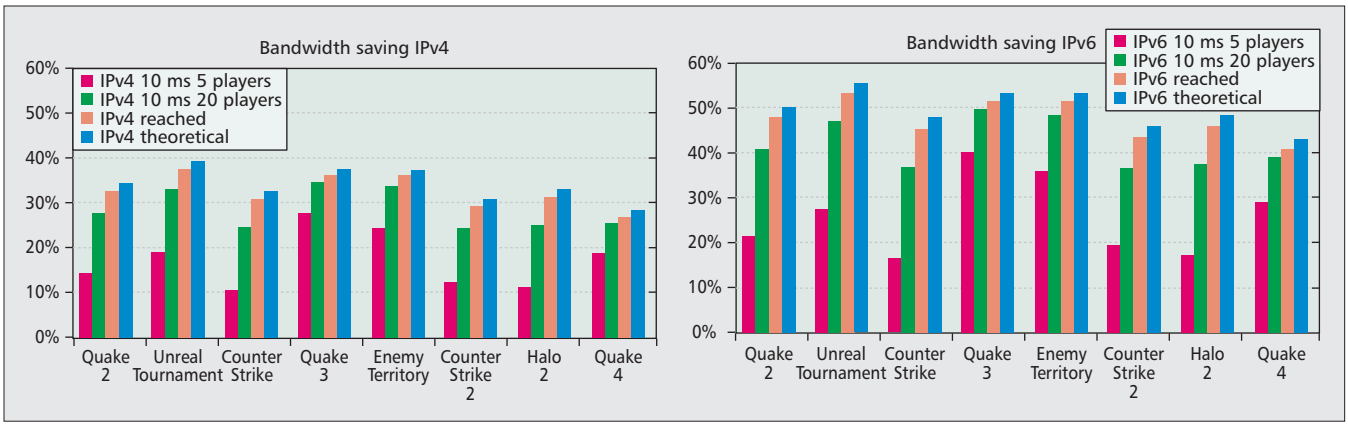

Figure 6. *Summary of bandwidth savings that can be achieved for each title.*

seen that all the games achieve good results for 20 players. This means that with a big number of them, the behavior will always be good. But if we look at the first column (which is the same value but for 5 players), we can observe important differences between titles. Once again, the games with high amounts of packets per second, like Quake 3, achieve good results even with a small number of players.

For IPv4, the maximum achieved savings range from 26 to 36 percent, whereas for IPv6 they go from 40 to 52 percent. The difference between the savings of IPv4 and IPv6 is around 15 percent. Logically, the extra overhead introduced by IPv6 makes the method achieve better results.

Subjective Evaluations

In the title of the article we talk about not annoying the players. Is this true? We have achieved bandwidth savings, but obviously there is a counterpart, which in our case is delay and jitter. The delay is mainly produced by the retention time in the multiplexer, which is half the period average. A small processing delay is also added. On the other hand, some delays may be reduced because of bandwidth savings (e.g., buffering delays). Finally, some jitter is added, as some packets are delayed more than others.

Other parameters like packet loss are not directly affected by TCM, although the loss probability can be modified by the increase of packet size. There is also another effect: bandwidth saving reduces the saturation of the router, so the loss probability may be reduced. But these two effects are not direct, so they will not be deeply studied here.

There are some studies concerning the behavior of gamers that recommend the delay not be over 200–225 ms. Other studies have developed perceived quality models, similar to the ones that exist for VoIP. We will use one of them as an example to find out whether the subjective quality will be harmed by TCM: Quake 4 G-Model [10]. Based on subjective tests, a formula for the mean opinion score (MOS), which ranges from 1 to 5, was developed. It has a network impairment factor, and the MOS is obtained using a polynomial function depending on it. If delay and jitter are added, it goes down.

We have used the presented results for Quake 4, and some new calculations have been performed in order to analyze the G-Model results. They are based on the delay and jitter added by TCM. We have obtained that MOS can be above 3.5 using a period of 20 ms with 20 players if network delay is also 20 ms. But if network delay is 40 ms, the period has to be reduced to 15 ms. MOS can be above 3 with 40 ms of network delay even with a period of 45 ms.

Conclusions

This article has presented a method to be used by the network infrastructure in order to save bandwidth when many flows of FPS games share the same path. It is based on tunneling, compressing, and multiplexing many packets into a bigger one.

The scenarios of interest have been discussed, showing that smarter network infrastructures would be interesting in order to improve the quality experienced by the gamers. These users are very difficult to satisfy, as they are very sensitive to delays and other network impairments.

The proposed method presents good performance: 30 percent of bandwidth can be saved for IPv4. This figure rises up to 50 percent for IPv6. Finally, it has been shown that the proposed method does add delay or jitter, but they do not impair the experienced quality.

Acknowledgments

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

References

[1] C. Chambers *et al.*, "Measurement-based Characterization of a Collection of On-line Games," *Proc. 5th ACM SIGCOM Conf. Internet Measurement*, Berkeley, CA, 2005.
[2] M. Mauve, S. Fischer, and J. Widmer, "A Generic Proxy System for Networked Computer Games," *Proc. NetGames '02*, 2002, pp. 25–28.
[3] D. Bauer, S. Rooney, and P. Scotton, "Network Infrastructure for Massively Distributed Games," *Proc. NetGames '02*, 2002, pp. 36–43.
[4] W. Feng *et al.*, "A Traffic Characterization of Popular On-Line Games," *IEEE/ACM Trans. Net.*, vol. 13, no. 3, June 2005, pp. 488–500.
[5] M. Gurol and T. Sevindik, "Profile of Internet Cafe Users in Turkey," *Telematics and Informatics*, vol. 24, issue 1, Feb. 2007, pp. 59–68.

[6] S. Ratti, B. Hariri, and S. Shirmohammadi, "A Survey of First-Person Shooter Gaming Traffic on the Internet," *IEEE Internet Computing*, Sept./Oct. 2010, pp. 60–69.
[7] J. Saldana *et al.*, "Bandwidth Efficiency Improvement for Online Games by the Use of Tunneling, Compressing and Multiplexing Techniques," *Proc. SPECTS '11*, The Hague, Netherlands, June 2011.
[8] E. Ertekin and C. Christou, "Internet Protocol Header Compression, Robust Header Compression, and Their Applicability in the Global Information Grid," *IEEE Commun. Mag.*, vol. 42, 2004, pp. 106–16.
[9] L. Colitti *et al.*, "Evaluating IPv6 Adoption in the Internet," *Proc. 11th Int'l. Conf. Passive and Active Network Measurement*, Springer-Verlag, Apr. 2010, pp 141–50.
[10] A. F. Wattimena *et al.*, "Predicting the Perceived Quality of a First Person Shooter: The Quake IV G-Model," *Proc. NETGAMES '06*, Article 42.

BIOGRAPHIES

JOSE SALDANA (jsaldana@unizar.es) received his B.S. and M.S. in telecommunications engineering from the University of Zaragoza, Spain, in 1998 and 2008, respectively. He is currently a Ph. D. candidate in the Department of Engineering and Communications of the same university. His research interests focus on quality of service in multimedia services, like VoIP and networked online games.

JULIAN FERNANDEZ-NAVAJAS (navajas@unizar.es) received his telecommunications engineering degree from the Polytechnic University of Valencia, in 1993, and his Ph.D. degree from the University of Zaragoza in 2000. He is currently an associate professor in the EINA, University of Zaragoza. His professional research interests are in QoS, network management, telephony over IP, mobile networks, online gaming, and other related topics.

JOSÉ RUIZ MAS (jruiz@unizar.es) received his telecommunications engineering degree from Universitat Politècnica de Catalunya (UPC) in 1991 and his Ph.D. degree from the University of Zaragoza in 2001. He is currently an associate professor in the EINA, University of Zaragoza. His research activity lies in the area of Quality of Service in Multimedia Services and the provision of methodologies and tools to assess the perception of the end user (Quality of Experience, QoE).

JOSÉ IGNACIO AZNAR (jiaznar@unizar.es) received his B.S. in telecommunications engineering in 2008 and M.S. in 2010 from the University of Zaragoza. His main research interests have been focused on the optimization and enhancement of IMS-based architectures; currently, he is involved in several research lines related to estimation, control, and monitoring of QoS parameters.

EDUARDO VIRUETE (eviruete@unizar.es) received his B.S. and M.S. in telecommunications engineering from the University of Zaragoza in 2003 and 2005, respectively. He is currently pursuing a Ph. D. degree in the Department of Electronic Engineering and Communications of the same university. His research interests focus on QoS in multimedia services and ambient intelligence.

LUIS CASADESUS (luis.casadesus@unizar.es) 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 QoS and quality of experience in multimedia services, like video streaming, videoconferencing, and networked online games.