Optimization of P2P-TV Traffic by Means of Header Compression and Multiplexing

Idelkys Quintana-Ramirez, Jose Saldana, Jose Ruiz-Mas, Luis Sequeira, Julian Fernandez-Navajas, Luis Casadesus
Communications Technology Group (GTC), Aragon Inst. of Engeneering Research (I3A)
Dpt. IEC. Ada Byron Building, EINA University of Zaragoza
50018, Zaragoza, Spain
Email: {idelkysq, jsaldana, jruiz, sequeira, navajas, luis.casadesus}@unizar.es

***Abstract*—This paper studies the optimization of the traffic of a P2P-TV application (SOPCast). First, a traffic characterization is deployed, and it is observed that the service generates a high rate of small UDP packet bursts between peers. Then, an optimization method based on header compression and multiplexing is used for sending together the packets with the same destination. Two multiplexing policies are defined and tested. The first one is based on a fixed multiplexing period, and the other one defines an inter-packet time threshold, with the aim of multiplexing together a whole traffic burst. Simulations using real traffic traces of SOPCast are performed in order to estimate the expected savings for both policies. The results show that the efficiency is improved, achieving uplink bandwidth savings between 26% and 33% for the period-based policy, and roughly 35% for the policy based on a threshold. The amount of packets per second is also reduced by a factor of 10 in both cases. As a counterpart, the addition of a small retention delay is necessary, but the tests show that it does not impair user's experience.**

***Keywords*- P2P-TV, SOPCast, header compression, multiplexing, traffic optimization.**

I. Introduction

The increasing success of real-time services (e.g., VoIP, online games, video streaming) and the growing community of end users who use this kind of multimedia services are modifying the "traffic mix" present on the Internet [1], since higher amounts of small packets are generated. P2P-TV is nowadays one of these emerging services, representing a very extended solution for delivering TV contents, since it allows the exchanging of audio and video with a large number of receivers simultaneously. The philosophy of the P2P model is to exploit the cooperation between users, so the clients (or peers) participate actively in content delivery, since they make available to their partners the data they receive. Thus, the resources and the costs are shared between the peers. A peer may become a broadcaster without the costs of earning a powerful server or having a huge amount of bandwidth. Thus, the P2P model may present a better scalability than client-server service models (e.g., the ones typically used by Content Delivery Networks, CDNs) [2].

However, P2P has become one of the greatest traffic-engineering challenges for Internet Service Providers (ISPs) because it is an important source of data flows [3]. P2P-TV applications are massively used by end users in residential networks with limited bandwidth and mid or low-end routers, which may represent a bottleneck [4]. Consequently, it is necessary to determine in more detail the P2P traffic behavior and to evaluate its impact on global network traffic. One of the most popular P2P-TV applications is SOPCast [5]. In [6] and [7] the authors studied its traffic, which is a mixture consisting of roughly 40% of large (video) and 60% of small (signaling) UDP packets. The high amount of small packets is caused by monitoring and control of video packets. These small packets present a very low efficiency, since a IP/UDP header is required for the transmission of some tens of bytes of payload.

With the aim of illustrating the packet exchange behavior between two peers, Fig.1 shows the packet size histogram for uploaded and downloaded packets from one peer, obtained from a real SOPCast trace of one match of the Champions League 2013, 30 $minutes$ long, and including $940,000$ packets. A clear distinction between packet size in the uplink and downlink can be appreciated: in the former, roughly 90% of the packets are small, since they are signaling packets generated to acknowledge big video downloaded packets. However, in the downlink the behavior is the opposite: the vast majority of packets are big because the include video information. This distribution corresponds to the typical behavior of a peer that is only downloading video at a certain moment: in the P2P structure, many peers are only data consumers during a large proportion of time [2].

This packet size distribution suggest that traffic optimization based on header compression and multiplexing can obtain significant savings in this scenario, since it is especially suited for long-term traffic flows of small packets. Multiplexing techniques have already shown their ability to achieve significant savings for other real-time services like VoIP [8] and online games [9]. The optimization can be implemented in different places: the SOPCast application itself, the routers between two peers, and it could even be implemented by network providers. As a counterpart, these techniques introduce an additional

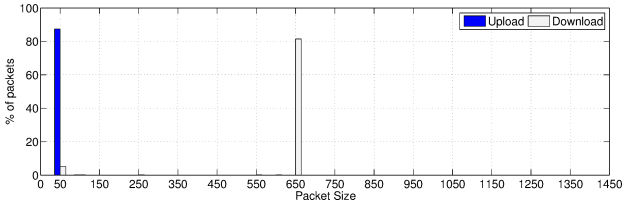


Fig. 1. Histogram of the size of SOPCast captured packets between 2 peers

delay caused by the retention of the packets at the multiplexer.

All in all, this work presents the analysis of the suitability of P2P-TV UDP traffic for being optimized, the definition of adequate multiplexing policies, and the calculation of the benefits obtained by means of these compressing and multiplexing techniques. On one hand, bandwidth saving will be beneficial for the usage of network resources, but on the other hand we will have to ensure that the added delays do not impair the user's perception of the application.

The rest of the work is organized as follows: Section II provides an overview of the related work. In Section III we describe the proposed compressing and multiplexing method. Section IV details the tests and results. Finally, the conclusions are drawn in Section V.

II. Related Work

A. P2P-TV Applications

In the last years, the interest on the analysis and characterization of P2P-TV is growing, and a number of research works have focused on the characterization of the behavior and traffic of different video streaming applications, e.g., PPLive, TVAnts, Coolstreaming, SOPCast, PPStream. The main motivations for these studies are the impact of this traffic in network communication [10], the improvements it brings to IPTV network operators [11], the Quality of Experience (*QoE*) provided [7], [12], alternative distribution algorithms [13] and its growing number of users [5].

Some studies have analyzed the different traffic patterns and the mechanisms of several P2P applications and their impact on network traffic, characterizing download and upload rates, packet types and the used protocols [6]. Other works are related to the characterization of the properties of the P2P overlay networks [14] and the implications of P2P-TV traffic for ISPs [3], [15]. However, few of the cited works are focused on the optimization of network resources and its effect on the traffic of other services sharing the Internet access.

B. SOPCast Behavior

Among these P2P-TV applications, SOPCast is becoming one of the most studied because of the increasing number of people that use it [5]. The study of its mechanisms, the analysis of its performance and the measurement of its *QoE* are important topics for researchers, operators and end users. It uses proprietary technologies and a closed communication protocol named sop or SoP technology [16]. In [2] and [17] the authors provide a detailed characterization of SOPCast, considering dynamic patterns and focusing particularly on inter-arrival times between packets of the flow exchanged between peers, the number of partnerships and the communication duration. In [6] and [7] the basic mechanisms of SOPCast were studied and the authors observed that it uses almost exclusively UDP for the transport of signaling and video traffic.

SOPCast uses a non-structured mesh-based overlay network, where the client requests the video chunks from one of its neighbors. This mechanism is resilient to failures (the peers may come and go to the overlay network at any moment) ensuring fewer errors in the video visualization. Traffic control is required to establish and maintain this network structure and comprises UDP packets with small payloads (less than $100\ bytes$) [7]. This control flow contributes substantially to the overall message exchange. Some works have proposed a three level classification of the peers sending content to a user's application. It has been observed that a small number of *super peers* (in the order of five) are responsible from the vast majority of the content delivered to the application (roughly 90%). *Ordinary peers* send some content, and finally *supplementary peers* only exchange signaling packets [2].

When this P2P-TV application is opened, it requires some time for searching peers and then it tries to download data from active neighbors. Next, the video stream is first stored in two buffers before being displayed. The former is the SOPCast application buffer, which produces a delay from the moment a channel is selected until the video streaming begins; the latter is the client buffer, adding a delay from the pop up of the media player to the beginning of the video playback. As a consequence, the total time required until the client can enjoy the live streaming ranges between 30 and $40\ seconds$ [7].

C. Traffic Optimization Methods

As we have seen, the high amount of small packets generated by SOPCast produces a significant network overhead, as it happens with the traffic of other multimedia services (e.g. VoIP and online games). In [9] and [18] the authors present a method called TCM (*Tunneling, Compressing and Multiplexing*), that saves bandwidth for the UDP traffic of online games. The headers are compressed using standard algorithms; a number of packets are multiplexed into a bigger one, and finally they are sent end-to-end using an L2TP tunnel. The results show that bandwidth savings of 38% can be obtained when IPv4 is used, at the cost of adding a new delay caused by the retention of the packets.

Regarding the header compression algorithm to be used for this P2P-TV traffic, we need one capable of compressing IP/UDP headers, so we can either use IPHC [19] or ROHCv2 [20], a more recent standard, which performs better in wireless networks, at the cost of a more complicated implementation. These methods use the high redundancy of the header fields in order to avoid sending some of them. They also use *delta* compression for reducing the number of bits of the fields with an incremental behavior (e.g., sequence numbers). They require the use of a *context*, which stores the values of the avoided fields in the ingress and egress of the tunnel.

Summing up, the optimization techniques presented in [9] can also be useful for P2P-TV applications, since this service also generates high rates of small packets. In this case we may be able to multiplex together a number of packets bound to the same peer (i.e., the small non-video packets), since the delay requirements are less severe than in other multimedia services (e.g., VoIP and online games). The increase of the efficiency of these flows can be especially beneficial for residential and aggregation networks, since on the one hand the achieved savings may lead to better utilization of network resources, and on the other hand the optimization of these resources may cause that peers contribute to video distribution [21].

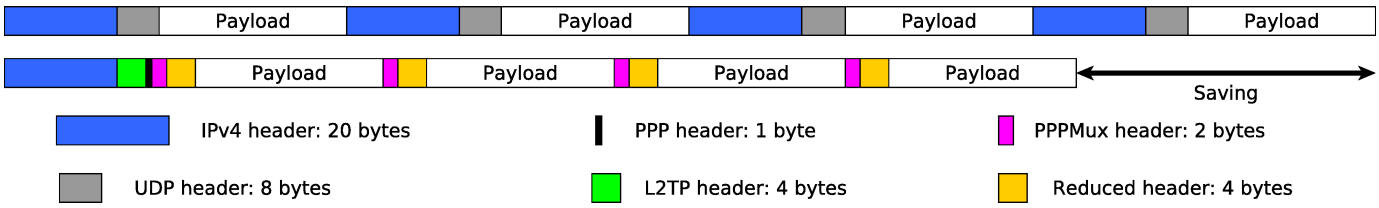


Fig. 2. Native and multiplexed traffic of SOPCast (*packet sizes are real scale*)

III. Compressing and Multiplexing Method

First, we need to select a protocol capable of compressing UDP/IP headers. We have selected IPHC because it is enough for our purposes, and it is easier to implement than ROHCv2. IPHC is able to compress UDP headers to 2 $bytes$, by using only 8 $bits$ for the CID (Context Identifier), and avoiding the optional checksum. Also, IPv4 header can be compressed to 2 $bytes$. So we will consider all headers as compressed (4 $bytes$), except for Full Headers that, according to IPHC [22], are sent every 5 $seconds$.

In order to illustrate the high overhead of the native traffic, and the bandwidth savings that can be obtained by means of header compression and multiplexing, Fig.2 presents (real scale) the traffic reduction achieved when four P2P-TV packets are multiplexed together.

Different policies can be used to select the packets to be included in each multiplexed one. We will use two multiplexing policies: *Fixed Period* and *Inter-Packet Time Threshold* (Fig.3(a) and Fig.3(b)). The packets generated by the application are referred as *native* in order to differentiate them from the multiplexed ones.

At the same time, we have to avoid the addition of delays that could harm the user's experience with the application. So we have to maintain the added delay under certain limits. Other multimedia flows, as VoIP or online games present very tight delay constraints. However, in the case of P2P-TV, this problem is less severe, since these applications store the downloaded content before reproducing it. In the case of SOPCast, the buffering is one minute long [5]. Thus, the major limitation here will be the MTU (Maximum Transfer Unit) of the multiplexed packets.

We will next explain separately the two multiplexing policies we have used.

A. Fixed Period

This method defines a *period*, and generates a multiplexed packet, including all the arrived ones, at the end of each interval (Fig.3(a)). There are three exceptions: if there is no packet to multiplex, nothing will be sent; if there is only one packet, it will be sent in its native form. Finally, if the MTU size is reached, a multiplexed packet is sent and a new *period* is started. Logically, it can be expected that an increase of the *period* will be beneficial for bandwidth saving (*BWS*) because the multiplexed packets will be larger. But it can not be increased indefinitely, because we may lose the contact with some peers.

B. Inter-Packet Time Threshold

SOPCast traffic characterization [17] reported that packets are generated following a bursty pattern. Thus, we will use this knowledge about the traffic characteristics (inter-packet time and packet size) in order to deploy a tailored multiplexing method (Fig.3(b)). For this aim, we rely on a state diagram with different possible events triggering the transitions between states (Fig.4). The method tries to locate a complete burst inside a multiplexed packet. If so, the number of multiplexed packets will be high without the need of significant extra delays. It works as follows: it remains in *waiting* state until the arrival of the first packet of a burst. When a packet arrives (*transition A*) the system goes to *storing* state, in which the subsequent arriving packets (*transition B*) are also stored. However, if the inter-packet time threshold is exceeded (*transition C*), or if the MTU size is reached, the system considers that the burst is finished, so a multiplexed packet is sent and the system returns to *waiting* state.

In order to adapt well to the traffic, a suitable value of the *threshold* has to be set, corresponding to the measured time between packets belonging to the same burst.

IV. Tests and Results

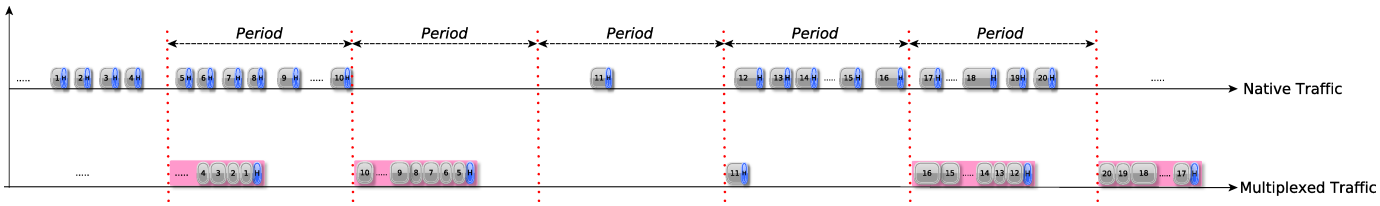
Real traces have been obtained with Tshark using a SOPCast client located in our campus network, during some Champions League 2013 matches. This machine has a public address; it runs Linux kernel $2.6.38 - 7$, and has a $Intel^{®}\ Core^{TM}\ i3\ CPU\ 2.4\ GHz$.

We isolated the traffic of the peer sending the highest amount of data to our node across the entire trace duration [6]. The initialization phase (contacting SOPCast Web server and the video selection steps) which generates TCP traffic were not used for the calculation of the results.

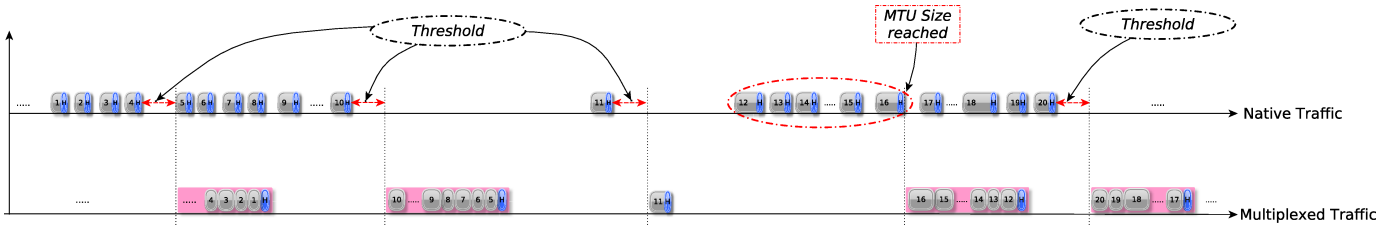
From the trace, we can notice that every video packet is confirmed with an application level acknowledge packet of 28 $bytes$ of payload (Fig.5); this situation generates a high amount of small packets with the same values in many header fields, which is interesting for header compression.

The SOPCast traces show that almost 84% of the packets have an inter-packet time between 10 and 50 ms. Thus, we have selected these values for the *period* and the *threshold* in the tests.

In order to confirm the size of the SOPCast buffer, reported as one minute in [5], we used Netem to filter the acknowledge packets sent by the local application to the peers. Since they stop receiving confirmations, they consider that the session has been disconnected, so video content is no longer received. However, the video stream is still played for a minute. This



(a) *Fixed Period*



(b) *Inter-Packet Time Threshold*

Fig. 3. Multiplexing Policies

result grants that the delays added by the multiplexing method (in the order of tens or hundreds of milliseconds) will not interrupt the communication with the peers.

For the obtaining of the bandwidth saving, MATLAB simulations have been run in order to obtain compressed and multiplexed traffic traces. We have focused the study on the upload traffic, since a high amount of small packets are present, and the most stringent limitation in residential networks is the upstream. To generate the traffic used in the tests, we extract the generation time and the packet size from the real trace. Then, header compression is applied to the flow and finally, times and sizes of multiplexed packets are calculated.

Fig.6 shows *BWS* percentages for both multiplexing policies. We can first observe that the obtained bandwidth saving is significant, between 25% and 35% of the global traffic going to the considered peer is saved in all cases. In *Inter-Packet Time Threshold* policy, a *BWS* between 33% and 35% is obtained. Inter-packet time is roughly $10\ ms$ for many packets of the trace, so selecting this value as the threshold does not produce the optimal savings. In this case, the number of multiplexed packets is small, since only one packet will be sent. However, for higher threshold values (from $12.5\ ms$) the *BWS* value is better, and it shows very little variation with different values of *threshold*. In *Fixed Period* policy, *BWS* is between 26% and 33%. The value of the saving presents an asymptotic behavior, similar to that obtained in [18].

Histograms of the size of multiplexed packet in both policies are shown in Fig.7, using a value of $20\ ms$ for the *period* and the *threshold*. If they are compared, we can see that *Fixed Period* policy generates a higher amount of small packets and fewer large packets than *Inter-Packet Time Threshold* policy. We can get a larger number of multiplexed packets using the latter, because it takes into account the inter-packet time and it can multiplex a higher number of consecutive packets, so some of them can reach the MTU size. That is why the *Inter-Packet Time Threshold* policy achieves better *BWS* values.

Fig.8 shows the number of packets per second (pps) generated with native and multiplexed traffic. There is a significant

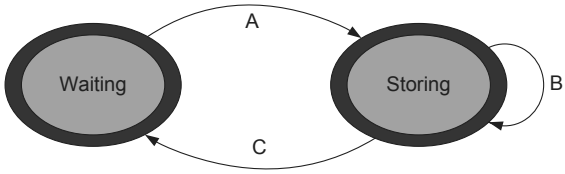


Fig. 4. State Diagram for Multiplexing by Inter-Packet Time Threshold

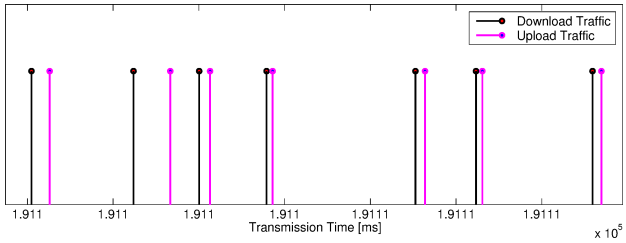


Fig. 5. Traffic during a typical flow between 2 peers (video and ACK packets)

reduction which are reduced from 50 to almost 5 pps, because of the high amount of multiplexed packets. For *Fixed Period* the obtained pps values decrease with the *period*. However, for *Inter-Packet Time Threshold* policy we can note that this result remains constant for *threshold* values above $12.5\ ms$.

All in all, it has been shown that multiplexing P2P-TV packets, even if a single peer is considered, can provide significant bandwidth savings and pps reductions. This can be especially interesting in residential networks in which uplink

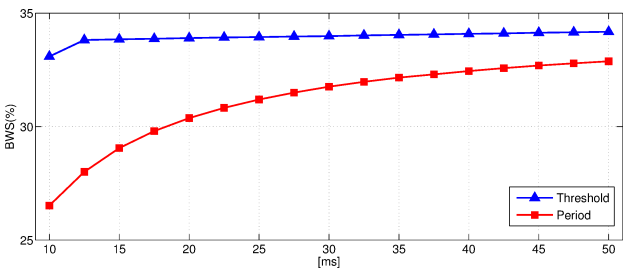
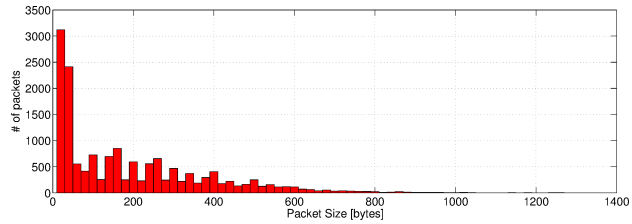
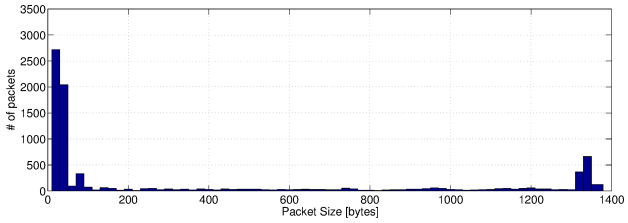


Fig. 6. Bandwidth Saving using the two multiplexing policies



(a) *Fixed Period*



(b) *Inter-Packet Time Threshold*

Fig. 7. Histogram of Multiplexed Traffic using the 2 multiplexing policies

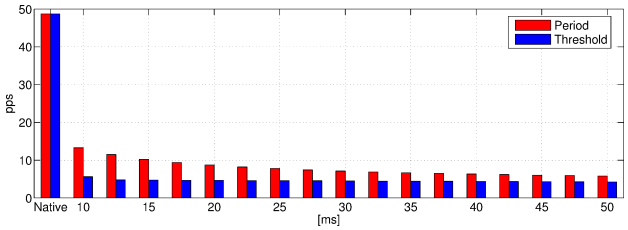


Fig. 8. Packets per second

bandwidth is limited and has to be shared with other real-time services. In addition, mid or low-end routers with a limited processing capacity are often used, so the reduction of the packets per second to be managed by the router is also interesting.

V. Conclusion

A multiplexing and header compressing method has been applied to the traffic generated by a popular P2P-TV application based on UDP. Taking into account the high amount of small packets generated, significant savings can be obtained in residential networks.

Simulations have been conducted in order to study the bandwidth savings obtained with two different multiplexing policies. The first policy is based on a *period*, and all packets that arrive in the interval are multiplexed and sent together. The second one is based on the definition of an inter-packet time threshold, with the aim of multiplexing together all the packets belonging to the same burst. The results show that the proposed method can produce significant savings: the uplink bandwidth can be reduced in the range of 26% to 33% for the policy based on a *period* and between 33% to 35% if an inter-packet time threshold is employed. An important reduction in the amount of packets per second can also be observed, and it has been shown that the added multiplexing delay would not harm user's experience.

Acknowledgment

This work has been partially financed by the European Social Fund in collaboration with the Government of Aragon, CPUFLIPI Project (MICINN TIN2010-17298), Ibercaja Obra Social, Project of Cátedra Telefónica, University of Zaragoza, Banco Santander and Fundación Carolina.

References

[1] C. Park, F. Hernandez-Campos, J. Marron, and F. D. Smith, "Long-range dependence in a changing internet traffic mix," *Computer Networks*, vol. 48, no. 3, pp. 401 – 422, 2005.

[2] P. Eittenberger, U. R. Krieger, and N. M. Markovich, "Measurement and analysis of live-streamed p2ptv traffic," in *Performance Modelling and Evaluation of Heterogeneous Networks in HET-NETs 2010*, T. Czachórski, Ed. IITiS, January 2010, pp. 195–212.

[3] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, ser. SIGCOMM '08. ACM, 2008, pp. 363–374.

[4] L. Sequeira, J. Fernandez-Navajas, J. Saldana, and L. Casadesus, "Empirically characterizing the buffer behaviour of real devices," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on*, 2012, pp. 1–6.

[5] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will iptv ride the peer-to-peer stream?" *Communications Magazine, IEEE*, vol. 45, no. 6, pp. 86–92, 2007.

[6] T. Silverston and O. Fourmaux, "Measuring p2p iptv systems," in *in Proc. of NOSSDAV07, International Workshop on Network and Operating Systems Support for Digital Audio & Video*, 2007.

[7] B. Fallica, Y. Lu, F. Kuipers, R. Kooij, and P. V. Mieghem, "On the quality of experience of sopcast," in *Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST'08. The Second International Conference on*, September 2008, pp. 501–506.

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

[9] J. M. Saldana, J. Fernandez-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.

[10] S. Tang, Y. Lu, J. M. Hernndez, F. A. Kuipers, and P. V. Mieghem, "Topology dynamics in a p2ptv network." in *Networking*, ser. Lecture Notes in Computer Science, L. Fratta, H. Schulzrinne, Y. Takahashi, and O. Spaniol, Eds., vol. 5550. Springer, 2009, pp. 326–337.

[11] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft, "On next-generation telco-managed P2P TV architectures," in *IPTPS '08*, 2008.

[12] U. R. Krieger and R. Schwessinger, "Analysis and quality assessment of peer-to-peer iptv systems," *Consumer Electronics 2008 ISCE 2008 IEEE International Symposium on*, pp. 1–4, 2008.

[13] Y. Liu, Y. Guo, and C. Liang, *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, March 2008.

[14] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Measurement and modeling of a large-scale overlay for multimedia streaming," in *The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness; Workshops*, ser. QSHINE '07. ACM, 2007, pp. 3:1–3:7.

[15] D. Moltchanov, Y. Koucheryavy, and B. Moltchanov, "The effect of biased choice of peers on quality provided by p2p file sharing." in *CCNC*. IEEE, 2012, pp. 608–613.

[16] http://www.sopcast.com/.

[17] A. B. Vieira, P. Gomes, J. A. M. Nacif, R. Mantini, J. M. Almeida, and S. V. A. Campos, "Characterizing sopcast client behavior." *Computer Communications*, vol. 35, no. 8, pp. 1004–1016, 2012.

[18] J. Saldana, L. Sequeira, J. Fernandez-Navajas, and J. Ruiz-Mas, "Traffic optimization for tcp-based massive multiplayer online games," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on*, July, pp. 1–8.

[19] M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression," RFC 2507, February 1999.

[20] G. Pelletier and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite," RFC 5225, April 2008.

[21] L. Sequeira, I. Quintana, J. Saldana, L. Casadesus, J. Fernández-Navajas, and J. Ruiz-Mas, "The utility of characterizing the buffer of network devices in order to improve real-time interactive services," in *Proceedings of the 7th Latin American Networking Conference*, ser. LANC '12. ACM, 2012, pp. 19–27.

[22] V. Jacobson. (1990, Feb.) RFC1144: Compressing TCP/IP headers for low-speed serial links.