Small-Packet Flows in Software Defined Networks: Traffic Profile Optimization

Jose Saldana, David de Hoz, Julián Fernández-Navajas, José Ruiz-Mas
I3A, University of Zaragoza
Ada Byron Building, 50018, Zaragoza, Spain
Email: {jsaldana, dhoz, navajas, jruiz}@unizar.es

Fernando Pascual, Diego R. Lopez, David Florez, Juan A. Castell, Manuel Nuñez
Telefonica I+D
Distrito T, Sur 3, Ronda de la Comunicación s/n, 28050 Madrid, Spain
Email: {fernando.pascualblanco, diego.r.lopez, david.florezrodriguez, juanantonio.castelllucia, manuel.nunezsanz}@telefonica.com

Abstract— **This paper proposes a method for optimizing bandwidth usage in Software Defined Networks (SDNs) based on OpenFlow. Flows of small packets presenting a high overhead, as the ones generated by emerging services, can be identified by the SDN controller, in order to remove header fields that are common to any packet in the flow, only during their way through the SDN. At the same time, several packets can be multiplexed together in the same frame, thus reducing the overall number of frames. The method can be useful for providing QoS while the packets are traversing the SDN. Four kinds of small-packet traffic flows are considered (VoIP, UDP and TCP-based online games, and ACKs from TCP flows). Both IPv4 and IPv6 are studied, and significant bandwidth savings (up to 68 % for IPv4 and 78 % for IPv6) can be obtained for the considered kinds of traffic. The optimization method is also applied to different public Internet traffic traces, and significant reductions in terms of packets per second are achieved. Results show that bandwidth consumption is also reduced, especially in those traces where the percentage of small packets is high. Regarding the effect on QoS, the additional delay can be kept very low (below 1 millisecond) when the throughput is high, but it may become significant for low-throughput scenarios. Thus, a trade-off between bandwidth saving and additional delay appears in those cases.**

Index Terms— **Software Defined Networks, multiplexing, traffic optimization, compression, small-packet services**

I. Introduction

Software Defined Networks (SDNs) are a new approach to networking, based on the radical separation of the control and data planes, connected by open interfaces, and including the direct programmability of the control plane. This allows for a logically centralized control of the network as a whole, bringing the possibility of dealing with the network as a single and programmable entity. This is especially interesting in current highly virtualized environments, most notably in cloud computing, because it allows managing network resources in a much more flexible and efficient way, making the network able to provide a QoS level adequate to the nature of each flow. The SDN paradigm is not only used in wired networks, but it has also been proposed for adding programmability to wireless solutions: in [2] a software defined Wi-Fi network architecture was proposed and tested, where a number of Wi-Fi Access Points were integrated in a programmable way; in [3], SDNs were also employed for deploying and managing an open public wireless network in the UK. OpenFlow [4] is the most extended and consolidated standard for SDNs.

At the same time, emerging real-time services (e.g. VoIP, online gaming) are becoming more and more widespread on the Internet, and they are not only used in desktop computers, but also in wireless scenarios. Their interactivity requirements make them send high rates of small packets (average payloads of tens of bytes). These traffic patterns are sometimes known as "small-packet flows" [5], in order to stress their low efficiency in terms of payload-header ratio. As an example, an RTP VoIP packet carrying two samples of the G729a codec requires 40 bytes of headers to carry 20 bytes of voice information.

Many online games, as e.g. First Person Shooters (FPSs) employ a similar traffic pattern, i.e. a high rate of UDP packets is sent (RTP is not used for this service) from the client to the server and vice versa. The problem is similar to that in VoIP: latency is very harmful for interactive game playing, since it may cause inconsistencies between the statuses of the virtual world observed by the different players. The size of the payload is typically of tens of bytes, especially in the client-server direction, since only the actions of a player are sent to the server. In the opposite direction, packets are bigger, since they include information of the rest of the players. These

This research has been partially supported by the EC H2020 Project "What to do With the Wi-Fi Wild West" (Wi-5), grant agreement no: 644262; Project TAMA, Government of Aragon; European Social Fund. Corresponding author: Jose Saldana.

A preliminary version of this article appeared in [1]. This version includes new tests, especially those performed using public Internet traces (see Section V.B).

games also send high rates of packets (inter-packet time can be about 25 or 50 milliseconds) [6].

In addition, services using TCP also generate large amounts of ACK packets without payload. These can also be considered as “small-packet flows,” with 40 bytes of header and no payload. For example, when a file is downloaded, a flow of ACKs is sent to the origin of the communication. For example, a 3 Mbps file download using packets of 1,500 bytes, may generate 125 ACKs per second, using the typical TCP parameters (an ACK sent every 2 downstream packets).

This is also true for certain online game genres, as Massively Multiplayer Online Role Playing Games (MMORPGs), which have become very popular in the last years, with a special significance in Asia [7]. Many of these games employ TCP flows for communicating the actions of the player to the server, and for sending updates of the game status to the client. So a bidirectional TCP connection is established, piggybacking ACKs into normal packets. Nevertheless, up to 56 % of the sent packets are still *pure* ACKs, i.e. they have no payload, as reported in [8].

These high rates of tiny packets translate into an inefficient usage of network resources, so there is a need for mechanisms able to reduce the overhead introduced by these low-efficiency flows. Bandwidth savings are interesting for network operators, since they may alleviate the traffic load in their networks.

In addition, the reduction of the amount of packets per second in the network is desirable for two main reasons: first, network equipment has a limitation in terms of the number of packets per second it can manage [9], i.e. many devices are not able to send small packets back to back due to processing delay; second, a lower amount of packets per second will reduce energy consumption in network equipment since, according to [10], internal packet processing engines and switching fabric require 60% and 18% of the power consumption of high-end routers respectively.

Thus, reducing the number of packets to be managed and switched will reduce the overall energy consumption. The measurements deployed in [11] on commercial routers corroborate this: a study using different packet sizes was presented, and the tests with big packets showed that energy consumption gets reduced, since a non-negligible amount of energy is associated to header processing tasks, and not only to the sending of the packet itself.

Aggregation is an option considered in wireless protocols: two different methods for multiplexing a number of frames together are included in 802.11n and subsequent versions [12]. They are especially useful when small frames are to be transported, since aggregation reduces the number of transmissions, thus increasing the airtime efficiency [13].

Header compression techniques capable to save bandwidth for long-term flows using small packets through the public Internet were developed long ago [14]. They are based on the fact that many header fields (known as NOCHANGE fields) remain the same for every packet in a flow. They also reduce the number of bits of increasing fields (e.g. sequence numbers), by sending the difference with the previous value (DELTA). To achieve this, they need to define a *context*, i.e. a set of variables synchronized between the sender and the receiver; and a *context identifier* has to be added to every packet, in order to allow the reconstruction of the received packet. The desynchronization between sender and receiver may result in a burst of erroneous packets.

The compression and decompression of the headers implies additional processing in the nodes. In addition, the most recent header compression techniques [15] provide a more robust synchronization between the sender and the receiver, including a set of advanced features that imply a higher computational cost [14].

Furthermore, header compression presents another limitation: compressed packets can only traverse a single Layer-3 hop, since they do not include a standard header. One solution is to compress and decompress them at each intermediate node. Another option is to use an end-to-end tunnel, so as to avoid the additional processing caused by compression and decompression, but in this case the tunneling overhead cancels the savings obtained by header compression.

A solution proposed in [16] is jointly to use multiplexing, header compression and tunneling. Thus, a number of header-compressed packets belonging to different flows can be multiplexed together in the same frame, to share the tunnel overhead, which becomes relatively smaller as the number of packets multiplexed in the same frame grows. This combination allows the packets to travel end-to-end through a public network while maintaining a good header reduction rate; as an additional benefit, the amount of packets per second traversing the intermediate nodes is significantly reduced, by a factor equivalent to the average number of multiplexed packets.

In order to pack a number of packets to be sent together, a multiplexing period *PE* can be defined in the device performing the traffic optimization process. A multiplexed frame is released at the end of the period, so the longer the period, the higher the number of multiplexed packets and the higher the savings. However, a tradeoff appears, since this multiplexing latency has to be maintained under a threshold in order to assure the delay requirements of the service.

This reduction in the header-to-payload rate of the small-packet flows is also desirable in SDNs. In this context, the contribution of the present paper is the proposal of an optimization method for providing significant bandwidth savings in an Openflow-based SDN, with three main advantages: *i)* the tunneling layer is not necessary, since the SDN provides it in a natural way; *ii)* the avoidance of the use of standard header compression techniques [14] which require a context synchronization between the sender and the receiver; and *iii)* multiplexing reduces the number of frames, so a number of Ethernet fields (header, inter-frame gap) are sent only once instead of being repeated for each packet. Four kinds of small-packet traffic flows will be considered (VoIP, UDP and TCP-based online games, and ACKs from TCP flows), and analyses using public Internet traces will also be presented.

The remainder of the article is as follows: the next section summarizes the Related Work. The proposed method is described in detail in Section III. The expected savings are analytically obtained in Section IV; the tests and results are presented in Section V and the paper ends with the Conclusions and Future Work Section.

II. Related Work

The combination of multiplexing and compression was first proposed in [16], by the joint use of Enhanced Compressed RTP [17], PPPMux [18] and L2TPv3 [19] protocols, with the aim of reducing the overhead of VoIP flows using RTP. In [20] another multiplexing method for VoIP was proposed, with the idea of maintaining a good quality level. The extension of [16] for other real-time services not based on RTP has also been proposed recently [21], taking into account that some applications (e.g. certain online game genres) present a traffic profile consisting of high rates of small UDP packets [9], very similar to that of VoIP. However, an end-to-end tunnel is required for sending compressed packets through the public Internet.

The use of header compression techniques within an OpenFlow SDN was first proposed in [22], with the aim of reducing overhead, and saving the compress-decompress delay at each hop. The controller would play the role of establishing the end-to-end tunnel. In order to avoid decompressing at each hop, L2 information was used to route the packet. By means of Openflow, packets compressed with standard techniques were still able to be correctly routed.

As remarked in [22], focused on IPv6 extension headers, Openflow 1.1 introduced the possibility of extensible matches, actions, messages and errors, thus allowing two controllers or switches to agree on different syntaxes when matching flows. Thus, different sets of fields can be selected for matching a flow. This feature is interesting, since additional fields can be included in the *tuple* that Openflow uses for defining a flow, and the value of these fields will be stored in the controller.

The effect of the required additional delay has also been explored: in real-time services it may have an influence on subjective quality. In [23] this effect was explored with VoIP traffic; in [24] a subjective quality estimator for a UDP-based game was used; and in [25] the effect on a TCP-based game was studied. In addition, if a flow of TCP ACKs is multiplexed, the additional delay may have an impact on TCP dynamics, taking into account that this protocol is RTT-based. In [26] the effect of this delay on TCP was explored, showing the conditions in which the throughput obtained by multiplexed flows may be penalized.

The method proposed in the present paper is able to significantly reduce the overhead in an Openflow-based SDN. In a similar way to [22], it multiplexes a number of packets, but it does not rely on standard header compression techniques based on context synchronization [14] as, e.g. IPHC or ROHC. We rely on the fact that the controller in an SDN stores the information about the fields that remain constant for all the packets in a flow. Thus, the difference with [22] is that the proposed method removes the fields that are the same for every packet on a flow, but these fields are not part of a "context" but are stored in the controller, as part of the network global state. In addition, our method does not require a tunneling protocol, since the SDN itself is able to provide it.

III. Proposed Optimization Method

In this section the method for optimizing the traffic is presented. We will use the word "optimized" when referring to compressed packets. Three steps are considered, which are explained subsequently.

A. Removing Header Fields Present in the Openflow Tuple

Under Openflow 1.0 [4] all the switches in a management domain are connected to a central controller, and each packet is associated to a flow by means of a 12-field *tuple* (Fig. 1), which is used for assigning the output port at each switch consequently.

When a flow traverses a path within an OpenFlow SDN, the IP and TCP *tuple* fields of all the packets are the same for all the tables of the switches of that path, and also in the controller. Thus, the IP and TCP protocol fields already included in the *tuple* are not needed for switching decisions but only for matching the packets with a flow. Thus, if we remove these fields and we substitute them by a *flow identifier* (FID), the packet can then travel in an optimized manner within the SDN. A new value of the *protocol* field of PPP could be defined, in order to flag the packet as optimized (Fig. 2): it begins with the FID, plus the compressed IP and TCP headers (i.e. the fields not present in the *tuple*), and the payload.

This would have some similarities with MPLS or other technologies in which labels are used for identifying a flow across the network. However, the idea of a generic FID would make our solution agnostic of the underlying technology. Another advantage with respect to those technologies is that in an SDN the controller has an overall view of the network, so the ingress and egress points of the tunnel can be dynamically defined according to traffic requirements.

An additional advantage of this proposal is that the FID could be linked to QoS identifiers used in other networks (e.g. MPLS label, IPv6 flow label, ATM Virtual Path Identifier VPI and Virtual Channel Identifier VCI), thus providing quality levels to the packets traversing the SDN, allowing horizontal QoS mapping over heterogeneous networks [27], [28].

The authors of [22] proposed an end-to-end header compression scheme in an SDN context. However, our proposal does not formally use standard header compression techniques [14], but it only removes NOCHANGE fields. The compression of DELTA fields is not considered, since it would only provide a marginal increase of the savings, at the cost of some additional

In port	Ethernet	VLAN	IP	TCP
	SA, DA, type	ID, prio	SA,DA,Prot,ToS	Sport, DPort

Figure 1. Tuple of Openflow 1.0.

complexity. For example, the number of bytes required by DELTA fields may vary between packets, so compressed headers with a variable size would appear. In addition, more processing power would be required for the compression and decompression of these fields, since the value of a DELTA field not only depends on the actual value included in the current packet, but in the "context", which depends on the previous ones. Therefore, when a packet is lost, the potential context desynchronization can be translated into a burst of bad packets. The marginal savings to be provided by the use of DELTA compression would introduce these potential problems, so we have decided not to consider this feature.

With Openflow 1.0, the skipped fields account for 13 bytes per packet for IPv4/TCP. Considering a 3-byte FID, 10 bytes per packet can be saved, i.e. 25 % of the header, which may imply a significant bandwidth reduction for services using small packets.

B. Removing Other Fields

But Openflow 1.1 and subsequent versions also allow switches and controllers to agree on different flow matching syntaxes, in order to avoid a too rigid match structure [29]. Taking advantage of this fact, we consider, as a second step of our proposal, the inclusion in the *tuple* of other NOCHANGE fields of Transport and Network layers. Although these fields are not strictly required for identifying a flow, including them in the *tuple* would make it possible to remove them from all the packets, thus allowing even higher header compression ratios. As a counterpart, we can expect a slight increase of the storage requirements of the ingress and egress switches and the controller, but it would only mean 40 bytes per flow. Furthermore, fields belonging to well-known application layer protocols can also be included in the *tuple*. As an example, RTP is often used for services based on small packets (VoIP), so removing RTP fields with a constant value will be translated into significant savings for these flows.

C. Multiplexing a number of packets in a single frame

Finally, taking advantage of its programmability, the SDN controller could be able to identify flows sharing a common path segment within the SDN. In this case, packets belonging to different flows could be multiplexed together and sent as a single Eth frame (Fig. 3) in all the hops of the path. This would require the use of a multiplexing protocol between the ingress and egress switches of the common path. PPPMux [18] can be used for multiplexing.

Finally, the egress switch will use the information stored in the controller in order to get the value of the original fields corresponding to each flow (using the FID). Thus, it will be able to rebuild the packets to their native form and send them as non-compressed individual frames.

IV. CALCULATION OF THE EXPECTED SAVINGS

In this section, we present the bandwidth savings that can be obtained using this traffic optimization method. The savings are measured as the difference between the number of bytes required at Eth level when using the optimization method with respect to the native Openflow protocol. They are obtained as a function of the number of multiplexed packets *N*. We have to consider the Eth Inter-frame gap in the calculations, since it also limits the throughput of the network.

Since the time for sending the compressed and the native traffic is the same, we can define Bandwidth Savings *(BS)* as:

$$BS = 1 - \frac{Bandwidth_{optimized}}{Bandwidth_{native}} = \\ = 1 - \frac{Bytes_{optimized}}{Bytes_{native}} \qquad (1)$$

$Bytes_{native}$ is defined as (see Fig. 3) the sum of the sizes of the Eth header *(E)*, the native network and transport headers *(NH)*, the expected size of the payload ($E[P]$), and the Eth trailer *(F)* and inter-frame gap *(G)*:

$$Bytes_{native} = N \bullet (E+NH+E[P]+F+G) \qquad (2)$$

And the expected size of the multiplexed packet will be the sum of:

- Ethernet header *(E)*.
- Common header: The PPP headers *(PH)*.
- Multiplex header: The size of the PPPMux separator included at the beginning of each multiplexed packet *(N•M)*.
- The *flow identifier* of each packet *(N • FID)*.
- The compressed Network and Transport level headers *(N • CH)*.
- The payload of each packet ($N \bullet E[P]$).
- The Ethernet trailer *(F)*.
- The inter-frame gap *(G)*.

$$Bytes_{optimized} = \\ E+PH+N \bullet (M+FID+CH+E[P]) + F+G \qquad (3)$$

If we substitute (3) and (2) in (1), we obtain the bandwidth savings, which can be separated into a fixed and a variable term (which depends on the number of multiplexed packets). The fixed term, which is the asymptote of the bandwidth savings, can be expressed as:

$$1 - \frac{M + FID + CH + E[P]}{E + NH + E[P] + F + G} \qquad (4)$$

And the term which depends on the number of packets, giving us an idea of how the common header is shared between the multiplexed packets, is:

$$-\frac{1}{N}\frac{E + PH + F + G}{E + NH + E[P] + F + G} \qquad (5)$$

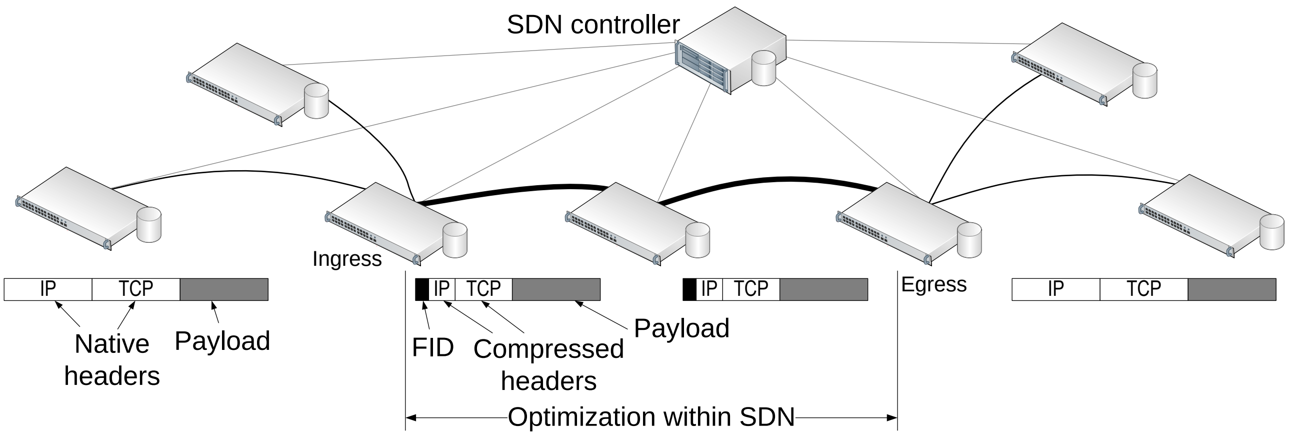


Figure 2. Scheme of the header compression within the SDN.

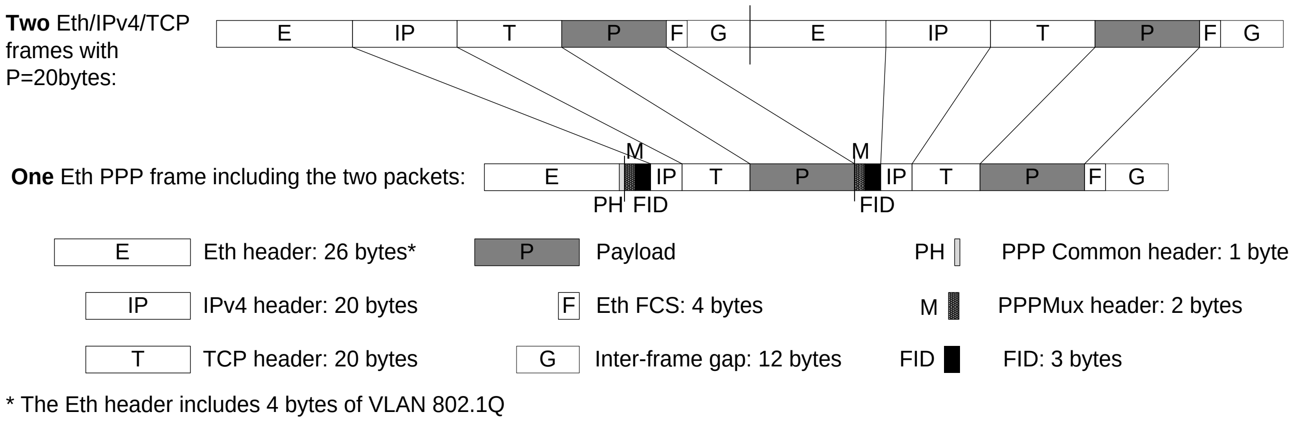

Two Eth/IPv4/TCP frames with P=20bytes:
E
IP
T
P
F
G
M
One Eth PPP frame including the two packets:
PH
FID
Eth header: 26 bytes*
IPv4 header: 20 bytes
TCP header: 20 bytes
Payload
Eth FCS: 4 bytes
Inter-frame gap: 12 bytes
PPP Common header: 1 byte
PPPMux header: 2 bytes
FID: 3 bytes
* The Eth header includes 4 bytes of VLAN 802.1Q

Figure 3. Scheme of two frames multiplexed together (real scale).

Regarding the reduction in the amount of packets per second, the results are similar to those reported in [21], i.e. a reduction by a factor of *N*.

V. TESTS AND RESULTS

This section is divided into the following parts: first, four different traffic patterns of small packets are tested, in order to calculate the savings expected by the use of the proposed method; second, different traffic traces publicly available are tested, in order to estimate the expected savings that could be obtained when using the traffic optimization method in a SDN. Finally, the effect of the additional delays is discussed.

A. Small-packet patterns

In order to evaluate the performance gains of this approach, four different traffic patterns have been selected:

a) VoIP using IP/UDP/RTP (40 bytes header for IPv4 and 60 for IPv6) and G.729 codec with 2 samples per packet (20 bytes payload) every 20 ms.

b) Client-to-server flows of a UDP-based online game [9] (28 or 48 bytes header), with 24.65 packets per second, and an average payload of 41.09 bytes.

c) Client-to-server flows of a TCP-based online game [8] (40 or 60 bytes header) of 9.51 packets per second with an average payload of 8.74 bytes.

d) IPv4/TCP ACKs of 40 or 60 bytes.

Table I enumerates the fields that present a static behavior for the considered traffic patterns, and can be considered as NOCHANGE. Other fields may also be selected depending on the application and the service (e.g. in VoIP or TCP ACKs using IPv6, the *Payload Length* field could also be avoided, since it is fixed).

The value of the asymptote (4) for the different traffic patterns is shown in Table II. As a consequence of the compression of the headers and multiplexing, which reduces the total amount of Eth frames, up to 72 % of the bandwidth can be saved if IPv4 is used. When using IPv6, this figure rises up to 81 %. The savings for all the flows are above 50 %. The ACKs flow is the one that obtains the best savings, due to the absence of payload. In the case of the UDP-based game, the header-to-payload ratio is the lowest, so it is the pattern which shows the lowest savings.

The variable term (5) reports the number of packets required for obtaining significant savings. In order to study its influence, we have built Fig. 4. It can be seen that high values of bandwidth savings are obtained not only when 20 packets are multiplexed, but we also obtain similar results for smaller numbers of packets (e.g. 10 packets); and even with 2 packets we can still save 40 % of bandwidth in some cases. These savings are even more significant if IPv6 is used (Fig. 4 b), since the overhead of this protocol is higher than that of its predecessor. In this case, bandwidth savings can reach 78 %.

B. Public Traffic Traces

In this subsection we present a series of tests with the aim of estimating the savings which can be obtained when applying the proposed optimization method in typical network scenarios. For that aim, we have used some publicly available traffic traces, and applied the method to them. We have taken an approach based on establishing a multiplexing period, and sending a multiplexed packet at the end of each interval. If the MTU size is reached before the end of the period, the sending is triggered and a new period begins (Fig. 5). A multiplexed packet including a number of compressed ones is sent at the end of each period.

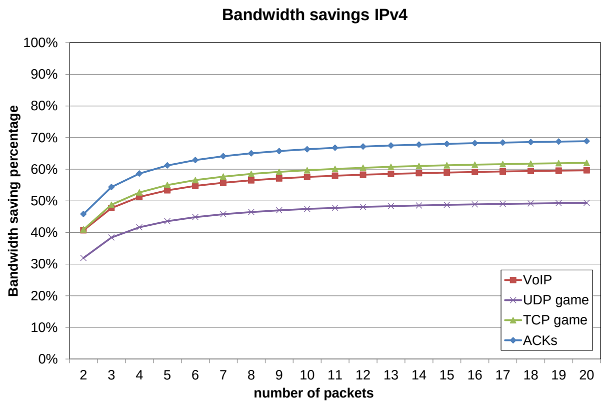
Taking into account that multiplexing achieves high gains when applied to small packets, a size limit can be established in the ingress switch (see Fig. 6): packets above the limit are sent to a queue, whereas those under the limit are sent to another one, where they are optimized (header compression and multiplexing) before being sent.

TABLE I. FIELDS CONSIDERED AS NOCHANGE FOR THE STUDIED PATTERNS

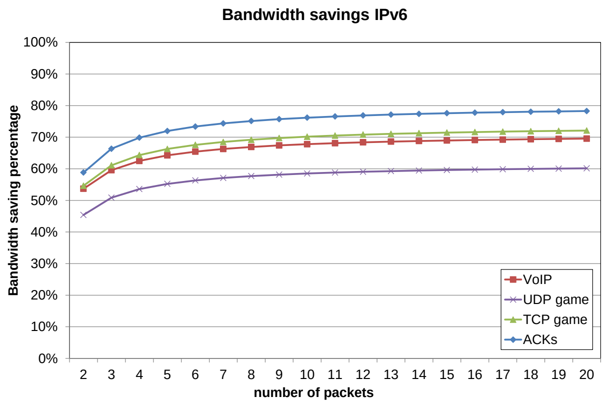
IPv4	IPv6	TCP/UDP	RTP
Version	Version	Source Port	Version
IHL	Traffic Class	Dest. Port	P
DSCP	Flow Label	Data Offset	X
ECN	Next Header	Reserved	CC
Time To Live	Hop Limit	Urgent Pointer	M
Protocol	Source Address		PT
Source Address	Dest. Address		SSRC id
Dest. Address			

TABLE II. ASYMPTOTIC SAVINGS FOR THE STUDIED PATTERNS

	VoIP	UDP game	TCP game	TCP ACKs
IPv4	62.75 %	52.21 %	65.02 %	72.62 %
IPv6	72.13 %	62.55 %	74.95 %	81.37 %



(a)



(b)

Figure 4. Bandwidth savings for each pattern a) using IPv4; b) using IPv6.

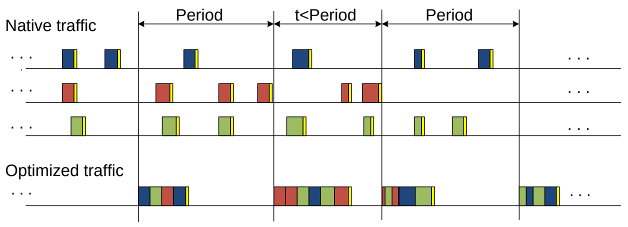


Figure 5. Multiplexing method scheme.

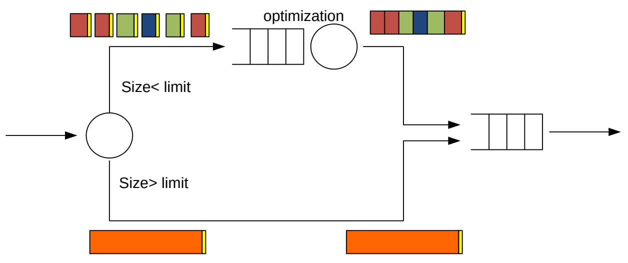


Figure 6. Scheme of the queues in the ingress device.

Four public traffic traces have been used, which main characteristics are summarized in Table III. They have been selected in order to consider scenarios with very different throughput levels (only the first 200,000 packets of each trace have been used for the tests):

a) *Chicago:* A trace obtained from CAIDA's passive monitor *'equinix-chicago'* [30], with 1.5 Gbps of traffic.

b) *DSL-uplink:* Two hundred ADSL customers, mostly student dorms, are connected to an access network [31]. The resulting throughput is 125 Mbps.

c) *Education-downlink:* A downlink trace obtained from a 100 Mbps Ethernet link connecting an educational organization (around 35 employees and a little over 100 students) to the Internet [31] [1]. The average throughput is only 107 kbps.

d) *Education-uplink:* The corresponding uplink part of trace c) [2]. The average throughput is even lower: just 13 kbps.

These traces have been exported to Matlab, where simulations implementing the optimization method have been performed. The Matlab script first separates the traffic according to its size, and then implements the multiplexing and compressing method, producing an output file with the sizes and departure times of the optimized frames.

Fig. 7 to 10 represent the obtained packet size histograms of each of the studied traffic traces. Please note that a logarithmic scale has been used for the Y axis. A peak corresponding to very low sizes can be observed in all the traces, which represents TCP ACKs and other small-packet flows. Other peak corresponding to near-to-MTU packets is always present. In the two *uplink* traces, the number of small packets is higher than the number of MTU ones: this corresponds to the fact that people tend to download files (big packets) and this generates high rates of ACKs (small packets) in the uplink.

The optimization method has been applied to the traces, using a multiplexing period *(PE)* of 100 ms, and different values for the limit used for defining a packet as "small". Fig. 11 to 14 represent the packet size histograms of the optimized flows when the size limit is set to 1000 bytes; Fig. 15 presents the reduction in terms of packets per second of each trace, as a function of the size limit; finally, Fig. 16 reports the bandwidth savings, obtained using equation (1).

TABLE III. CHARACTERISTICS OF THE USED TRACES

	Chicago	DSL uplink	Education downlink	Education uplink
Duration	0.783 sec	7.081 sec	14 882 sec	18 364 sec
Number of packets	200 000	200 000	200 000	200 000
Throughput	1 547 Mbps	125 Mbps	107 kbps	13 kbps
Packets per second	255 332 pps	28 245 pps	13.43 pps	10.8 pps
Average packet size	757.7 bytes	553.7 bytes	991.6 bytes	150.9 bytes

[1] The trace has been filtered with Wireshark using 'ip.src!=192.168.0.0/16' for obtaining the downlink packets.
[2] The filter in this case is 'ip.dst!=192.168.0.0/16'

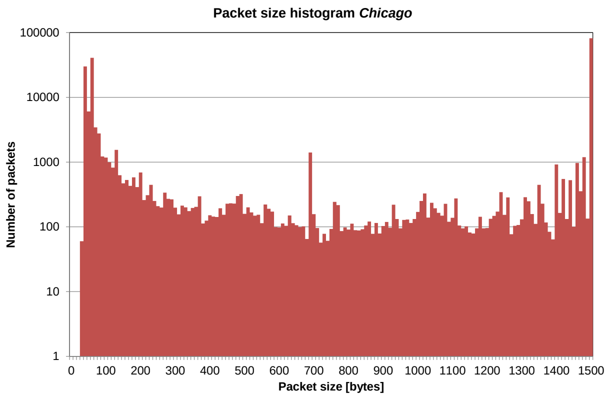


Figure 7. Packet size histogram of the *Chicago* trace.

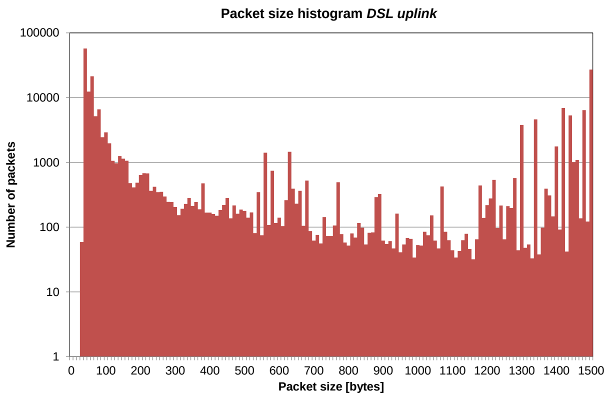


Figure 8. Packet size histogram of the *DSL_uplink* trace.

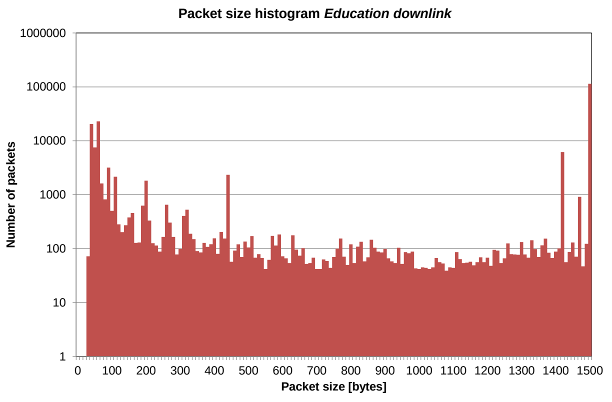


Figure 9. Packet size histogram of the *education_downlink* trace.

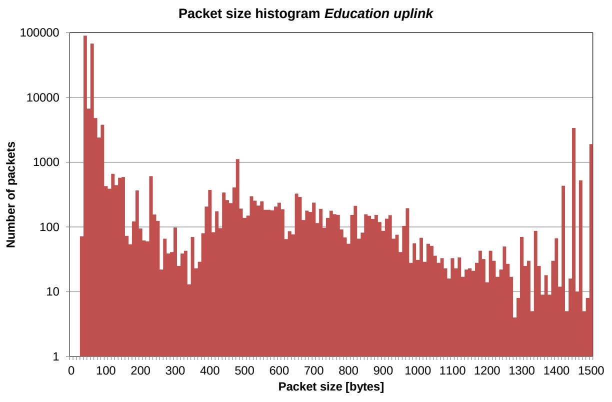


Figure 10. Packet size histogram of the *education_uplink* trace.

© 2015 ACADEMY PUBLISHER

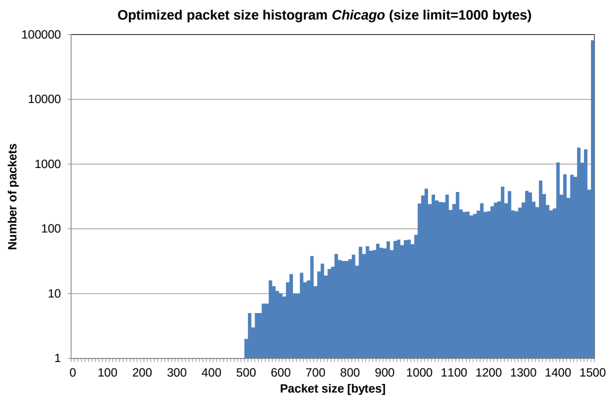


Figure 11. Packet size histogram of the optimized *Chicago* trace.

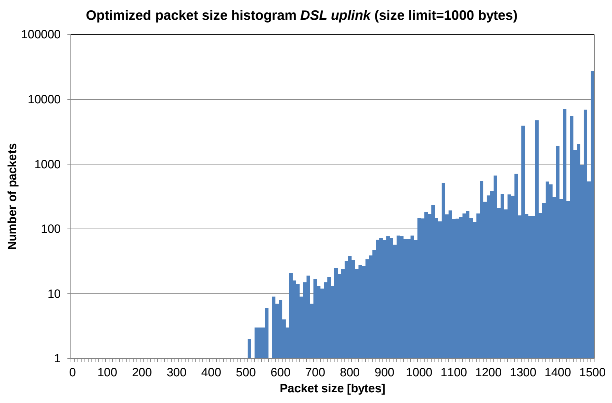


Figure 12. Packet size histogram of the optimized *DSL_uplink* trace.

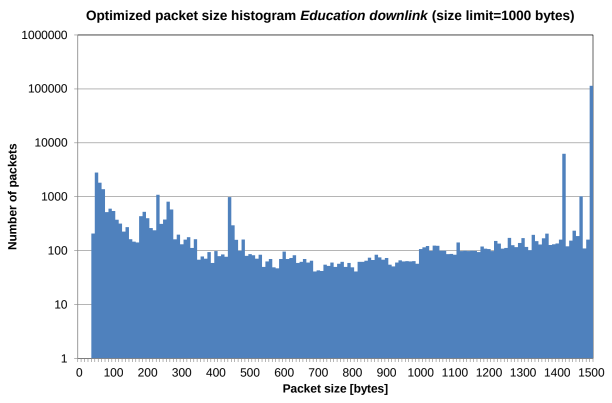


Figure 13. Size histogram of the optimized *education_downlink* trace.

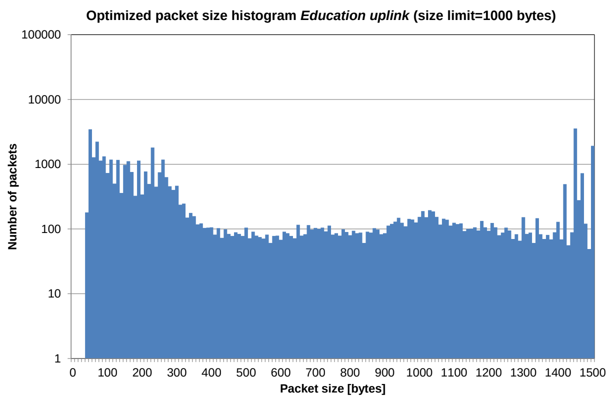


Figure 14. Size histogram of the optimized *education_uplink* trace.

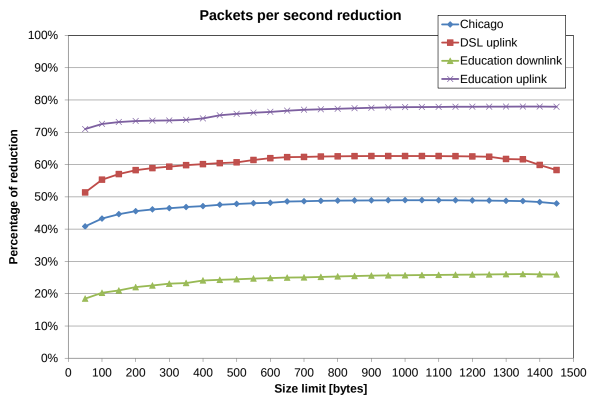


Figure 15. Packets per second savings for each trace using a multiplexing period of 100 milliseconds.

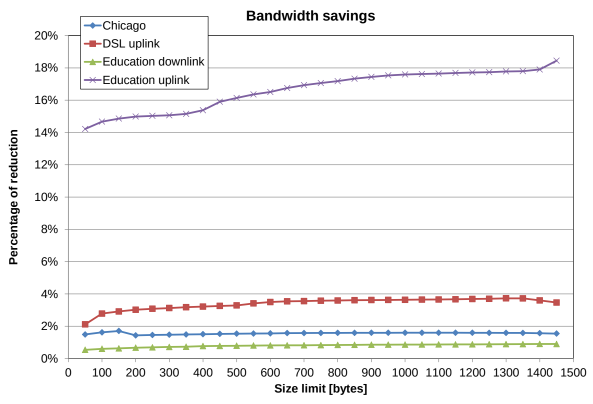


Figure 16. Bandwidth savings for each trace using a multiplexing period of 100 milliseconds.

It can be seen that for the two high-throughput traces (*Chicago* and *DSL_uplink)*, the small packets can be almost totally avoided (see Fig. 11 and 12). The cause is that the amount of packets per second is very high, so a significant number of small packets can be multiplexed together. As shown in Fig. 15, this is translated into significant reduction in the amount of packets per second (50 % for *Chicago* and 60 % for *DSL_uplink)*. However, the reduction is not very significant in terms of bandwidth (between 1 and 4 %), see Fig. 16. The cause is that big packets account for the vast majority of the bandwidth of the trace.

Regarding the two *Education* traces, it can be seen that the amount of small packets is significantly reduced in both cases. However, the savings are higher in the uplink (75 % of packets per second reduction and 17 % of bandwidth savings), since the amount of big packets in the *uplink* is significantly lower than in the *downlink* (see Fig. 9 and 10).

All in all, a conclusion can be drawn: the method is able to modify the traffic profile of all the flows (compare Fig. 7 to 10 with their corresponding ones Fig. 11 to 14), making a sort of “shifting” of the small packets towards higher sizes. This is translated into significant reductions in the number of packets to be managed by the network, which can, in turn, lower the energy consumption, taking into account that packet processing and switching is the most energy-consuming task in high-end routers [10].

Traffic optimization will certainly require an increase of the processing capacity in the *ingress* and *egress* switches, taking into account that they will have to optimize and rebuild the packets respectively. However, this increase would be cancelled out over the network, since the number of frames to switch would be reduced in the intermediate elements. As a consequence, the processing capacity could be increased in the edge devices and reduced in the intermediate ones.

C. Effect of the Additional Delays

As a counterpart of savings, an additional latency must be added to the multiplexed packets, caused by the retention time required in the *ingress* switch in order to get a number of packets before building the multiplexed frame. This new multiplexing delay may have a different effect depending on the nature of the service which packets are being multiplexed. As stated in [32], the most important factors on e.g. a voice conversation are transmission quality and ease of communication; whereas in web browsing the most important parameter is the download time.

Another problem is that the translation between objective QoS parameters and subjective QoE is not straightforward. In [33] a generic quantitative relationship between them was proposed, and an exponential formula was devised.

In the case of real-time services, the proposed method should avoid the addition of delays which could impair user’s experience, so an upper bound must be defined for the multiplexing period. Another potential problem, studied in [26], can be the reduction of the sending rate of TCP flows, since TCP is a closed loop protocol governed by ACK arrivals. However, as we will next see, these additional delays can be really small, and in some cases can be an almost negligible contribution to the end-to-end latency.

Table IV shows the average delay and jitter (standard deviation of the delay) added to the packets of the public traffic traces, as a consequence of the use of the optimization method. We have repeated the tests using *PE*=10 ms, in addition to the value of 100 ms employed before. The reduction in packets per second and bandwidth (size limit = 1000 bytes) are also reported.

As it can be seen in the table, in the case of *Chicago* and *DSL_uplink* traces, the additional delay is negligible (less than 1 ms). This means that the multiplexing period never expires, but the sending is triggered by the fact of the multiplexed packet being near the MTU. In fact, it can be observed that the values are roughly the same for both values of the period.

However, for the *Education* traces, the average delay is roughly *PE*/2, which would correspond with a delay uniformly distributed between 0 and *PE*. The value is slightly lower taking into account that some periods do not end because of MTU completion. The value of the additional jitter is *PE*/sqrt(12), as explained in [24].

TABLE IV. ADDITIONAL DELAY AND JITTER ADDED TO EACH TRACE (MILLISECONDS)

PE	[ms]	Chicago	DSL uplink	Education downlink	Education uplink
10 ms	delay	0.057	0.439	5.01	4.96
	jitter	0.050	0.384	2.91	2.90
	pps savings	48.95 %	62.63 %	9.55 %	45.63 %
	BW savings	1.59 %	3.62 %	0.17 %	8.68 %
100 ms	delay	0.057	0.439	46.05	45.92
	jitter	0.050	0.397	29.77	29.67
	pps savings	48.95 %	62.64 %	25.71 %	77.78 %
	BW savings	1.59 %	3.63 %	0.86 %	17.59 %

Regarding bandwidth and pps savings, they remain the same for the *Chicago* and *DSL_uplink* traces, despite the value of the multiplexing period, since in fact the period never expires. However, the savings become lower for the *Education* traces if the period gets reduced from 100 to 10 ms: in the uplink, the pps savings get reduced from 77 to 45 %, and the bandwidth saving becomes only 8 %. In the downlink, the pps savings drop from 25.7 to 9.5 %. Bandwidth savings are marginal in both cases. The cause of the savings being reduced with the period is clear: if the period is shorter, a lower number of packets can be multiplexed. However, the savings in terms of packets per second are still significant, especially in the uplink, and only 5 ms of latency are added.

In [34] a study of the factors influencing subjective quality for mobile users was presented, and the increase of the delay was related to MOS reduction. However, in that study the values of the RTT were much higher than the ones considered in the current proposal. In addition, the influence of the delays caused by optimization techniques on subjective quality for real-time services has been explored before for VoIP [20], [23] and online games [24], and it has been shown that user’s perceived quality can be maintained in acceptable levels.

The provision of a good quality is possible because of three facts: first, the high rates of real-time traffic patterns: VoIP generates a packet every 20 ms; inter-packet time for the studied games are 40 and 105 ms respectively; and a 100 pps ACK flow can be easily found on the Internet. If the aggregated traffic is high enough, the delay can be kept low. Second, an upper bound can be set on the added delay, by the use of a suitable value of the multiplexing interval. For this aim, traffic classification based on flows’ statistics can automatically detect traffic patterns corresponding to certain services, and this information can be used to establish the interactivity requirements of a given flow [35], and to set the multiplexing period accordingly. Third, significant bandwidth savings are obtained even if only two packets are multiplexed.

VII. Conclusions and Future Work

A method able to save bandwidth, and to reduce the amount of packets per second, for services using small packets in OpenFlow SDNs, has been proposed. It is based on multiplexing together different flows sharing a common network path, and compressing packet headers. For this aim, the fields that are the same for all the packets in a flow are included in the Openflow tuple, and then avoided in the intermediate hops.

Bandwidth savings up to 68 % for IPv4 and 78 % for IPv6 can be obtained for isolated flows of small-packet services. As a counterpart, latency would be slightly increased, but the additional delay can be kept under tolerable limits for services sending high packet rates.

The optimization method has also been applied to four public Internet traffic traces, and significant reductions in terms of packets per second have been demonstrated. Bandwidth is also reduced, especially in those traces where the percentage of small packets is high. The additional delay can be kept very low (below 1 millisecond) if the throughput is high, but it may become significant for low-throughput traces. Thus, a trade-off between bandwidth saving and additional delay appears in those cases.

Some lines for future work have been identified: a prototype of the proposed traffic optimizer would allow an extensive battery of measurements, including e.g. the processing delays which would appear when compressing the packets. In addition, a running implementation would permit to study the trade-off between the energy savings (mainly produced by the reduction in terms of packets per second), and the increase of the processing charge at the *ingress* and *egress* elements. Finally, the use of machine-learning methods for automatic identification of traffic patterns would enable a better control of the optimization period, so as not to add undesired delays to services with delay limits.

References

[1] J. Saldana, F. Pascual, D.de Hoz, J. Fernandez-Navajas, J. Ruiz-Mas, D. R. Lopez, D. Florez, J. A. Castell, M. Nunez, “Optimization of Low-efficiency Traffic in OpenFlow Software Defined Networks,” Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems SPECTS 2014, pp. 550-555, Monterey, CA, USA, July 6-10, 2014.

[2] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hühn, R. Merz, “Programmatic orchestration of wifi networks,” In 2014 USENIX Annual Technical Conference (USENIX ATC 14) (pp. 347-358). USENIX Association.

[3] A. Sathiaseelan, C. Rotsos, C. S. Sriram, D. Trossen, P. Papadimitriou, J. Crowcroft, “Virtual Public Networks,” Second European Workshop on Software Defined Networking (EWSDN), Berlin, October 2013.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, J. Turner, “OpenFlow: enabling innovation in campus networks,” in ACM SIGCOMM Computer Communication Review, 38(2), pp. 69-74, 2008.

[5] Huawei, “Smartphone Solutions White Paper”, Issue 2, 2012.07.17. Available at: http://www.huawei.com/ilink/en/download/HW_193034

[6] S. Ratti B. Hariri, S. Shirmohammadi, “A Survey of First-Person Shooter Gaming Traffic on the Internet,” IEEE Internet Computing 14, 5: 60-69, 2010.

[7] K-T. Chen, P. Huang, C-Y. Huang, C-L. Lei. “Game traffic analysis: An MMORPG perspective,” Computer Networks 50.16 (2006): 3002-3023.

[8] P. Svoboda, W. Karner, M. Rupp, “Traffic Analysis and Modeling for World of Warcraft,” in Proc. IEEE International Conference on Communications, ICC, Urbana-Champaign, IL, USA, 2007.

[9] W.C. Feng, F. Chang, W. Feng, J. Walpole, “A traffic characterization of popular on-line games,” IEEE/ACM Transactions on Networking 13.3: 488-500, 2005.

[10] R. Bolla, R. Bruschi, F. Davoli, F .Cucchietti, “Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures”, Communications Surveys & Tutorials, IEEE, vol.13, no.2, pp.223-244, Second Quarter 2011.

[11] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, S. Wright, “Power Awareness in Network Design and Routing”, INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, vol., no., pp.457-465, 13-18 Apr. 2008.

[12] B. Ginzburg, A. Kesselman, “Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n,” Sarnoff Symposium, 2007 IEEE.

[13] A. Saif, M. Othman, S. Subramaniam, N. AbdulHamid, “Impact of aggregation headers on aggregating small MSDUs in 802.11n WLANs,” Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference on , pp.630,635, 5-8 Dec. 2010.

[14] E. Ertekin, 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, Nov. 2004.

[15] K. Sandlund, G. Pelletier, L-E. Jonsson, “The RObust Header Compression (ROHC) Framework,” RFC 5795, 2010.

[16] B. Thompson, D. Wing, T. Koren, “Tunneling Multiplexed Compressed RTP (TCRTP),” RFC4170. 2005.

[17] T. Koren, S. Casner, J. Geevarghese, B. Thompson, P. Ruddy, “Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering,” RFC 3545, 2003.

[18] R. Pazhyannur, I. Ali, C. Fox, “PPP Multiplexing,” RFC 3153, 2001.

[19] J. Lau, M. Townsley, I. Goyret, “Layer Two Tunneling Protocol - Version 3 (L2TPv3),” RFC 3931, 2005.

[20] R. M. Pereira, L.M. Tarouco, “Adaptive Multiplexing Based on E-model for Reducing Network Overhead in Voice over IP Security Ensuring Conversation Quality,” in Proc. Fourth international Conference on Digital Telecommunications, Washington, DC, pp. 53–58, Jul. 2009.

[21] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, D. Wing, M. Perumal, M. Ramalho, G. Camarillo, F. Pascual, D. R Lopez, M. Nunez, D. Florez, J.A. Castell, T. de Cola, M. Berioli, “Emerging real-time services: optimizing traffic by smart cooperation in the network,” Communications Magazine, IEEE, vol.51, no.11, pp.127,136, Nov 2013.

[22] S. Jivorasetkul, M. Shimamura, K. Iida, “End-to-End Header Compression over Software-Defined Networks: A Low Latency Network Architecture,” in Proc. Int. Conf. on Intelligent Networking and Collaborative Systems, Washington DC, USA, pp. 493-494, 2012.

[23] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, J. Murillo, E. Viruete, J. I. Aznar, "Evaluating the Influence of Multiplexing Schemes and Buffer Implementation on Perceived VoIP Conversation Quality," Computer Networks (Elsevier), vol 56, Issue 7, pp. 1893-1919, May 2012.
[24] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, E. Viruete Navarro, L. Casadesus, "Online FPS Games: Effect of Router Buffer and Multiplexing Techniques on Subjective Quality Estimators," Multimedia Tools and Applications, Vol. 71, Issue 3, pp 1823-1856, Aug 2014, Springer.
[25] J. Saldana, "The Effect of Multiplexing Delay on MMORPG TCP Traffic Flows," Consumer Communications and Networking Conference, CCNC 2014. Las Vegas, pp 447-452, Jan 2014.
[26] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, "Can We Multiplex ACKs without Harming the Performance of TCP?," Consumer Communications and Networking Conference, CCNC 2014. Las Vegas, pp 921-922, Jan 2014.
[27] M. Marchese, "Quality of Service over Heterogeneous Networks," Wiley & Sons, Chichester, UK, 2007.
[28] T. Braun, M. Diaz, J.E. Gabeiras, T. Staub, "End-to-End Quality of Service Over Heterogeneous Networks," Springer Science & Business Media, 2008.
[29] R.R. Denicol, E.L. Fernandes, C.E. Rothenberg, Z.L. Kis, "On IPv6 support in OpenFlow via Flexible Match Structures," Nov 2011.
[30] The CAIDA UCSD equinix-chicago- 20140619-130100, https://data.caida.org/datasets/passive-2014/equinix-chicago/20140619-130000.UTC/
[31] R. R. R. Barbosa, R. Sadre, A. Pras, R. Meent, "Simpleweb/university of Twente traffic traces data repository, 2010". DSL trace: https://traces.simpleweb.org/traces/TCP-IP/location4/loc4-20040207-2001.bz2
Education trace: https://traces.simpleweb.org/traces/TCP-IP/location6/loc6-20070615-1644.gz
[32] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, G. Pujolle, "Quality of Experience of VoIP Service: A Survey of Assessment Approaches and Open Issues," Communications Surveys & Tutorials, IEEE, Vol. 14, Issue 2, 2012.
[33] M. Fiedler, T. Hossfeld, P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," Network, IEEE, Vol. 24, Issue: 2, 2010, pp. 36-41.
[34] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.H. Hong, A.K. Dey, "Factors influencing quality of experience of commonly used mobile applications," Communications Magazine, IEEE, Vol. 50, Issue: 4, 2012, pp 48-56.
[35] T. T. Nguyen, G. Armitage, P. Branch, S. Zander, "Timely and continuous machine-learning-based classification for interactive IP traffic," IEEE/ACM Trans. on Networking, 20(6), pp 1880-1894, 2012.

Jose Saldana, San Sebastian, 1974, received his B.S. and M.S. in Telecommunications Engineering from University of Zaragoza, in 1998 and 2008, respectively. He received his Ph.D. degree in Information Technologies in 2011.

He is currently a research fellow in the Department of Engineering and Communications of the same University. He has published a number of research articles, as "First person shooters: can a smarter network save bandwidth without annoying the players?, Communications Magazine, IEEE 49 (11), 190-198; "Evaluating the influence of multiplexing schemes and buffer implementation on perceived VoIP conversation quality," Computer Networks 56 (7), 1893-1919. His research interests focus on Quality of Service in Real-time Multimedia Services, as VoIP and networked online games.

Dr. Saldana is a Member of IEEE and ISOC. He has participated in the Organization and Technical Committees of international conferences, as IEEE CCNC, Globecom or ICC.

Jorge David de Hoz Diego was born in Valladolid, Spain in 1983. He received his Telecommunications Engineering degree from the University of Valladolid in 2008, and his M.S. from the University of Zaragoza, Spain in 2013.

He worked from 2008 to 2009 at FURIA government consortium. He focused on researching and the validation of DVB-H/DVB-SH standards and involved technologies. In 2011 he cofounded Ateire Tecnología y Comunicación in Zaragoza, Spain where participated developing Digital Signage technology. In 2013 he also cofounded Servicios de TI de Durango in Mexico as a subsidiary of the Spanish firm. He is currently leading in this business a research project partially funded by CONACY about Digital Signage networks and its architecture.

Julián Fernández-Navajas was born in Alfaro (La Rioja, Spain), graduating with honours in Telecommunications Engineering degree from the Polytechnic University of Valencia, Spain, in 1993, and receiving his Ph.D. degree "cum Laude" from 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. He is and was responsible investigator in several national and international projects related to the research mentioned above and has published several research works on impact journals and magazines such as IEEE Communications Magazine, IEEE Communications Letters, Computer Networks, Multimedia Tools and Applications, International Journal of Medical Informatics and so on.

José Ruiz Mas received the Engineer of Telecommunications degree from the Universitat Politècnica de Catalunya (UPC), Spain, in 1991 and the Ph.D. degree from the University of Zaragoza (UZ) in 2001. He worked as a software engineer at the company TAO Open Systems from 1992 to 1994. In 1994 he joined the Higher Engineering and Architecture School of UZ as an Assistant Professor until 2003, when he became an Associate Professor.

He is with the Department of Electronics Engineering and Communications in the Higher Engineering and Architecture School of UZ. He is member of the Aragón Institute of Engineering Research (I3A). He is co-investigator since 1995 of research grants from the Ministry of Science and Technology, the Sanitary Research Funds and the Government of Aragon (Spain) in the areas of distributed multimedia systems and wireless networks. At present 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).

Fernando Pascual Blanco, Madrid, 1983, received his Telecommunications Engineering degree from the Alcalá de Henares University (UAH) in 2007, and his Master's in Information Technologies and Communications, major in Computer Engineering and Networks, from the same university in 2011.

He has worked at Telefonica I+D since 2006, and his professional career is focused on network intelligence and M2M services development for the Telefonica group. He is currently working for Telefonica Global CTO department where his responsibilities are focused on the mobile core networks as core network engineer expert.

Diego R. Lopez, PhD, joined Telefonica I+D in 2011 as a Senior Technology Expert on network middleware and services. He is currently in charge of the Technology Exploration activities within the GCTO Unit of Telefónica I+D. Before joining Telefónica he spent some years in the academic sector, dedicated to research on network service abstractions and the development of APIs based on them. During this period he was appointed as member of the High Level Expert Group on Scientific Data Infrastructures by the European Commission.

Dr. Lopez is currently focused on identifying and evaluating new opportunities in technologies applicable to network infrastructures, and the coordination of national and international collaboration activities. His current interests are related to network virtualization, infrastructural services, network management, new network architectures, and network security. Diego is actively participating in the ETSI ISG on Network Function Virtualization (chairing its Technical Steering Committee), the ONF, and the IETF WGs connected to these activities.

David Flórez was born in Madrid on 28/05/1967. He finished his high school education in 1984 and obtained a Degree in Telecommunications Engineering for the ETSIT of Madrid, Spain in 1990.

He served his military term in the Spanish Army during 1991. In 1992 started working in the Image Processing Group of the ETSIT of Madrid, year when he also joined Telefonica, where he currently works in charge of coordinating the signaling strategy and procurement for the Telefonica group, located in Madrid, Spain. In the past he has been involved in the European projects EMN, RAMA, Mintour, Mobicome, Vital++, Secured, R2D2 and Romeo, the last two ones as coordinator. He also collaborated in the multigroup PSSBA project and its follow-up PSA, for the DSL deployment in South America, where he contributed as software quality expert. Before that he worked as programmer in the project VEMMI, SIMMA and Connected Life and to coordinate later the GIE, AvisaBuzon and LM for Telefonica online site.

Juan Antonio Castell was born in Madrid, 1970. He holds a Technical Engineering Degree from the Polytechnic University of Madrid (Spain) in 1991.

He currently works as Technological Expert at Telefonica, in the Global CTO office, sited at Madrid, Spain. He is in charge of the architectural solution for the data access (elements like the PCRF, DPI, network proxy, etc.) from the technical point of view. Since 2010 until December 2013 he was the technical coordinator of the Telefónica I+D innovation activities related with network intelligence and signaling/control protocols. In his professional beginning, he started working as analyst-programmer in different companies always for telecommunications projects; after that, he joined Telefónica I+D where the first requested task as employee was to implement the first AAA server for its IP network, Infovia+, in 1997. Since that moment until 2010 he was responsible of the AAA and DNS implementations for Telefonica Spain IP services.

He is also author or contributor of several patents regarding these concepts.

Manuel Núñez Sanz received a six years degree in Telecommunications Engineering from the Polytechnic University of Madrid (Spain) in 1997. That same year he was employed as research engineer in Telefónica I+D and he is currently head of core network in TEF Global CTO where he is in charge of technological coordination and economical control in core network topics.

He is author or coauthor of several international patents in control technologies and network signaling (see WO2012065658, EP2782320, etc.). He has collaborated in some published articles (see "Emerging Real-Time Services…" in IEEE Communications Magazine in November 2013) and books (see "3D Future Internet Media" by Springer) and has been invited as speaker in some international forums (WebRTC Global Summit 2014 London, IMS World Forum 2014 Barcelona).