2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

Improving Network Efficiency with Simplemux

Jose Saldana, Ignacio Forcén, Julián Fernández-Navajas, José Ruiz-Mas
CeNITEQ Group, Aragon Institute of Engineering Research (I3A)
EINA, University of Zaragoza
Zaragoza, Spain
{jsaldana, nforcen, navajas, jruiz}@unizar.es

***Abstract*—The high amount of small packets currently transported by IP networks results in a high overhead, caused by the significant header-to-payload ratio of these packets. In addition, the MAC layer of wireless technologies makes a non-optimal use of airtime when packets are small. Small packets are also costly in terms of processing capacity. This paper presents Simplemux, a protocol able to multiplex a number of packets sharing a common network path, thus increasing efficiency when small packets are transported. It can be useful in constrained scenarios where resources are scarce, as community wireless networks or IoT. Simplemux can be seen as an alternative to Layer-2 optimization, already available in 802.11 networks. The design of Simplemux is presented, and its efficiency improvement is analyzed. An implementation is used to carry out some tests with real traffic, showing significant improvements: 46% of the bandwidth can be saved when compressing voice traffic; the reduction in terms of packets per second in an Internet trace can be up to 50%. In wireless networks, packet grouping results in a significantly improved use of air time.**

***Keywords*—traffic optimization; small packets; network efficiency; multiplexing**

I. Introduction

An inherent characteristic of packet-switched networks is the need for a number of headers, which are added to the payload in order to make it possible its transport from the source to the destination. This overhead is not a problem when the payload is big, but the inefficiency caused by payloads in the order of tens of bytes may become excessive in certain scenarios with limited resources.

However, small packets are ubiquitous in nowadays' networks. The first example, and perhaps the most evident one, are TCP Acknowledgements, which are required by the traffic control mechanisms inherent to this protocol. These packets do not usually carry any payload, so they can be considered as *100% overhead* packets. However, they are carried by the network in the same way as any other packet. In addition, some multimedia applications (e.g. IPTV) send big UDP packets, but also require application-level acknowledgements [1].

In addition, emerging real-time services (e.g. VoIP, online games) also send high rates of small packets, required to grant the interactivity level demanded by the end users. Finally, small packets may also be present in Machine to Machine and IoT scenarios with limited network and energy resources.

In Fig. 1, the packet-size histogram of a traffic trace from *The CAIDA Anonymized Internet Traces 2015 Dataset* [2] is presented (only the first 200,000 packets have been used). This trace was captured at a backbone link (10 GigE) of a Tier1 ISP between Chicago and Seattle. As it can be observed, 44% of the packets are big, i.e. their size is near 1500 bytes, which corresponds to the Maximum Transmission Unit (MTU). At the same time, a high amount of small packets are present: 33.4% packets are 60 bytes or smaller, and 41.4% are 200 bytes or smaller.

The presence of these amounts of small packets has some drawbacks: first, the *overhead* drawback, i.e. the amount of header bytes required for sending a small payload is high. For example, a VoIP packet carrying 20 bytes requires at least 40 bytes corresponding to IP, UDP and RTP headers.

Second, when the packet is passed to Layer 2, the Media Access Control (MAC) mechanisms make it necessary to define a number of waiting intervals until a host gets the channel. This delay may be negligible in Ethernet (a preamble, a header and an inter-packet gap are required), but it may result significant when packets are using a wireless technology as e.g. 802.11. We will refer to this problem as the *airtime* drawback.

The third drawback is related to the limited processing capacity of the machines managing the packets. Datasheets of commercial devices often report the performance in terms of both bandwidth and packets per second. For example, a commercial 50-Port switch [3] has *"13.6 Gbps switching*

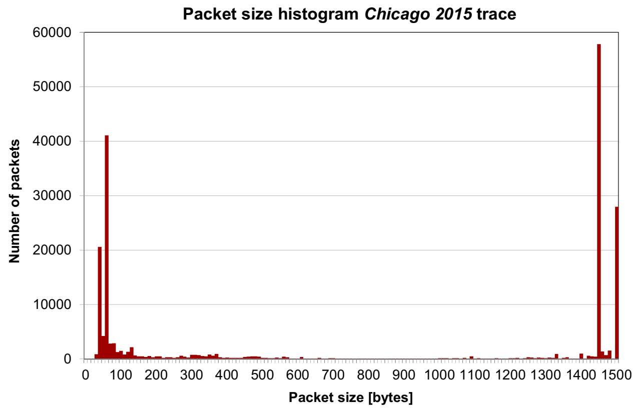


Fig. 1. Packet size histogram of a traffic trace obtained in a backbone link

This work has been partially financed by the EU H2020 Wi-5 project (Grant Agreement no: 644262), and European Social Fund in collaboration with the Government of Aragon.

capacity, 10.1 million pps". This means that, for 100-byte packets, the throughput of the switch would in fact be limited to 8.08 Gbps. Something similar happens with Layer-3 packet routing: the so-called *"64 Byte Throughput"* has become a figure of merit, and specific features are being added to processors and network interfaces in order to improve the way they manage these small packets [4]. This would be the *processing* drawback of small packets.

Other drawbacks may also appear, but they will not be considered in this paper. For example, the energy consumed by a router is expected to rise not only with the generated and received throughput, but also with the amount of packets per second it is managing. According to [5], internal packet processing engines and switching fabric require 60% and 18% of the power consumption of high-end routers respectively.

In this context, the aggregation (multiplexing) of a number of small packets to form a single and bigger one can be seen as a means to jointly overcome these drawbacks. Some initiatives aimed to increase the average size of the Internet packets are already in place. For example, Ethernet Jumbo Frames, with an MTU of 9000 bytes instead of 1500, present some advantages in terms of throughput, and even in the TCP window growth speed [6]. Furthermore, in Wi-Fi networks, frame aggregation was added as an option to 802.11n [7], and this supposed a mitigation of the *overhead* and *airtime* drawbacks.

In this same line, the present work presents Simplemux, a generic and lightweight multiplexing protocol aimed to enable the multiplexing of a number of packets belonging to a protocol (the "multiplexed packets"), into another protocol (i.e. the "tunneling protocol"), as shown in Fig. 2. Small separators are inserted between the packets, including the length and an identifier of the protocol of the multiplexed packets. Thus, any tunneling protocol may be virtually able to carry a number of packets of any other protocol.

Simplemux does not act at Layer 2, but at upper layers, so it has been proposed to the IETF [8]. It does not require a session setup, since the idea is that a specific IANA *Protocol Number* [9] could be reserved to Simplemux. Thus, if the destination machine receives a packet with this protocol number, and if it implements Simplemux, it just demultiplexes it and extracts the packets. There is no need to set up a session or to store any session parameter.

The objective of Simplemux is not to multiplex packets at the end hosts, i.e. the ones originating the flows. Although this is still possible, the benefit would be small, since the probability of having a high number of small-packet flows between the same pair of machines is low. Thus, the idea is to run Simplemux between a pair of networks machines (*Ingress* and *Egress*), Fig. 3, to optimize traffic flows sharing this path.

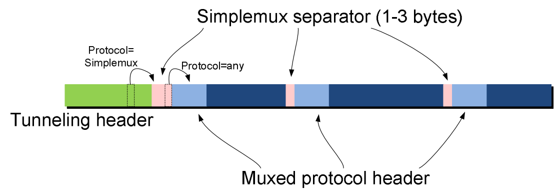


Fig. 2. Scheme of a Simplemux packet

So the optimization covers a number of network hops, but the intermediate routers just have to route packets at IP level, so they do not have to implement Simplemux.

The contribution of this paper consists of the presentation of Simplemux, a detailed explaining of its design, an analysis of the expected benefits, and a series of evaluation tests deployed with an experimental Linux implementation developed by our group. The results show that up to 46% of the bandwidth can be saved when compressing voice traffic. The reduction in terms of packets per second in an Internet trace can be up to 50%. In wireless networks, packet grouping results in a significantly improved use of air time

The rest of the paper is structured as follows: The related work is summarized in the next Section. Section III summarizes the scenarios of interest. The design of Simplemux is presented in Section IV. Section V includes an analysis of the expected improvements, Section VI presents the implementation and Section VII details the obtained experimental results. The paper ends with the Conclusions.

II. Related Work

Taking into account that this paper is presenting and studying a new standard proposal for traffic optimization, this Section has been divided into a first part summarizing related standards, and a second part summarizing related research.

A. Multiplexing standards

The aim of multiplexing is to save bandwidth (i.e. minimizing the *overhead* drawback) and to reduce the amount of packets per second (i.e. reducing the *airtime* drawback in wireless networks). Although it can be deployed in Layer 2 or in upper layers, there are some differences between these two possibilities. If frames are multiplexed at Layer 2, they will traverse a single link together, and then they will be de-aggregated. If the common path includes a number of links, a demux-mux process will be required at each node. However, if multiplexing is deployed at upper levels, packets can travel together during the whole path, thus reducing the *processing* drawback, at the cost of adding a tunneling header which allows end-to-end delivery.

Regarding Layer 2, two frame multiplexing policies are included in 802.11n [7] (and subsequent versions), namely MAC Service Data Unit aggregation (A-MSDU) and MAC Protocol Data Unit aggregation (A-MPDU). Aggregation has become even more important in 802.11ac [10], where all the frames must have an A-MPDU format, even if they include a single sub-frame. The former allows multiple MSDUs to be

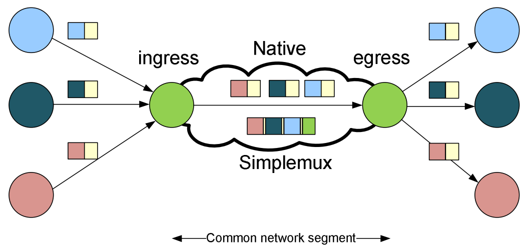


Fig. 3. Simplemux optimization between an *Ingress* and an *Egress* machine

sent to the same receiver, as a single MPDU. The latter is able to join a number of MPDU sub frames sharing a single leading PHY header. The main difference is that A-MPDU works after the MAC header encapsulation. This has two advantages: a higher bandwidth reduction, and the fact that each sub-frame includes its own CRC, which avoids the need for discarding the whole frame because of an error in a single sub-frame: if a sub-frame is wrong, only that one will be retransmitted.

Different standard protocols for multiplexing packets at higher layers do exist. First, TMux [11] was approved as RFC1692 in 1994 to include a number of transport-level payloads (mainly TCP) into a single packet. This protocol is therefore limited to multiplexing packets traveling between the same pair of machines. Mini-headers of 4 bytes are inserted before each of the multiplexed headers.

Another IETF multiplexing protocol is PPPMux [12], an extension of PPP allowing the inclusion of a number of packets into a single frame. This protocol permits to multiplex whole packets. However, when packets have to travel over an IP network, the use of L2TP (Layer 2 Tunneling Protocol) is necessary, as it happened in TCRTP [13], which combined header compression with multiplexing and tunneling, in order to optimize VoIP RTP flows.

B. *Research Works Evaluating Standards' Performance*

A number of research works have studied the achievable savings when using 802.11 aggregation. In [14], the authors presented an analytic model for estimating the improvements of A-MPDU and A-MSDU. Their findings showed a significant efficiency increase, and a better performance of A-MPDU, which was stressed when packet-error rate was high. In [15] a scheduler for selecting the packets to be multiplexed together was proposed, taking into account that the 802.11 standard only specifies the frame format, but scheduling schemes are left as vendor's choice. An optimal frame size was calculated, and packets were grouped trying to fit this size.

Multiplexing at higher layers has been studied in other works: [16] and [17] presented different proposals for VoIP optimization, trying to keep a good voice quality. Our group has also proposed the adaptation of TCRTP for other real-time services as online games. The work was summarized in [18], where other references can also be found.

***

The specific contribution of the present work, with respect to the previous ones, is the proposal of a new and lightweight multiplexing protocol. It may work as an alternative for 802.11 aggregation, with the additional benefit of end-to-end optimization including a number of hops. It may also be useful in legacy systems not implementing the aggregation mechanisms defined by 802.11n and subsequent versions.

Simplemux can also be seen as an improvement with respect to existing Layer 3 optimization mechanisms, since it is able to accomplish the same tasks with a reduced management. It can also substitute PPPMux, mainly aimed for point-to-point links. In fact, the L2TP and the PPP sessions required by TCRTP could be avoided.

III. SCENARIOS

In [18] some scenarios of interest where multiplexing can provide benefits were identified:

- The aggregation network of an operator. Multiplexing can be deployed between different devices in residential scenarios. Traffic can be grouped at different levels as e.g. at gateway level in LTE or ISP edge routers.
- In corporate environments, a number of simultaneous small-packet flows (as e.g. VoIP calls or remote desktop sessions) between two central offices of a company can be multiplexed.
- In Machine-to-Machine and IoT scenarios with bandwidth and processing constraints, small packets generated by e.g. a sensor can be aggregated.

In addition to these scenarios, a new one has been identified, namely Community Networks or other network deployments alternative to traditional ones [19], which are becoming popular in the last years, with a special relevance in developing countries. In these networks, users create, own and manage the network infrastructure, which serves as a backhaul for providing Internet access. They are not only deployed in remote villages, but also in urban environments [20].

In this kind of scenario, Simplemux can be used for traffic optimization, taking into account some special questions: first, these networks are mainly based on wireless (Wi-Fi) links, so the packets traverse a series of hops before reaching their destination. In that case, an end-to-end optimization can reduce the amount of packets (*airtime* drawback) with an advantage with respect to Layer-2 optimization, which should be applied at each intermediate router (*processing* drawback). In Fig. 4, thick blue lines represent the paths where Simplemux could be employed to optimize the traffic. In these scenarios, especially in developing countries, new versions of 802.11 including frame aggregation may not be widely deployed.

In some cases, a network operator and the owners of the infrastructure may sign an agreement, so the operator deploys a 3G femtocell in a remote village (see the figure), using the community network as a backhaul for connecting to the Internet. This constitutes a win-win scenario, since the operator avoids the costs of extending the network to that village, which can get connected with the mobile network [21].

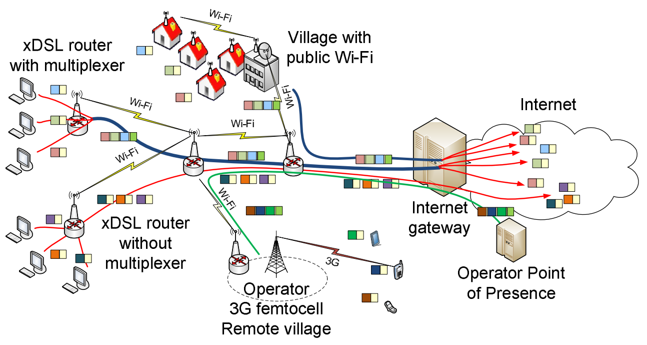

Wi-Fi
xDSL router with multiplexer
Village with public Wi-Fi
Wi-Fi
Wi-Fi
Wi-Fi
Wi-Fi
Internet
Wi-Fi
xDSL router without multiplexer
Internet gateway
3G
Operator 3G femtocell Remote village
Operator Point of Presence

Fig. 4. Simplemux optimization in an *alternative network* scenario

IV. Protocol Design

This section briefly describes the protocol, which is specified in [8]. The aim of the protocol is the reduction of the overhead, while maintaining its design as simple as possible. The main element are the *separators,* i.e. the small headers inserted before each packet. The *length* of the multiplexed packet that comes next is included in these separators.

The protocol allows the inclusion of packets belonging to different protocols. This provides a high degree of flexibility, since it avoids the necessity of creating one tunnel for each kind of traffic. As an example, ROHC-compressed packets *(Protocol number* = 142) may travel with normal IP packets *(Protocol number* = 4), as shown in Fig. 5.

However, there will be many cases where all the packets included in a multiplexed bundle belong to the same protocol. Therefore, the format of the first separator is slightly different from the rest. It includes a *Single Protocol Bit (SPB)* which is set when packets belonging to a single protocol are carried, thus saving a byte on each of the subsequent separators.

In addition, all the bytes of the separators include a *Length eXTension bit (LXT)* which is set when the length of the packet requires an extra byte. This permits to multiplex packets of any length, also allowing to adjust the size of the field according to the packet length.

The format of the separators is shown in Fig. 6. In a), the format of the *first* separator is shown, in b) the *non-first* separators when a single protocol is carried, and in c) the *non-first* separator when packets belong to different protocols.

It can be observed that the overhead required by Simplemux separators is minimum. For example, if a set of small packets belonging to the same protocol travel together, the first separator would require two or three bytes, but the rest of the packets would only need a single byte, or two if they are bigger than 128 bytes.

V. Analytical Calculation of the Expected Savings

This section has three parts: in Subsection A, the savings in terms of packets per second are obtained. Subsection B presents the calculation of the bandwidth reduction. Finally, the last subsection presents a comparison between the savings obtained by Simplemux (Layer-3) and 802.11 (Layer-2) aggregation mechanisms.

A. Packets per second savings

The scenario we are considering is a network node where packets of different sizes arrive. They are classified according to their size, and only small ones are sent to a multiplexer (Fig. 7). A period *PE* is defined at the multiplexer and every time the period expires, a packet is sent including all the arrived packets. However, if a number of packets *N* have arrived before the end of *PE*, a multiplexed packet is sent and a new period begins.

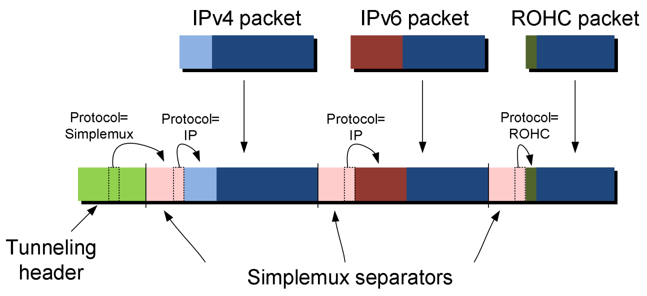
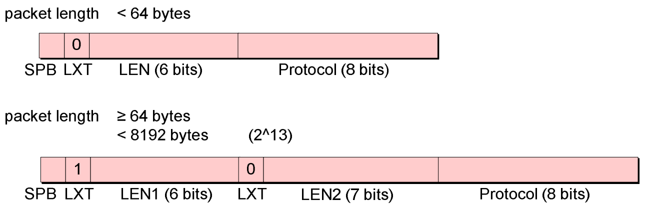
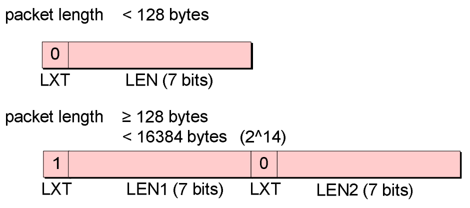


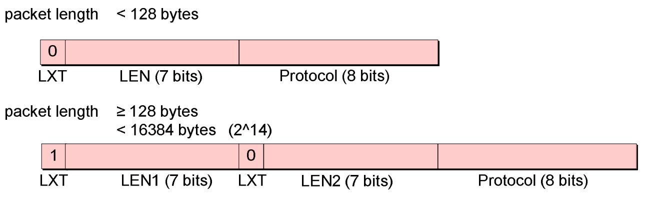
Fig. 5. Scheme of a Simplemux packet including different protocols


packet length < 64 bytes
0
SPB LXT LEN (6 bits) Protocol (8 bits)
packet length ≥ 64 bytes < 8192 bytes (2^13)
1 0
SPB LXT LEN1 (6 bits) LXT LEN2 (7 bits) Protocol (8 bits)

(a)


packet length < 128 bytes
0
LXT LEN (7 bits)
packet length ≥ 128 bytes < 16384 bytes (2^14)
1 0
LXT LEN1 (7 bits) LXT LEN2 (7 bits)

(b)


packet length < 128 bytes
0
LXT LEN (7 bits) Protocol (8 bits)
packet length ≥ 128 bytes < 16384 bytes (2^14)
1 0
LXT LEN1 (7 bits) LXT LEN2 (7 bits) Protocol (8 bits)

(c)

Fig. 6. Scheme of Simplemux separators: a) *first* separator; b) *non-first* separator when all the packets belong to the same protocol; c) *non-first* separator including the *Protocol Number*

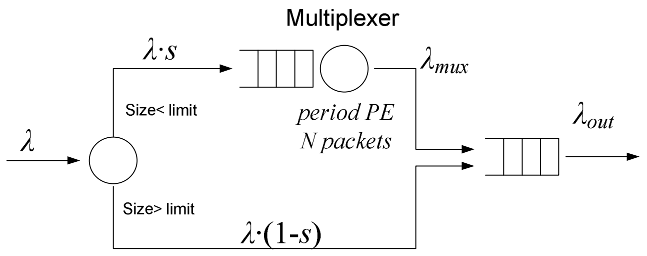


Fig. 7. Scheme of a node classifying and multiplexing traffic

Let λ be the packet arrival rate. If we set a size limit at the classifier, we have:

$$s = \Pr\left[size \leq limit\right] \qquad (1)$$

$$1 - s = \Pr\left[size > limit\right] \qquad (2)$$

We want to find λ_{out} and BW_{out}, i.e. the packet rate and the bandwidth at the output. In the following, we are assuming that no packets are lost in the queues.

Let k be the number of packets arrived to the multiplexer in a period. Three cases can be distinguished: *a)* no packets arrived; *b)* k packets arrived ($k \in [1, N\text{-}1]$); and *c)* N packets arrived. Therefore, we can express the value of λ_{mux} as:

$$\lambda_{mux} = \lambda_{mux|k=0} \cdot \Pr(k=0) + \lambda_{mux|1 \le k \le N-1} \cdot \Pr(1 \le k \le N-1) + \\ + \lambda_{mux|k=N} \cdot \Pr(k=N) \quad (3)$$

In order to simplify this expression, we must take into account that:

- If no packets have arrived, we have $\lambda_{mux|k=0} = 0$.
- If $k \in [1, N\text{-}1]$, then a multiplexed packet will be sent at the end of the period, so $\lambda_{mux|1 \le k \le N-1} = 1/PE$.
- If N packets have arrived, we have $\lambda_{mux|k=N} = \lambda \cdot s/N$.

So we can express λ_{mux} as:

$$\lambda_{mux} = \frac{1}{PE} \Pr(1 \le k \le N-1) + \frac{\lambda \cdot s}{N} \cdot \Pr(k=N) \quad (4)$$

From (4), we can see that, if the amount of packets per second arriving to the multiplexer is high, then N packets will always be multiplexed together, and the end of the period will never be reached, so $\Pr(k=N) \approx 1$, and in that case:

$$\lambda_{mux} \approx \frac{\lambda \cdot s}{N} \quad (5)$$

$$\lambda_{out} \approx \frac{\lambda \cdot s}{N} + \lambda \cdot (1-s) = \lambda \left[\frac{s}{N} + (1-s) \right] \quad (6)$$

This simplified expression (only valid for high amounts of traffic) is ruled by two parameters, namely the number of packets that can be multiplexed N, and the probability of having a small packet s. So the reduction in terms of packets per second will depend on the packet size distribution, but it will also depend on the MTU of the underlying technology employed, since N is related to both parameters.

Finally, if a header compression algorithm as ROHC [22] is employed, the compression ratio will also have an influence on the maximum number of native packets to be included in a multiplexed bundle. However, if the traffic only consists of small packets (e.g. VoIP packets between two offices), we will have s=1, so the amount of packets is directly reduced by a factor of N.

In Fig. 8, the relationship between the amount of packets per second at the input and at the output (λ_{out}/λ) is presented. The graph "Uniform size distribution" has been obtained considering a uniform packet size distribution (between 40 and 1500 bytes). In this case, the reduction in terms of packets per second could theoretically be about 50%. The graph decreases monotonically, so in a first approach it would seem that the best choice is to send all the packets to the multiplexer.

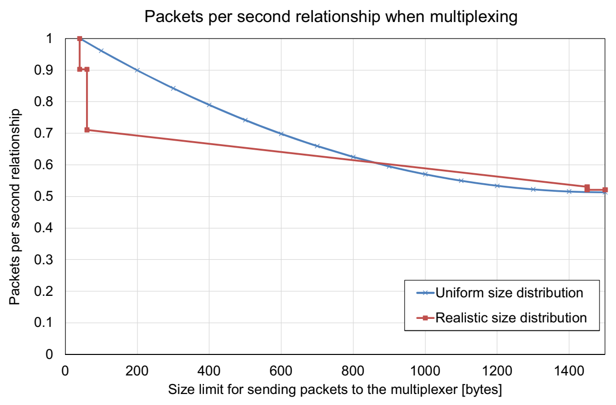


Fig. 8. Theoretical reduction of the packets per second when multiplexing 1000 packets per second

However, as shown in Fig. 1, real Internet traffic does not have a uniform size distribution; in fact, a clear division between *small* and *near-to-MTU* packets can be observed. Thus, the graph "Realistic size distribution" represents the relationship, when the traffic is modeled this way: a peak of 40-byte packets (10%); a peak of 60-byte packets (20%); a peak of 1450-byte packets (29%); a peak of 1500 bytes (14%); the other 27% of the packets are uniformly distributed between 61 and 1449 bytes. In this case, it can be observed that a significant decrease in terms of packets per second is achieved even if only small packets are sent to the multiplexer. If bigger packets are also multiplexed, the curve decreases slowly. So it would make sense to send to the multiplexer only the small ones, thus avoiding additional processing, delays, etc. to affect the big packets. In the test section we will present some results obtained with a real Internet trace.

B. Bandwidth savings

In order to obtain the bandwidth at the output, we need the average size of the multiplexed packets, E[*size*]. The bandwidth will be:

$$BW_{out} = \lambda_{mux} \cdot \mathrm{E}[size_{muxed\_packets}] + \lambda \cdot (1-s) \cdot \mathrm{E}[size_{big\_packets}] \quad (7)$$

Since the expected size of big packets will normally be about one order of magnitude over that of small ones (i.e. 1500 with respect to 150 bytes), the savings in terms of bandwidth will only be significant if $\lambda_{mux} >> \lambda \cdot (1-s)$. Otherwise, the savings will mainly be achieved in terms of packets per second reduction (*airtime* and *processing* improvement).

Let *CO (Common Overhead)* be the size of the common tunneling header of the multiplexed packet; *SO (Single Overhead)* be the size of the multiplexing separator added to each packet; and *CR* the *Compression Ratio* of a hypothetical header compression mechanism employed in combination with multiplexing. The size of a multiplexed packet is a function of *k,* the number of packets arrived to the multiplexer in a period:

$$\mathrm{E}[size_{muxed\_packets}] = CO + \mathrm{E}[k] \cdot \big(SO + CR \cdot \mathrm{E}[size_{small\_packets}] \big) \quad (8)$$

Where $\mathrm{E}[k] = \lambda \cdot s / \lambda_{mux}$ ($\mathrm{E}[k] \approx N$ when the traffic load is high), *CO* and *SO* depend on the multiplexing method (e.g. if

IPv4 is used, CO=20 bytes). For Simplemux, SO will be the average size of the separators (between 1 and 3 bytes), CR depends on the header compression algorithm. Finally, $E[size_{small\_packets}]$ and $E[size_{big\_packets}]$ will depend on the statistical distribution of the packet size.

C. *Layer-2 and Layer-3 optimization in 802.11*

The analytical method proposed in [14] for calculating the savings of 802.11 aggregation mechanisms, has been adapted in order to include the possibility of multiplexing at higher layers. As a result, we have obtained Fig. 9, where the efficiency *(data rate /PHY rate)* is presented for small 200-byte frames, for both UDP and TCP traffic. It can be observed that Simplemux behaves very similar to both 802.11 frame grouping options. The number of frames that can be optimized with A-MPDU is significantly higher, because of the different MTU allowed by 802.11n standard.

VI. Description of the Protocol Implementation

An implementation of Simplemux has been developed in C, mainly for testing and demonstration purposes. It has been made publicly available [1]. The default tunneling protocol is IPv4, using *Protocol Number* 253, which is reserved by IANA "for experimentation and testing" [9]. In addition, in order to avoid firewalls dropping packets belonging to an unknown protocol, the option of using UDP has been added, at the cost of 8 additional bytes per multiplexed bundle.

The multiplexed packets are native IP, but the implementation also relies on an open-source ROHC implementation [2] for compressing headers before multiplexing the packets. This combination of Tunneling, Compressing and Multiplexing (TCM) makes it possible to send header-compressed packets end-to-end, being the common tunneling overhead shared by a number of packets.

The program runs in user space. A Linux *tun* device [3] is created, and the multiplexer captures all the packets that are forwarded to this device. Therefore, the decision about what packets are sent to the multiplexer can be implemented this way: packets are first marked with *iptables,* according to the desired policies (defined by IP, port, packet length, etc.); then *iproute* is used to forward the marked flows to the *tun* device.

The implementation includes the next features:

- Two ROHC compression modes are included: *Unidirectional* and *Bidirectional Optimistic*. A feedback channel can be created from the decompressor to the compressor. ROHC profiles included: IP/UDP/RTP, IP/UDP, IP/TCP, IP/ESP and IP/UDP-Lite.
- Four different multiplexing policies can be combined: a) sending a fixed number of native packets; b) filling a packet size; c) sending if a packet arrives after a timeout expiration; and d) sending in a periodic way.

[1] Simplemux in Github, https://github.com/TCM-TF/simplemux
[2] The OpenSource ROHC library, https://rohc-lib.org/
[3] TUN/TAP enables packet reception and transmission for user space programs in Linux.

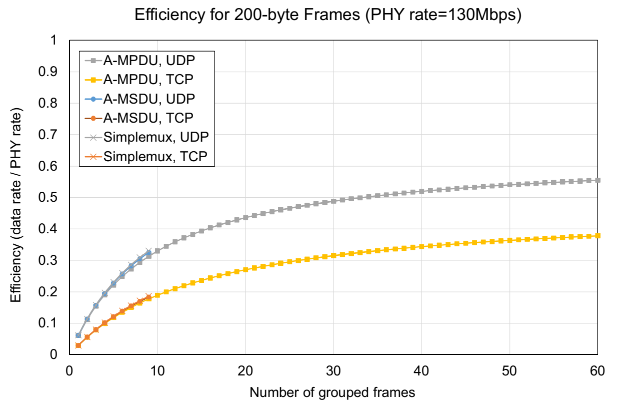


Fig. 9. Network efficiency when grouping 200-byte packets at Layer 2 or Layer 3

The implementation has 2,700 lines of code (ROHC library not included). The processing delay in a commodity PC (Intel Core i3) has been measured in roughly 0.25 ms. In a low-cost OpenWRT Access Point (TP-Link TL1043ND) it is about 3.5 ms.

VII. Tests and Results

A. *Test Setup*

a) Test scenario: Four machines are used for the tests: *generator*, *receiver*, *multiplexer* and *demultiplexer*, being the bottleneck the link between these two (Fig. 10). The bottleneck can either be wired or wireless. These are the tools employed in the tests:

- The machines are commodity PCs with Core i3 processors, running Debian 6.
- In the tests with wired connections, 3Com Ethernet switches (100Mbps) are used.
- For the wireless tests, 802.11 USB devices (TP-Link AC600, Archer T2U) are used. The tests are carried out at 5.56 GHz in order to reduce the potential interferences with other devices operating in the 2.4 GHz band. The MTU is set to 2304 bytes, which allows a higher amount of packets to be multiplexed.
- D-ITG traffic generator [23] is employed in the *generator* and the *receiver*.

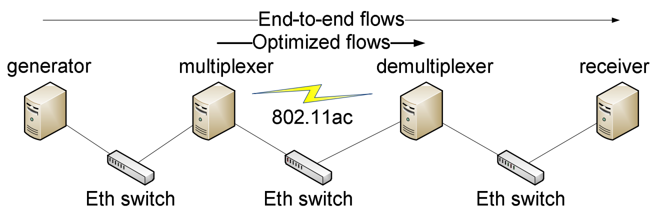


Fig. 10. Test scenario

b) Traffic traces: Two kind of traffic traces have been generated with D-ITG: in tests requiring a fixed size (e.g. small UDP packets, RTP flows), the standard options of the traffic generator are used. For generating variable size traffic, we have extracted the packet size and inter-packet time distributions from the trace in [2], so as to preserve the packet size distribution of real Internet traffic. When required, the inter-packet time has been scaled in order to reduce the bandwidth to a target amount.

B. Modification of the Traffic Profile with Multiplexing

As explained in Section V, multiplexing can achieve significant reductions in terms of packets per second. In this subsection we have used the Internet trace [2], scaled down to 8 Mbps, and sent it through the multiplexer, using different values for the size limit of the multiplexed packets.

Fig. 11 presents the packet size of 15 sec. of the Internet trace. It can be seen that a thick line corresponding to small packets (40-60 bytes), and two lines of big packets (1450 and 1500) are present, which correspond to the peaks in Fig. 1.

If we multiplex the small packets (using 200 bytes as the size limit), we obtain the traffic distribution shown in Fig. 12, where red "x" dots correspond to the multiplexed packets. It can be observed that small packets are multiplexed into bigger ones. In this case, 7,908 packets travel into 395 multiplexed bundles. The rest of the packets (10,932) are not multiplexed.

Obviously, if we increase the size limit, more packets will be sent to the multiplexer, so the amount of packets will be further reduced. The reduction in terms of packets per second can be up to 50 % if the size limit is higher. However, Fig. 13 shows that the improvement is small, since the saving is mainly produced by the multiplexing of the very small packets. It can be observed that the savings are similar to those predicted analytically (Fig. 8).

C. Bandwidth Savings

a) RTP using the wired bottleneck: Five concurrent RTP sessions, using different codecs, are sent through the multiplexer (Fig. 14) using the wired bottleneck. It can be observed that the maximum savings are obtained for the codecs with the highest header-to-payload ratio. In this case, the saving is mainly produced by header compression. Multiplexing allows the sending of compressed headers end-to-end, and maintains the tunneling overhead low. The compression ratio can be up to 46% for certain codecs.

b) Small UDP traffic using the wireless bottleneck: A flow of 15,000 small packets per second (60 bytes) is sent through the multiplexer (7.2 Mbps at IP level). The bottleneck is an 802.11ac link at 9 Mbps, but the *airtime* drawback makes it impossible for the flow to traverse the link: in the first column of Fig. 15 *(native)*, it can be seen that when multiplexing is not activated, packet loss is very high. The cause is that the MAC protocol has to act before each packet is sent, so airtime efficiency is really poor, taking into account that the packets are only 60 bytes long.

However, if packets are multiplexed (blue left columns in Fig. 15), it can be observed that packet loss gets reduced dramatically (if the number of multiplexed packets is higher than 1). In this case, the improvement of the *airtime* drawback achievable with multiplexing is clear: significant savings are obtained because the number of times MAC mechanisms have to be employed is reduced, as a consequence of the packet per second reduction. In addition, if ROHC is activated (patterned right columns), the saving becomes significantly higher, but in this case the reason is the size reduction achieved by header compression of small packets.

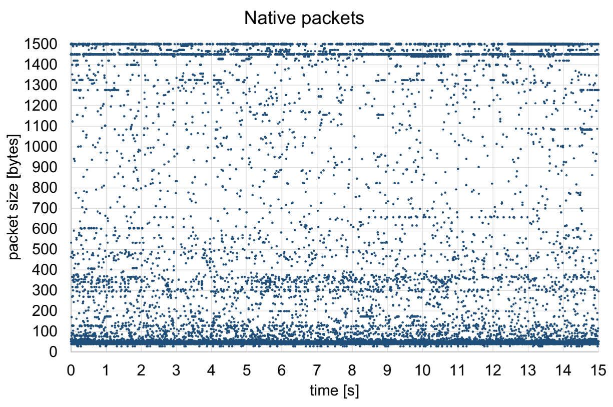


Fig. 11. Packet size distribution of the original Internet traffic trace [2]

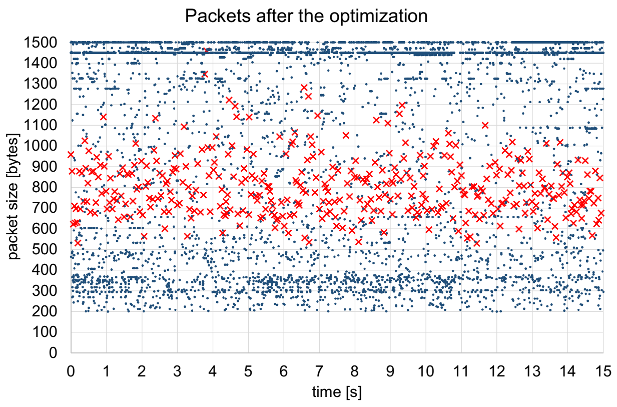


Fig. 12. Packet size distribution of the Internet traffic trace, once packets smaller than 200 bytes have been multiplexed

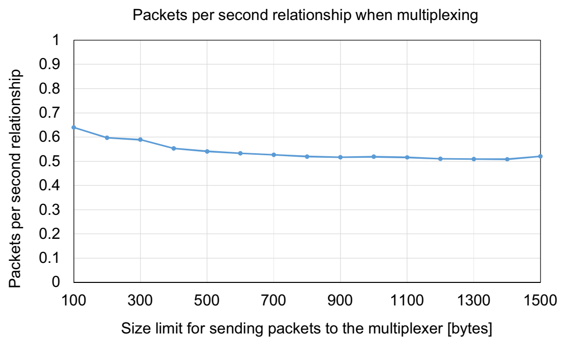


Fig. 13. Relationship between the amount of packets per second in the original and the optimized traces

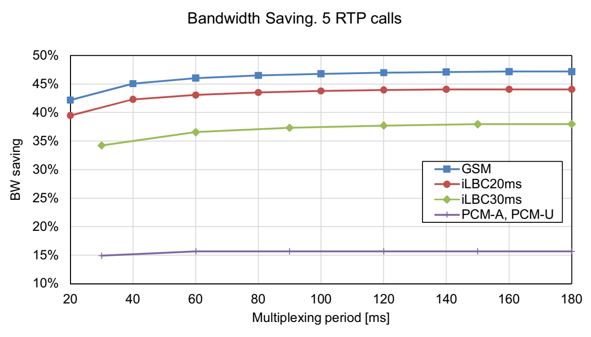


Fig. 14. Bandwidth savings for VoIP RTP traffic in a wired connection

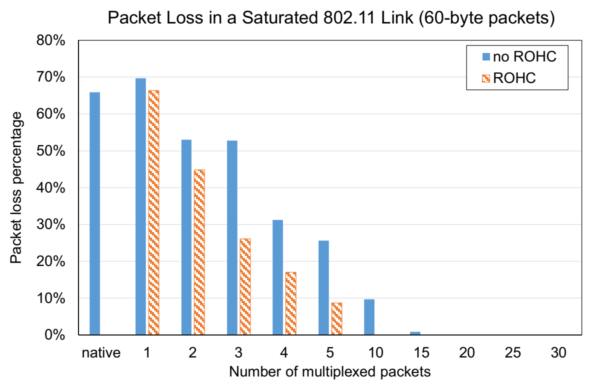


Fig. 15. Packet loss reduction in a saturated wireless link

VIII. Conclusions

This paper has presented Simplemux, a traffic optimization protocol aimed to improve efficiency in wired and wireless networks managing high rates of small packets with limited resources. The design of Simplemux has been presented, and some deployment scenarios have been detailed. An analysis of the improvements achievable with Simplemux has been carried out, and a battery of tests has been deployed using a real implementation. The savings in terms of bandwidth are significant (up to 46%) when small packets are multiplexed. A public Internet traffic capture has been analyzed, showing a high amount of small packets. In this case, the reduction is mainly achieved in terms of packets per second (up to 50%). In wireless networks, packet grouping results in a significantly improved use of air time, as shown in the tests.

References

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

[2] The CAIDA UCSD equinix-chicago- 20150219-130000, https://data. caida.org/datasets/passive-2015/equinix-chicago/20150219-130000. UTC/equinix-chicago.dirA.20150219-125911.UTC.anon.pcap.gz

[3] 3Com® Switch 4200 Family. Data Sheet, available at http:// ftp.maxdata.de/Accessories/Connectivity/3Com/Datasheets/200806_3C om_OfficeConnect_4226T.pdf

[4] Intel White Paper "Impact of the Intel Data Plane Development Kit (Intel DPDK) on Packet Throughput in Virtualized Network Environments," available at https://networkbuilders.intel.com/docs/ IntelDPDK_PacketThroughputVirtualNetwork.pdf

[5] 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, 2nd Quarter 2011.

[6] D. Murray, T. Koziniec, K. Lee, and M. Dixon, "Large MTUs and internet performance," in High Performance Switching and Routing (HPSR), IEEE 13th International Conference on, pp. 82-87, 2012.

[7] IEEE 802.11n, IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput, IEEE Std 802.11n-2009, 2009.

[8] J. Saldana, "Simplemux. A generic multiplexing protocol," draft-saldana-tsvwg-simplemux-03, Jul. 2015, available at http://datatracker. ietf. org/ doc/ draft-saldana-tsvwg-simplemux/

[9] Internet Assigned Numbers Authority (IANA) Assigned Internet Protocol Numbers, available at http://www.iana.org/assignments/ protocol-numbers/protocol-numbers.xhtml

[10] IEEE 802.11ac, IEEE Draft Standard for IT - Telecommunications and Information Exchange Between Systems - LAN/MAN - Specific Requirements - Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications - Amd 4: Enhancements for Very High Throughput for operation in bands below 6GHz, June 2012.

[11] P. Cameron, D. Crocker, D. Cohen, J. Postel, RFC1692, Transport Multiplexing Protocol (TMux), Aug. 1994.

[12] R. Pazhyannur, I. Ali, C. Fox, RFC3153, PPP Multiplexing, Aug. 2001.

[13] B. Thompson, T. Koren, D. Wing, RFC4170, Tunneling Multiplexed Compressed RTP (TCRTP), Nov. 2005.

[14] B. Ginzburg, A. Kesselman, "Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n," Sarnoff Symposium, 2007 IEEE , vol., no., pp.1,5, April 30 2007-May 2 2007.

[15] T. Selvam, S. Srikanth, "A frame aggregation scheduler for IEEE 802.11n," Communications (NCC), 2010 National Conference on , vol., no., pp.1,5, 29-31 Jan. 2010.

[16] 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, 53-58 , July 2009.

[17] K. Pentikousis, E. Piri, J. Pinola, F. Fitzek, T. Nissilä, I. Harjula, "Empirical evaluation of VoIP aggregation over a fixed WiMAX testbed," In Proc. 4th international Conference on Testbeds and Research infrastructures For the Development of Networks & Communities. Innsbruck, Austria, Mar. 2008.

[18] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, D. Wing, M. A. 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," IEEE Comm Magazine, Vol. 51 n. 11 pp 127-136, Nov 2013.

[19] J. Saldana *et al,* "Alternative Network Deployments. characterization, technologies and architectures," draft-irtf-gaia-alternative-network-deployments-01, Jul 2015, , available at http://datatracker.ietf.org/ doc/draft-irtf-gaia-alternative-network-deployments/

[20] L. Cerda-Alabern, "On the topology characterization of Guifi.net," Proceedings Wireless and Mobile Computing, Networking and Communications, IEEE 8th International Conf, pp. 389-396, 2012.

[21] C. Rey-Moreno, I. Bebea-Gonzalez, I. Foche-Perez, R. Quispe-Taca, L. Linan-Benitez and J. Simo-Reigadas, "A telemedicine WiFi network optimized for long distances in the Amazonian jungle of Peru," Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition, ExtremeCom '11 ACM, 2011.

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

[23] A. Botta, A. Dainotti, A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," Computer Networks (Elsevier), 2012, Volume 56, Issue 15, pp 3531-3547.