Enhancing Throughput Efficiency via Multiplexing and Header Compression over LISP Tunnels

Jose Saldana[1], Luigi Iannone[2], Diego R. Lopez[3], Julián Fernández-Navajas[1], José Ruiz-Mas[1]
[1]Communication Technologies Group (GTC) - Aragon Institute of Engineering Research (I3A)
Dpt. IEC. Ada Byron Building. EINA, University of Zaragoza, 50018 Zaragoza, Spain
e-mail: {jsaldana, navajas, jruiz}@unizar.es

[2]Telecom-ParisTech
75214 PARIS Cedex 13
e-mail: Luigi.Iannone@Telecom-ParisTech.fr

[3]Telefonica I+D
Don Ramon de la Cruz 82-84, 28006 Madrid, Spain
e-mail: diego@tid.es

Abstract— **This article explores the possibility of using traffic optimization techniques within the context of the LISP (Locator/Identifier Separation Protocol) framework. These techniques use Tunneling, Multiplexing and header Compression of Traffic Flows (TCMTF) in order to save bandwidth and to reduce the amount of packets per time unit. Taking into account that encapsulation is necessary in LISP, bandwidth can be drastically reduced in flows using small packets, which are typical of many real-time services. The ability of the LISP framework to manage the signaling of TCMTF options is also studied. An analytical expression of the savings, as a function of the different header sizes, is devised and used to calculate the maximum expected savings. Different services and scenarios of interest are identified, and this allows the consideration of tests with real traffic traces, showing the savings as a function of the multiplexing period, and demonstrating that the additional delays can be acceptable for real-time services.**

Keywords-LISP; multiplexing; header compression; real-time

I. Introduction

In the last years we are witnessing the rise of new real-time services, which divide the information into small packets (some tens of bytes), sent at a high rate, in order to provide interactivity to the user. In fact, today already almost half of the packets flowing in the Internet are less than 100 bytes long [1]. Thus, the overhead introduced by protocol headers may become extremely significant for services using tiny packets, and this has increased the interest on exploring different ways for improving throughput efficiency.

More specifically, a proposal has been presented within the Transport Area Working Group of the Internet Engineering Task Force (IETF), looking at ways to multiplex, compress, and tunnel such kind of traffic (TCMTF - Tunneling Compressed Multiplexed Traffic Flows [2]). Leveraging on header compression, a number of multiplexed packets are sent end-to-end by the use of encapsulation mechanisms. Different options are considered at the compression and multiplexing layer. The first versions of the proposal only considered L2TPv3 (Layer 2 Tunneling Protocol Version 3) for the encapsulation layer. However, LISP (Locator/Identifier Separation Protocol) is being considered as an additional scenario of application.

In 2006, the IAB's Routing and Addressing Workshop [3] highlighted the need for new architectures which would overcome the problems the Internet will face in the next years; mainly concerning the scalability of the routing system. Among the different proposals, LISP [4] is getting a growing interest from the Industry. LISP distinguishes between two address spaces (Fig. 1): a “Routing Locator” (RLOC) space, used to address stub networks' border routers; and “Endpoint Identifiers” (EIDs) which are assigned to hosts inside the stub networks. A stub network (e.g., the one of a company or a campus) is only able to route packets to and from itself, not providing any transit service for other networks.

Border routers are located between the EID and the RLOC spaces, hence, being able to find the route for traffic flows travelling to a different domain [4]. When they receive a packet, they perform a “map and encap” process: they first obtain a mapping between the destination EID and its RLOC(s), and then they encapsulate the packet, adding three headers: LISP, UDP and an external IP header that corresponds to the RLOC space. The packet is then sent through the RLOC space and, when it arrives to the edge router of the destination domain, the encapsulation header is removed. In the case of IPv4, the different headers account for 36 bytes added to each packet before being transmitted through the RLOC space. Such overhead is not significant for the so-called “traditional services”, e.g., web browsing, e-mail or FTP, since they tend to use packets corresponding to the Maximum Transfer Unit (MTU), which is normally set to 1,500 bytes (for common networks access technologies, e.g., Ethernet, WiFi, etc.), but can go up to 9,000 bytes when jumbo frames are used.

It can be frequent that a number of machines in the same stub network communicate simultaneously with a set of hosts belonging to another stub network. In those cases, because encapsulation is already required in the RLOC space (Fig. 1), multiplexing packets travelling between the same pair

978-1-4673-5753-1/13/$31.00 ©2013 IEEE

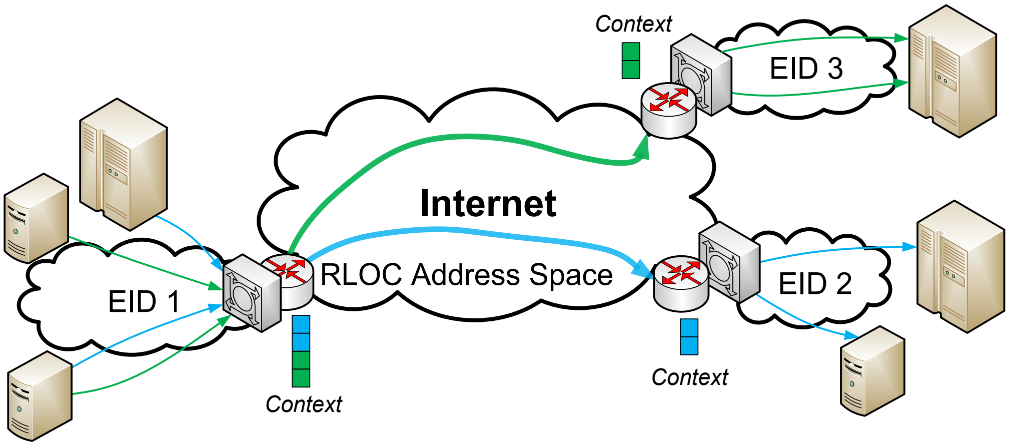


Figure 1. Encapsulating, multiplexing and compressing among different stub networks.

of networks can be a way of improving efficiency, since the encapsulation overhead would then be shared by a number of packets.

In the general context of TCMTF, besides multiplexing, header compression schemes [5] are able to save even more bandwidth in long-term flows. These algorithms, by the use of a *context,* shared between the origin and the destination, avoid the sending of some header fields which are constant for all the packets in the flow. From this point of view the LISP framework offers another added value consisting in its signaling mechanism able to carry meta-information, which can be useful for the negotiation of the different TCMTF options between the two extremes of the tunnel. The core contribution of this work lay exactly on the above-mentioned points: putting together LISP and TCMTF and exploring the synergies produced by the combination of both technologies.

The rest of the paper is organized as follows: the context of this work and the scenarios of application are first presented, followed by the signaling method. Then, the savings obtained in the tests are presented, including results from both an analytical model and real traces analysis, before ending the paper with the main conclusions.

II. Context and Scenarios of Application

This section overviews some representative scenarios where traffic optimization techniques provide significant savings, in particular when combined with LISP. The section also summarizes the related work.

One of the first real-time services being deployed in the Internet was VoIP (Voice over IP). A typical scenario of application can be a company with a number of offices, in which it can be frequent that a number of calls are simultaneously established between the same pair of offices. In this context, the IETF approved in 2005 a multiplexing method [6], which was mainly thought for optimizing VoIP flows. Other proposals for multiplexing VoIP were GeRM [7] and the scheme proposed by Sze *et al*. [8].

Other works [9] considered the multiplexing of non real time flows, namely First Person Shooter (FPS) games, obtaining bandwidth savings of about 30%. The saving can be even higher for Massively Multiplayer Online Role Playing Games (MMORPG) using TCP [10], since packets have a 40 bytes TCP/IP header, and generate a high rate of ACKs with no payload [11].

In these services, a multiplexer can be placed in certain locations of the aggregation network where a number of flows can be bundled together [12]. If the number of flows is high enough, significant bandwidth savings can be obtained, while adding short delays.

Finally, in addition to VoIP, FPS and MMORPG, another scenario is represented by popular Web or CDN (Content Delivery Networks) servers accessed by a high number of users. In this case, ACKs travelling to a certain Web/CDN EID network can be multiplexed in order to share the encapsulation header in the RLOC space. However, header compression cannot be used in this scenario, since we do not have stable flows between pairs of computers, generating high amounts of packets per second.

III. Multiplexing/Compression Signaling

This section presents how, besides the encapsulation features, the LISP framework can be used to signal multiplexing and compression capabilities.

As previously stated, LISP is based on a map-and-encap approach, where mappings binding EIDs to RLOCs are used to perform encapsulation. These mappings are stored on the border routers in a temporary cache. Indeed, border routers do not store all existing mappings, rather they retrieve only the mappings necessary for their ongoing traffic from a general service usually called *mapping system*, which is part of the LISP architecture. LISP has an open design allowing the use of several different mapping systems. Depending on the specific LISP deployment scenario considered (i.e., private vs. public) the mapping system can be a privately deployed system, only accessible by partners and/or different sites of the same virtual private network, or the public mapping system [13]. The latter is currently based on the LISP-DDT technology (LISP Delegated Database Tree [14]), which is a DNS-like hierarchy of servers managing different parts of the EID space. Border routers query such mapping system to obtain mappings related to a specific EID (or a whole EID-Prefix of a stub network).

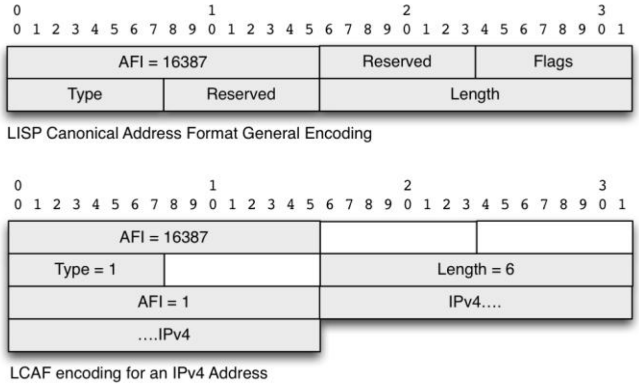
In the context of the present work it is not relevant the specific architecture of the mapping system, rather it is important to understand how the LISP signaling (i.e., mappings and their publication and retrieval) can be enriched to carry meta-information concerning whether or not to multiplex packets destined to a block of EIDs and whether or not and how to compress their headers. To this end, the LISP Canonical Address Format (LCAF) [15] can be used.

LCAF offers the possibility to encode different address families (from layer 2 MAC addresses up to geo location coordinates) in a standard (i.e., canonical) format, as shown in Fig. 2. Basically, the LCAF mechanism wraps addresses in a LISP AFI (Address Family Identifier) record encoding the type and length of addresses and few signaling flags.[1] Furthermore, LCAF allows defining opaque keys, basically allowing transporting any type of meta-information, and can be recursive, i.e., an LCAF encoded address can contain another LCAF encoded address. The last two properties are key for the flexibility of LCAF, since they allow creating complex sequences of addresses and meta-information.

Because of the above-mentioned flexibility, it is easy to understand that LCAF allows effective encoding of the information concerning which traffic has to be multiplexed over a LISP tunnel, based on several parameters (e.g., source and destination addresses, ToS, application, etc.). Other meta-information (e.g., the header compression mechanism to use) can also be associated. It is out of scope of the present paper to provide the exact encoding of the different TCMTF mechanisms in an LCAF Record, since it has no influence on the gains provided by the LISP framework and on the analysis provided. The key point is that such a framework does not only provide a standard format to express what to multiplex and compress, but also provides a signaling mechanism thanks to the already defined mapping system. This signaling mechanism can be applied to start or adapt the multiplexing and compression procedures on demand, according to traffic or other conditions at each endpoint, providing dynamic capacity sharing between the two EIDs. Actually, by leveraging on the LCAF standard mechanism, it is possible to use the already deployed LISP-DDT mapping system as signaling infrastructure, hence, providing ready to use signaling resources at no further costs (either design, deployment, or operational costs). The use of LISP introduces an initial query delay during which, when no mapping is locally available, TCMTF techniques cannot be applied, however, packets can be still be forwarded natively, hence causing no harm.

IV. Expected Bandwidth Savings

The present section first introduces an example that clearly shows the savings that can be obtained for a real-time service (VoIP). Then, a theoretical analysis of the expected bandwidth savings is presented. Finally, simulations are used to confirm the results.



LISP Canonical Address Format General Encoding

LCAF encoding for an IPv4 Address

Figure 2. LISP LCAF Record encoding (top) and example of IPv4 address encoding (bottom). The Address Family Identifier (AFI) 16387 has been allocated by IANA to the LISP architecture and protocols, while AFI 1 indicates the IPv4 address family as for RFC 1700 [16].

A. Example of the expected savings

As an example of the savings that can be achieved, Fig. 3 shows the scheme of traffic optimization applied to VoIP for IPv4, using G729a codec (with two samples per packet). The first row shows four native packets encapsulated using LISP and IPv4. It can be observed that 36 bytes have to be added to each packet, corresponding to the IP/UDP/LISP headers (*TH* - Tunnel Header). Since the Native Header (*NH*) is of 40 bytes, a total amount of 76 bytes of headers are required in order to transport 20 bytes of payload (*P*).

The next row shows the result of multiplexing the four packets. It can be observed that only one encapsulation header is now required, which is shared by all of the packets. A PPP header (*MH*) of one byte is also added to *TH.* Two additional bytes are required at the beginning of each packet, corresponding to the PPPMux separator (*SH* - Separator Header). As shown in the figure, the saving is significant, although only four packets are considered.

The third row shows the result of using header compression, which can reduce the original IP/UDP/RTP headers to 4 or 5 bytes (*CH* - Compressed Header), thus reducing packet size dramatically.

B. Theoretical analysis

The Bandwidth Saving (*BS*) for some kinds of traffic where optimization is especially interesting can be expressed analytically by the following formula:

$$BS = \frac{BW_{native} - BW_{optmized}}{BW_{native}} = 1 - \frac{BW_{optmized}}{BW_{native}} \qquad (1)$$

Where BW_{native} is the bandwidth of the native traffic and $BW_{optimized}$ is the bandwidth after compression, multiplexing, and encapsulation have been applied. By using the parameters defined in Sec. IV.A for Fig. 3, it is possible to obtain the bandwidth requirements of *N* native packets arrived in a time interval *T*:

$$BW_{native} = N\,(\,TH + NH + P\,)\,/\,T \qquad (2)$$

[1] A detailed description of the purpose and use of the different flags of an LCAF Record is out of scope of the analysis provided in the present paper. The interested reader can refer to [15].

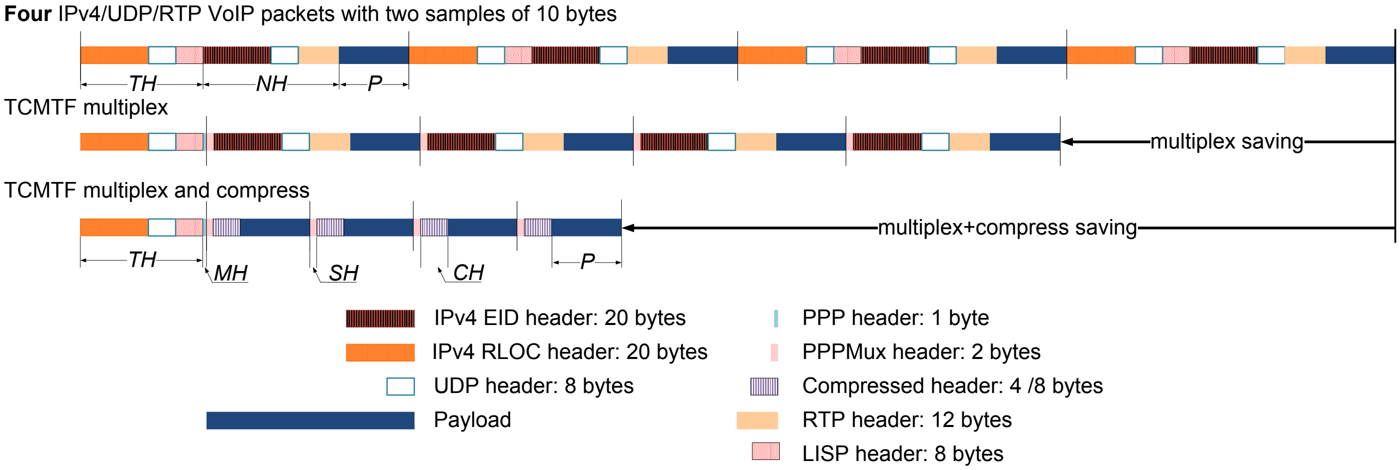

Four IPv4/UDP/RTP VoIP packets with two samples of 10 bytes
TH
NH
P
TCMTF multiplex
multiplex saving
TCMTF multiplex and compress
multiplex+compress saving
TH
MH
SH
CH
P
IPv4 EID header: 20 bytes
IPv4 RLOC header: 20 bytes
UDP header: 8 bytes
Payload
PPP header: 1 byte
PPPMux header: 2 bytes
Compressed header: 4 /8 bytes
RTP header: 12 bytes
LISP header: 8 bytes

Figure 3. Schemes of native, multiplexed and multiplexed-compressed VoIP packets

TABLE I. VARIABLE VALUES FOR DIFFERENT SERVICES (BYTES)

			VoIP. G729a 2samples (RTP/UDP/IP)			**FPS *Counter Strike* (UDP/IP)**			**MMORPG *World of Warcraft* (TCP/IP)**			**ACKs (TCP/IP)**		
EID	**RLOC**	**LISP *TH***	***NH***	***CH***	***P***	***NH***	***CH***	***P***	***NH***	***CH***	***P***	***NH***	***CH***	***P***
IPv4	IPv4	36	40	6.4	20	28	4.25	41.1	40	8.72	8.74	40	40	0
IPv6	IPv4	36	60	6.4	20	48	4.25	41.1	60	8.72	8.74	60	60	0
IPv4	IPv6	56	40	6.4	20	28	4.25	41.1	40	8.72	8.74	40	40	0
IPv6	IPv6	56	60	6.4	20	48	4.25	41.1	60	8.72	8.74	60	60	0

The bandwidth amount when the same packets are optimized is:

$$BW_{optimized} = [\ TH + MH + N\,(\ SH + CH + P\ )\ ]\ /\ T \qquad (3)$$

Hence, the amount of achievable savings *BS* can be expressed as:

$$BS = 1 - \frac{TH + MH}{N(TH + NH + P)} - \frac{SH + NH + P}{TH + NH + P} + \frac{NH - CH}{TH + NH + P} \qquad (4)$$

It can be seen that the second term gets reduced with the increase of the number of multiplexed packets, which will mainly depend on the packets rate (i.e., the number of received packets per second that can be potentially optimized) and on the time period used to select the packets to be multiplexed together. The third term is constant for each service, and determines the value of the asymptote of the saving when only multiplexing is applied. The fourth term augments the saving depending on the efficiency of the compression algorithm used (represented by *NH-CH*).

Table I shows some examples of the asymptote values for different services. The values of *CH* may vary depending on the compressing algorithm. For *CH* and *P* we have used the same values obtained in previous works ([17] for VoIP; [9] for FPS; [10] for MMORPG). Even if we explore all IP encapsulation combinations (i.e., IPv4 over IPv4, IPv4 over IPv6, IPv6 over IPv4, and IPv6 over IPv6), as a first approach, we consider that IPv4 and IPv6 headers can be compressed to the same number of bytes. Indeed, although the size of the IP addresses is different, they are absent in all the compressed packets. For ACKs' multiplexing, no header compression is considered, since we are considering ACKs belonging to traditional services as web browsing or e-mail, which do not constitute a continuous flow between the same pair of machines.

In Fig. 4, the values of the asymptote for the different services are presented. The highest savings are obtained for VoIP and MMORPG, in which *NH* has a big value and the payload is small. FPS games also achieve savings above 50%, since the compressing algorithm presents a high compressing ratio. Finally, regarding ACKs, the saving is smaller for the simple reason that no compression is used, however, it is near 50% in all the combinations. In this case, since header compression is not used, IPv6 saves less bandwidth than IPv4.

C. Results using different multiplexing periods

While previous sections dealt with the maximum values for the expected bandwidth savings, this subsection presents some tests that illustrate the behavior of the second term of equation (4), hence, studying if the maximum saving can be reached, and in what conditions. Taking into account that header compression can be a task requiring an intensive use of the processor, and an amount of memory for storing the *context* of each flow [5], two possibilities have to be considered: using or not using header compression. In the latter case, the router would simply multiplex a number of packets inside a bundled one, by means of a LISP tunnel.

First, Fig. 5 shows the analytical results obtained with equation (4) for VoIP traffic, when different numbers of flows are multiplexed together. Four different asymptotes can be noted. If header compression is used, bandwidth saving with respect to the native use of LISP can be up to 70% for IPv4, and even higher if IPv6 is considered. These results corroborate the ones presented in Fig. 4. If, instead, header compression is not used, more than 30% of the bandwidth can still be saved. The saving gets near the asymptote above roughly 10 flows.

In order to test the traffic of a FPS game, the original packet sizes and inter-packet times obtained in real parties of *Counter Strike 1* [18] have been reproduced and introduced to a Matlab simulated multiplexer, which calculated the compressed packet trace. The results are shown in Fig. 6. Since only IPv4 is used, two asymptotes can be appreciated. In contrast with the VoIP case, now it is necessary to define a multiplexing period, since packets do not have a fixed cadence. All the packets arriving during that period will be included in the same bundle. It can be noted that a period of roughly 25 ms is enough for obtaining savings close to the asymptote. The added delay would be half the multiplexing period, which can be tolerable for players [19]. If compression is used, savings of 50% can be obtained, whereas if not used, the saving is roughly 30%.

The same kind of simulation has been carried out in order to obtain the savings for an MMORPG game (Fig. 7), namely *World of Warcraft*. The obtained savings are even higher, because these games use TCP packets with small payloads [10], and thus they also send a high volume of ACK packets. So the saving when compressing is really high, over 70%, and 40% can be reached without compression.

Finally, some results using equation (4) have been obtained regarding the multiplexing of a high volume of small ACK packets travelling from one stub network to another, i.e., the last scenario presented in Sec. II. As shown in Fig. 8, if the amount of ACKs is big enough, the bandwidth saving can be about 40% with respect to the native use of LISP.

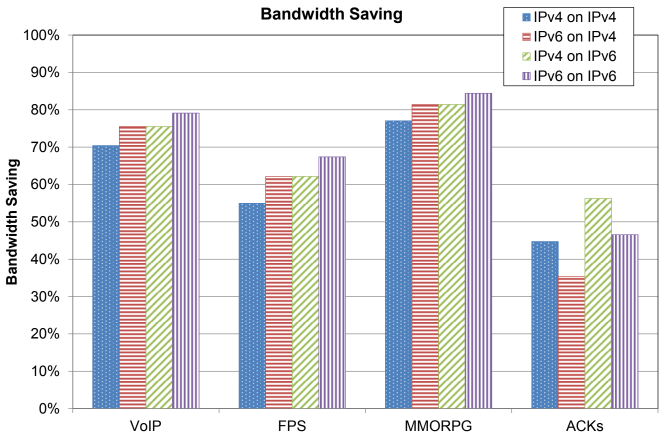


Figure 4. Bandwidth saving for different services

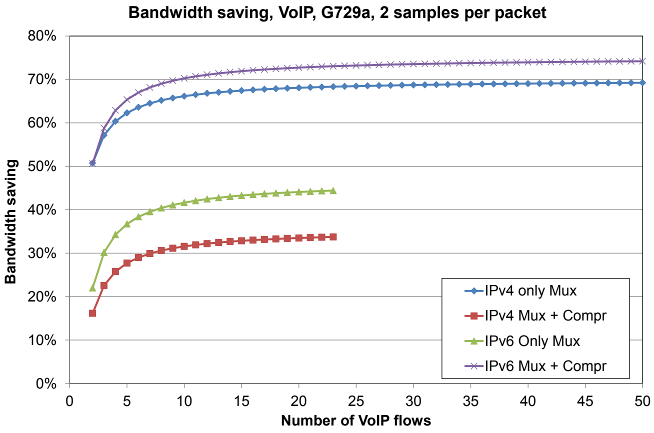


Figure 5. Bandwidth saving for G.729a VoIP flows

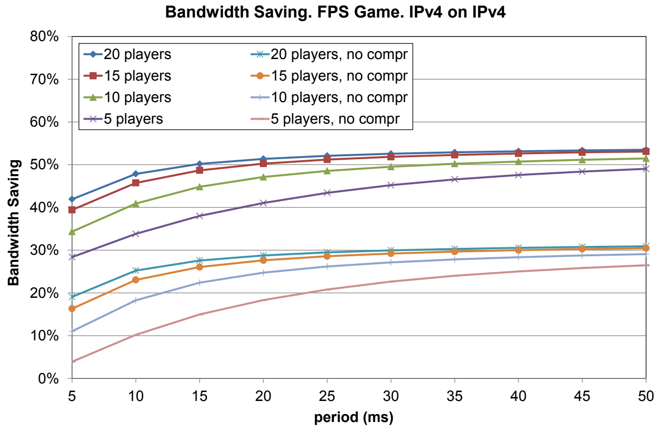


Figure 6. Bandwidh saving for a FPS game (*Counter Strike 1)*

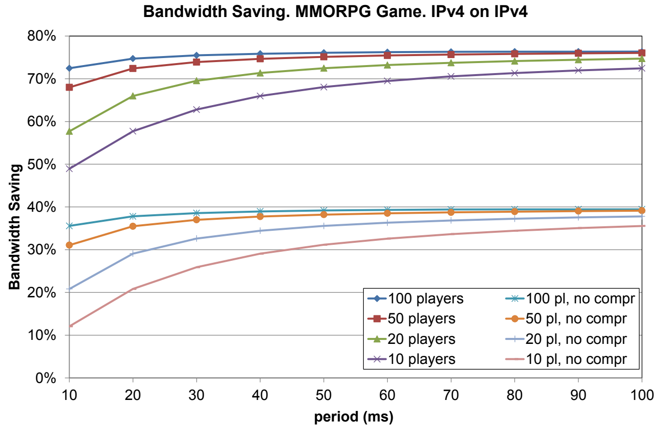


Figure 7. Bandwidh saving for an MMORPG game (*World of Warcraft)*

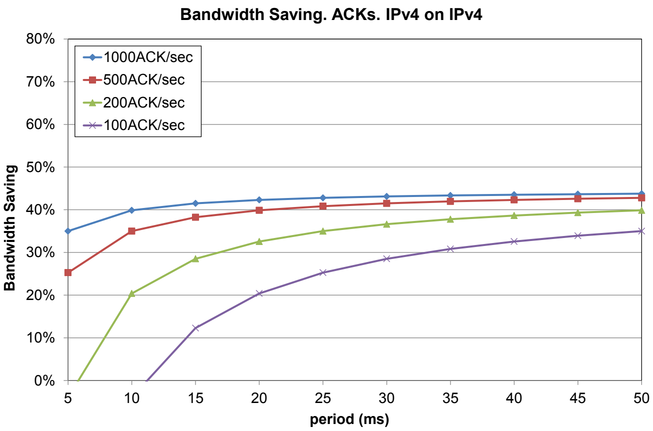


Figure 8. Bandwidth saving when multiplexing ACKs between EIDs

What has been shown by our results so far is that in all the cases the bandwidth saving when using header compression is high. In addition, even in the case of no header compression the saving is still significant. This is due to the fact that the original packets need to be LISP-encapsulated. The decision of whether or not to use compression mainly depends on the available resources in the two border routers at the tunnel end-points. All in all, multiplexing and compressing techniques have been demonstrated as a good complement to LISP, and may be useful to overcome the inefficiencies inherent to the use of tunnels, a problem that is stressed for services using small packets.

V. Conclusions and Future Work

Considering the raise of real-time services (which use small bandwidth-inefficient packets) the Internet is witnessing and the forthcoming all-IP Telco networks, it becomes fundamental to explore solutions able to increase throughput efficiency. In this paper, the possibility of using traffic optimization techniques (TCMTF) within the context of the LISP framework has been explored. This looks like a natural fit because of the ability of LISP to manage the signaling of TCMTF options.

The presented results clearly indicate that throughput efficiency can be highly improved by reducing headers overhead and per-packet processing. An analytical expression of the savings, as a function of the different header sizes, has been obtained, and the maximum savings have been calculated. Finally, tests with real traffic traces have been developed, showing the expected savings as a function of the multiplexing period.

The corroboration of these results via the deployment of TCMTF-able LISP border routers is the natural next step in future work. And this will allow to experiment with the possibilities that LCAF offers for TCMTF signaling.

Acknowledgements

This work has been partially financed by CPUFLIPI Project (MICINN TIN2010-17298), MBACToIP Project, of Aragon I+D Agency and Ibercaja Obra Social, European Social Fund in collaboration with the Government of Aragon, NDCIPI-QQoE Project Catedra Telefonica, Univ. of Zaragoza.

References

[1] D. Murray and T. Koziniec, “The State of Enterprise Network Traffic in 2012,” Proc. The 18th Asia-Pacific Conference on Communications (APCC 2012),” Jeju Island, Korea, Oct. 2012.

[2] J. Saldana et al, “Tunneling Compressed Multiplexed Traffic Flows (TCMTF),” draft-saldana-tsvwg-tcmtf-04, Jan. 2013.

[3] D. Meyer, L. Zhang, and K. Fall, “Report from the IAB Workshop on Routing and Addressing, ” IETF RFC 4984, Sep. 2007.

[4] J. N. Chiappa, “An Introduction to the LISP Location-Identity Separation System, ” draft-chiappa-lisp-introduction-01, Jul. 2012.

[5] E. Ertekin, C. Christou, “Internet protocol header compression, robust header compression, and their applicability in the global information grid,” IEEE Communications Magazine, vol. 42, 2004, pp. 106-116.

[6] B. Thompson, T. Koren, D. Wing, RFC 4170, “Tunneling Multiplexed Compressed RTP (TCRTP),” Nov. 2005.

[7] C. Perkins, RTP: Audio and Video for the Internet, Addison-Wesley Professional. 2003.

[8] H.P. Sze, S. C. Liew, J.Y.B. Lee and D.C.S.Yip, “A Multiplexing Scheme for H.323 Voice-Over-IP Applications,” IEEE J. Select. Areas Commun, Vol. 20, pp. 1360-1368, Sep. 2002.

[9] J. Saldana, J. Fernández-Navajas, J. Ruiz-Mas, J. I. Aznar, E. Viruete and L. Casadesus, “First Person Shooters: Can a Smarter Network Save Bandwidth without Annoying the Players?,” IEEE Communications Magazine, Consumer Communications and Networking Series, vol. 49, no. 11, pp. 190-198, Nov. 2011.

[10] J. Saldana, L. Sequeira, J. Fernandez-Navajas and J. Ruiz-Mas, “Traffic Optimization for TCP-based Massive Multiplayer Online Games,” Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems SPECTS 2012, Jul. 2012, Genoa, Italy.

[11] C. Wu, K. Chen, C. Chen, P. Huang and C. Lei, “On the Challenge and Design of Transport Protocols for MMORPGs,” Multimedia Tools and Applications, Vol. 45, No. 1, pp. 7-32, Oct 2009.

[12] J. Saldana, D. Wing, J. Fernandez-Navajas, J. Ruiz-Mas, M.A.M. Perumal, G. Camarillo, “Widening the Scope of a Standard: Real Time Flows Tunneling, Compressing and Multiplexing,” IEEE ICC 2012, Workshop on Telecommunications: from Research to Standards, Jun. 2012, Ottawa, Canada.

[13] The LISP Beta Network. Available online at: http://www.lisp4.net/beta-network/

[14] V. Fuller, D. Lewis, V. Ermagan, A. Jain, “LISP Delegated Database Tree,” draft-ietf-lisp-ddt-00, Oct. 2012.

[15] D. Farinacci, D. Meyer, J. Snijders, “LISP Canonical Address Format (LCAF),” draft-ietf-lisp-lcaf-00, Aug. 2012.

[16] J. Reynolds and J. Postel, “Assigned Numbers,” RFC 1700, Oct 1994.

[17] 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, Vol. 56, Issue 7, pp 1893-1919, May 2012.

[18] L. Stewart, P. Branch: HLCS, Map: dedust, 5 players, 13Jan2006. Centre for Advanced Internet Architectures SONG Database, http://caia.swin.edu.au/sitcrc/hlcs_130106_1_ dedust_5_fragment.tar.gz

[19] S. Zander and G. Armitage, “Empirically Measuring the QoS Sensitivity of Interactive Online Game Players,” Australian Telecom. Networks & Apps. Conference 2004 (ATNAC2004), Sydney, Australia, Dec. 2004.