TSVWG
Internet-Draft
Intended status: Standards Track
Expires: 30 June 2024

J. Saldana
CIRCE Technology Center

28 December 2023

Simplemux. A generic multiplexing protocol
draft-saldana-tsvwg-simplemux-12

Abstract

The high amount of small packets present in nowaday's networks results in a low efficiency, as the size of both the headers and the payload in these packets can be comparably large, often falling within the same order of magnitude. In some situations, multiplexing (i.e. aggregating) a number of small packets into a bigger one is desirable to improve the efficiency. For example, a number of small packets can be sent together between a pair of machines if they share a common network path. This may happen between machines in different locations or even inside a datacenter with a number of servers hosting virtual machines. The traffic profile can be shifted from small to larger packets, thus reducing the network overhead and the number of packets per second to be managed by intermediate devices.

This document describes Simplemux, a protocol able to encapsulate a number of packets belonging to different protocols into a single packet. Small headers (separators) are added at the beginning of each multiplexed packet, including some flags, the packet length and a "Protocol" field. The presence of this "Protocol" field allows it to aggregate packets belonging to any protocol (the "multiplexed packets"), in a single packet belonging to other (or the same) protocol (the "tunneling protocol").

To reduce the overhead, the size of the multiplexing headers is kept very low (it may be a single byte when multiplexing packets of small size).

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-saldana-tsvwg-simplemux/.

Discussion of this document takes place on the Transport Area Working Group (tsvwg) Working Group mailing list (mailto:tsvwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tsvwg/. Subscribe at https://www.ietf.org/mailman/listinfo/tsvwg/.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The high amount of small packets present in nowaday's networks results in a low efficiency, when the size of the headers and the payload fall within the same order of magnitude. In some situations, multiplexing (i.e. aggregating) a number of small packets into a bigger one is desirable to improve the efficiency. In these cases, a number of small packets can be sent together between a pair of machines if they share a common network path. This may happen between machines in different locations or even inside a datacenter with a number of servers hosting virtual machines. The traffic profile can be shifted from small to larger packets, thus reducing the overhead and the number of packets per second to be managed by intermediate devices.

This document describes Simplemux, a protocol able to encapsulate a number of packets belonging to different protocols into a single packet.

Simplemux generic: it includes a "Protocol" field, so it can be used to aggregate a number of packets belonging to any protocol, in a single packet belonging to other (or the same) protocol.

In this document we will talk about the "multiplexed" protocol, and the "tunneling" protocol, being Simplemux the "multiplexing" protocol. The "external header" will be the one of the "tunneling" protocol. In this document, we will also refer to the Simplemux header with the terms "separator," "Simplemux separator" or "mux separator". In the figures we will also use the abbreviation "Smux". These elements are presented in Figure 1.

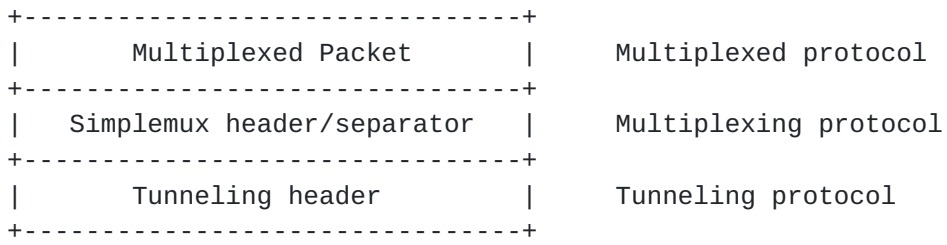


Figure 1

A Simplemux packet will have the structure shown in Figure 2.

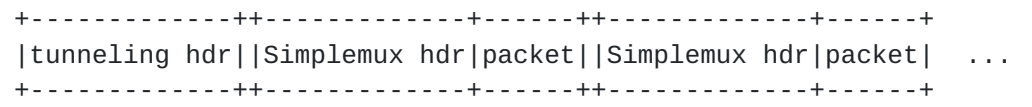


Figure 2

As an example, if a number of small IPv6 packets have to traverse an IPv4 network, they can be multiplexed together into a single IPv4 packet. In this case, IPv4 will be the "tunneling" protocol and IPv6 will be the "multiplexed" protocol. The IPv4 header is called in this case the "tunneling" or the "external" header. The scheme of this packet will be(the "Protocol" field inside the Simplemux header will be 41 (IPv6), according to IANA Assigned Internet Protocol Numbers):

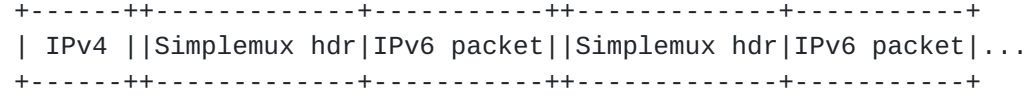


Figure 3

Another example is the tunneling of Ethernet frames between different networks using UDP. In this case, IPv4/UDP will be the "tunneling" protocol and Ethernet will be the "multiplexed" protocol. The IPv4/UDP header is called in this case the "tunneling" or the "external" header. The scheme of this packet will be (the "Protocol" field inside the Simplemux header will be 143 (Ethernet), according to IANA Assigned Internet Protocol Numbers):

Figure 4

Simplemux has two flavors:

* Fast: all the separators are 3-byte long, and all have the same structure, so it sacrifices some compression on behalf of speed.

* Compressed: it tries to compress the separators as much as possible. For that aim, some single-bit fields are used.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Benefits of multiplexing

The benefits of multiplexing are:

- Tunneling a number of packets together. If a number of packets have to be tunneled through a network segment, they can be multiplexed and then sent together using a single external header. This will avoid the need for adding a tunneling header to each of the packets, thus reducing the overhead.

- Reduction of the amount of packets per second in the network. Network equipment has a limitation in terms of the number of packets per second it can manage, i.e. many devices are not able to send small packets back to back due to processing delay.

- Bandwidth reduction. The presence of high rates of tiny packets translates into an inefficient usage of network resources, so the overhead introduced by low-efficiency flows can be reduced. When combined with header compression, as done in TCRTP [RFC4170] multiplexing may produce significant bandwidth savings, which are interesting for network operators, since they may alleviate the traffic load in their networks.

- Energy savings: a lower amount of packets per second will reduce energy consumption in network equipment since, according to [Bolla], 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 [chab] on commercial routers corroborate this: tests 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.

Some tests measuring the benefits of Simplemux were published in [Saldana].

4. Existing multiplexing protocols

Different multiplexing protocols have been approved by the IETF in the past:

4.1. Tmux

TMux [RFC1692] is able to combine multiple short transport segments, independent of application type, and send them between a server and host pair. As stated in the reference, "The TMux protocol is intended to optimize the transmission of large numbers of small data packets. In particular, communication load is not measured only in bits per seconds but also in packets per seconds, and in many situation the latter is the true performance limit, not the former. The proposed multiplexing is aimed at alleviating this situation."

A TMux message appears as:

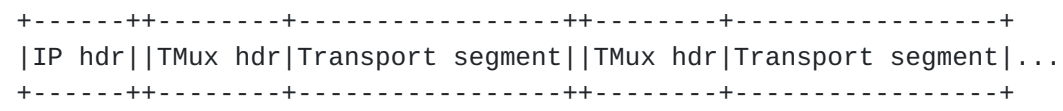


Figure 5

Therefore, the Transport Segment is not an entire IP packet, since it does not include the IP header.

TMux works "between a server and host pair," so it multiplexes a number of segments between the same pair of machines. However, there are scenarios where a number of low-efficiency flows share a common path, but they do not travel between the same pair of end machines.

4.2. PPPmux

PPPMux [RFC3153] "sends multiple PPP encapsulated packets in a single PPP frame. As a result, the PPP overhead per packet is reduced." Thus, it is able to multiplex complete IP packets, using separators.

However, the use of PPPMux requires the use of PPP and L2TP to multiplex a number of packets together, as done in TCRTP [RFC4170]. Thus, it introduces more overhead and complexity.

Using PPPMux, an IP packet including a number of packets appears as:

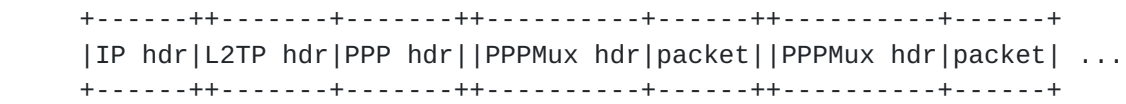


Figure 6

The scheme proposed by PPPMux is similar to the Compound-Frames of PPP LCP Extensions [RFC1570]. The key differences are that PPPMux is more efficient and that it allows concatenation of variable sized frames.

4.3. Advantages of Simplemux

The definition of a protocol able to multiplex complete packets, avoiding the need of using other protocols as e.g. PPP is seen as convenient. The multiplexed packets MAY belong to any protocol, since a "Protocol" field is added to each of them. Note that Ethernet frames can also be aggregated together.

5. Scenarios of interest

Simplemux works between a pair of machines. It creates a tunnel between an "ingress" and an "egress". They MAY be the endpoints of the communication, but they MAY also be middleboxes able to multiplex packets belonging to different flows. Different mechanisms MAY be used in order to classify flows according to some criteria (sharing a common path, kind of service, etc.) and to select the flows to be multiplexed and sent to the egress (see Figure 7).

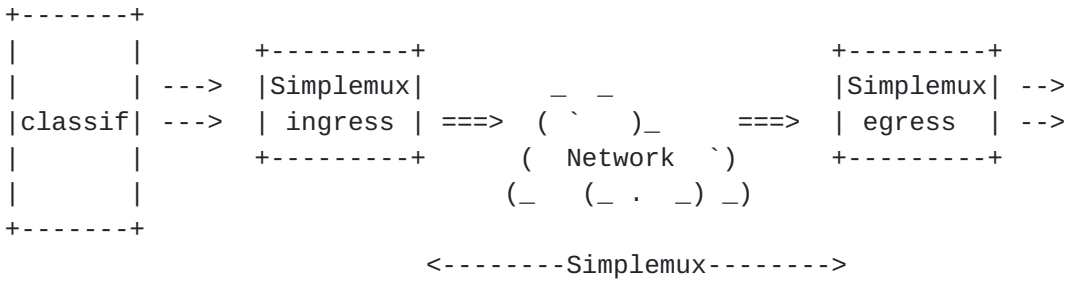


Figure 7

6. Protocol description

Simplemux has two flavors:

* Fast: all the separators are 3-byte long, and all have the same structure, so it sacrifices some compression on behalf of speed.

* Compressed: it tries to compress the separators as much as possible. For that aim, some single-bit fields are used.

6.1. Fast flavor

In Fast flavor, all the Simplemux headers have the same format, with two fields:

- Length (LEN, 16 bits) of the multiplexed packet (in bytes).

- Protocol (8 bits) of the multiplexed packet, according to IANA "Assigned Internet Protocol Numbers".

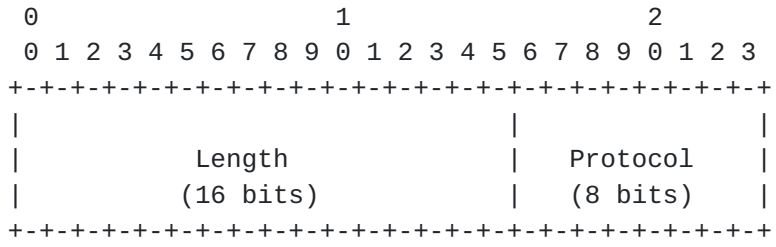


Figure 8

6.2. Compressed flavor

In Compressed flavor, the Simplemux header has two different forms: one for the "First Simplemux header," and another one for the rest of the Simplemux headers (called "Non-first Simplemux headers"):

6.2.1. First Simplemux header

In is placed after the tunneling header, and before the first multiplexed packet.

In order to allow the multiplexing of packets of any length, the number of bytes expressing the length is variable, and a field called "Length Extension" (LXT, one bit) is used to flag if the current byte is the last one including length information. This is the structure of a First Simplemux header:

- Single Protocol Bit (SPB, one bit) only appears in the first Simplemux header. It is set to 1 if all the multiplexed packets belong to the same protocol (in this case, the "Protocol" field will only appear in the first Simplemux header). It is 0 when each packet MAY belong to a different protocol.

- Length Extension (LXT, one bit) is 0 if the current byte is the last byte where the length of the first packet is included, and 1 in other case.

- Length (LEN, 6, 13, 20, etc. bits): This is the length of the multiplexed packet (in bytes). If the length of the multiplexed packet is less than 64 bytes (less than or equal to 63 bytes), the first LXT is 0 and the 6 bits of the length field are the length of the multiplexed packet. If the length of the multiplexed packet is equal or greater than 64 bytes, additional bytes are added. The first bit of each of the added bytes is the LXT. If LXT is set to 1, it means that there is an additional byte for expressing the length. This allows to multiplex packets of any length (see the next figures).

- Protocol (8 bits) of the multiplexed packet, according to IANA "Assigned Internet Protocol Numbers."

A First Simplemux header before a packet smaller than 64 (2^6) bytes will be 2 bytes long:

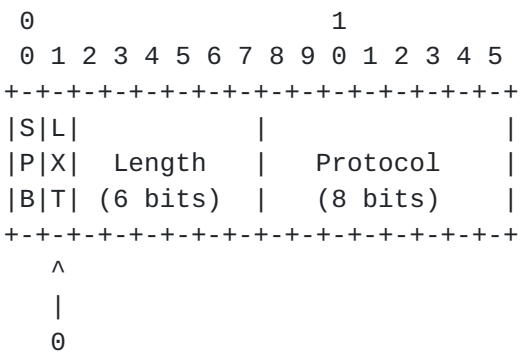


Figure 9

A First Simplemux header before a packet with a length greater or equal to 64 bytes, and smaller than 8192 bytes (2^13) will be 3 bytes long:

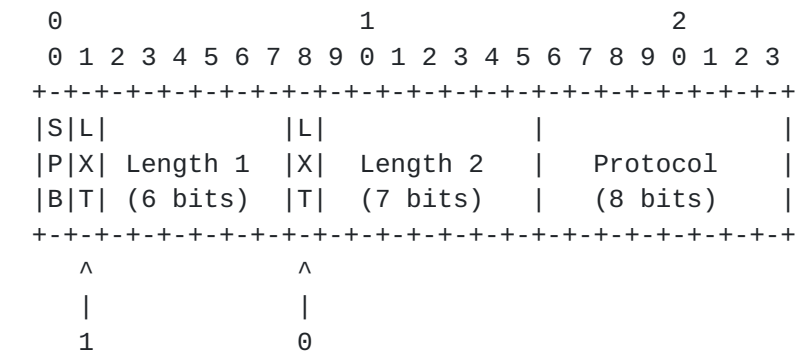


Figure 10

In this case, the length of the packet will be the number expressed by the concatenation of the bits of Length 1 and Length 2 (total 13 bits). Length 1 includes the 6 most significant bits and Length 2 the 7 less significant bits.

A First Simplemux header before a packet with a length greater of equal to 8192 bytes, and smaller than 1048576 bytes (2^20), will be 4 bytes long:

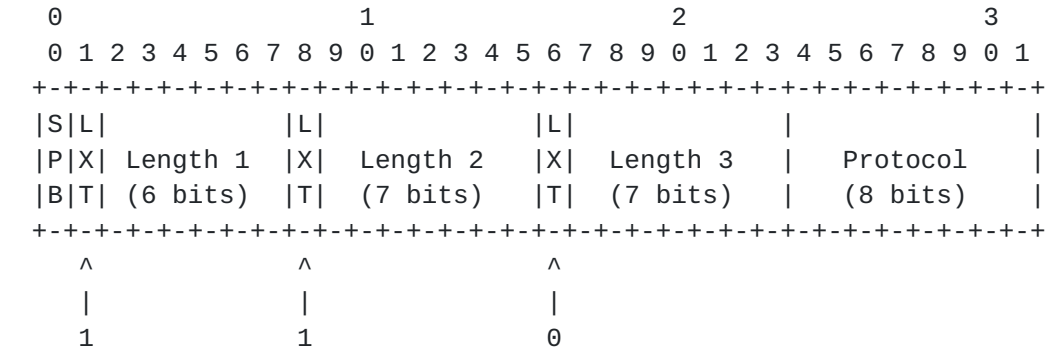


Figure 11

In this case, the length of the packet will be the number expressed by the concatenation of the bits of Length 1 , Length 2 and Length 3 (total 20 bits). Length 1 includes the 6 most significant bits and Length 3 the less 7 significant bits.

More bytes can be added to the length if required, using the same scheme: 1 LXT byte plus 7 bits for expressing the length.

6.2.2. Non-first Simplemux header

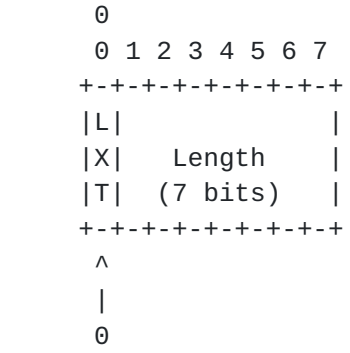
The Non-first Simplemux headers also employ a format allowing the multiplexing of packets of any length, so the number of bytes expressing the length is variable, and the field Length Extension (LXT, one bit) is used to flag if the current byte is the last one including length information. This is the structure of a Non-first Simplemux header:

- Length Extension (LXT, one bit) is 0 if the current byte is the last byte where the length of the packet is included, and 1 in other case.

- Length (LEN, 7, 14, 21, etc. bits): This is the length of the multiplexed packet (in bytes), not including the length field. If the length of the multiplexed packet is less than 128 bytes (less than or equal to 127 bytes), LXT is 0 and the 7 bits of the length field represent the length of the multiplexed packet. If the length of the multiplexed packet is greater than 127 bytes, additional bytes are added. The first bit of each of the added bytes is the LXT. If LXT is set to 1, it means that there is an additional byte for expressing the length. This allows to multiplex packets of any length (see the next figures).

- Protocol (8 bits) field of the multiplexed packet, according to IANA "Assigned Internet Protocol Numbers". It only appears in Non-first headers if the Single Protocol Bit (SPB) of the First Simplemux header is set to 1.

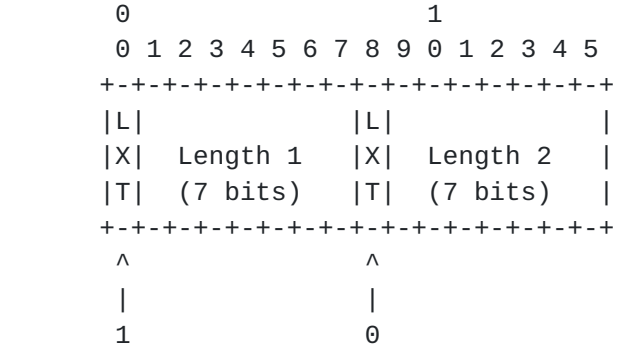
As an example, a Non-first Simplemux header before a packet smaller than 128 bytes, when the protocol bit has been set to 0 in the First header, will be 1 byte long:



SPB = 1 in the first header

Figure 12

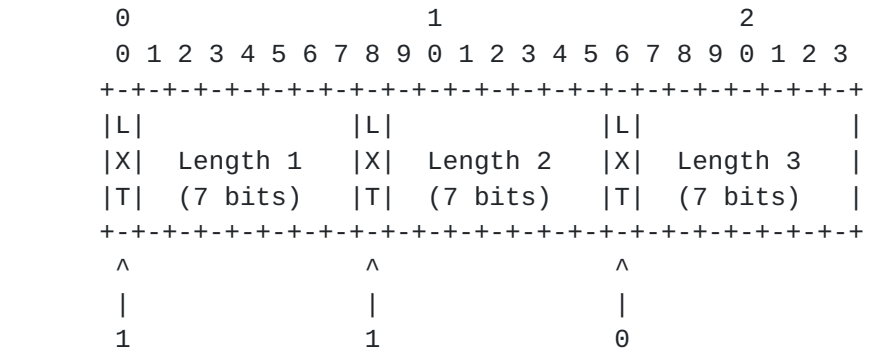
A Non-first Simplemux header before a packet witha a length greater or equal to 128 bytes, and smaller than 16384 (2^14), when the protocol bit has been set to 0 in the First header, will be 2 bytes long:



SPB = 1 in the first header

Figure 13

A Non-first Simplemux header before a packet with a length greater or equal to 16384 bytes, and smaller than 2097152 bytes (2^21), when the protocol bit has been set to 0 in the First header, will be 3 bytes long:



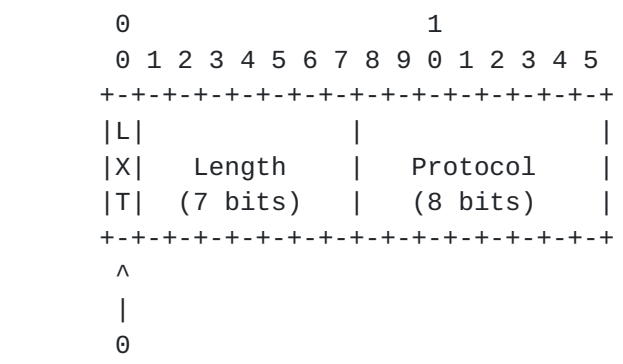
SPB = 1 in the first header

Figure 14

In this case, the length of the packet will be the number expressed by the concatenation of the bits of Length 1, Length 2 and Length 3 (total 21 bits). Length 1 includes the 7 most significant bits and Length 3 the 7 less significant bits.

More bytes can be added to the length if required, using the same scheme: 1 LXT byte plus 7 bits for expressing the length.

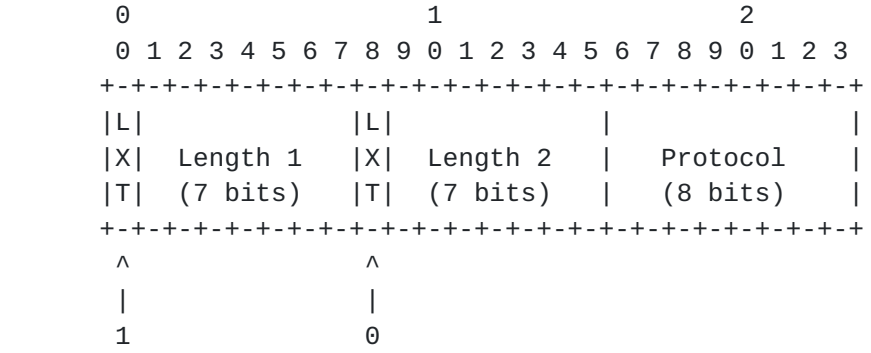
A Non-first Simplemux header before a packet smaller than 128 bytes, when the protocol bit has been set to 1 in the First header, will be 2 bytes long:



SPB = 0 in the first header

Figure 15

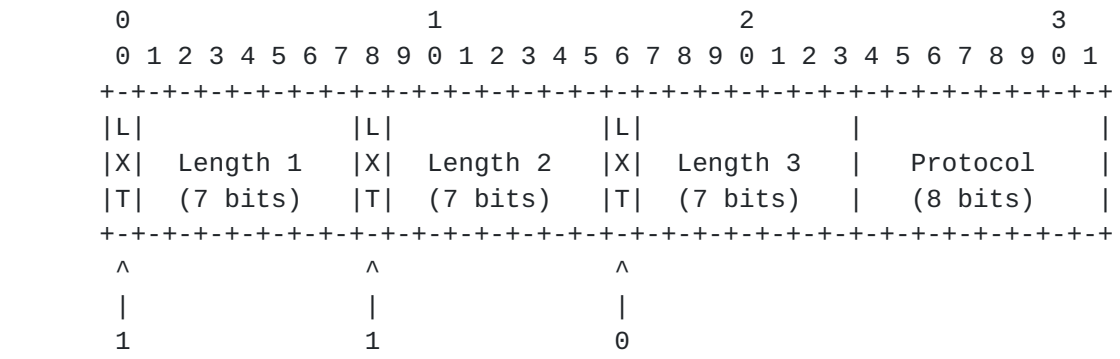
A Non-first Simplemux header before a packet with a length greater or equal to 128 bytes, and smaller than 16384 (2^14), when the protocol bit has been set to 1 in the First header, will be 3 bytes long:



SPB = 0 in the first header

Figure 16

A Non-first Simplemux header before a packet with a length greater of equal to 16384 bytes, and smaller than 2097152 bytes (2^21), when the protocol bit has been set to 1 in the first header, will be 4 bytes long:



SPB = 0 in the first header

Figure 17

In this case, the length of the packet will be the number expressed by the concatenation of the bits of Length 1, Length 2 and Length 3 (total 21 bits). Length 1 includes the 7 most significant bits and Length 3 the 7 less significant bits.

More bytes can be added to the length if required, using the same scheme: 1 LXT byte plus 7 bits for expressing the length.

Next, some examples of the whole bundles are presented

Case 1: All the packets belong to the same protocol: The first Simplemux header will be 2 or 3 bytes (for usual packet sizes), and the other Simplemux headers will be 1 or 2 bytes. For small packets (length < 128 bytes), the Simplemux header will only require one byte.

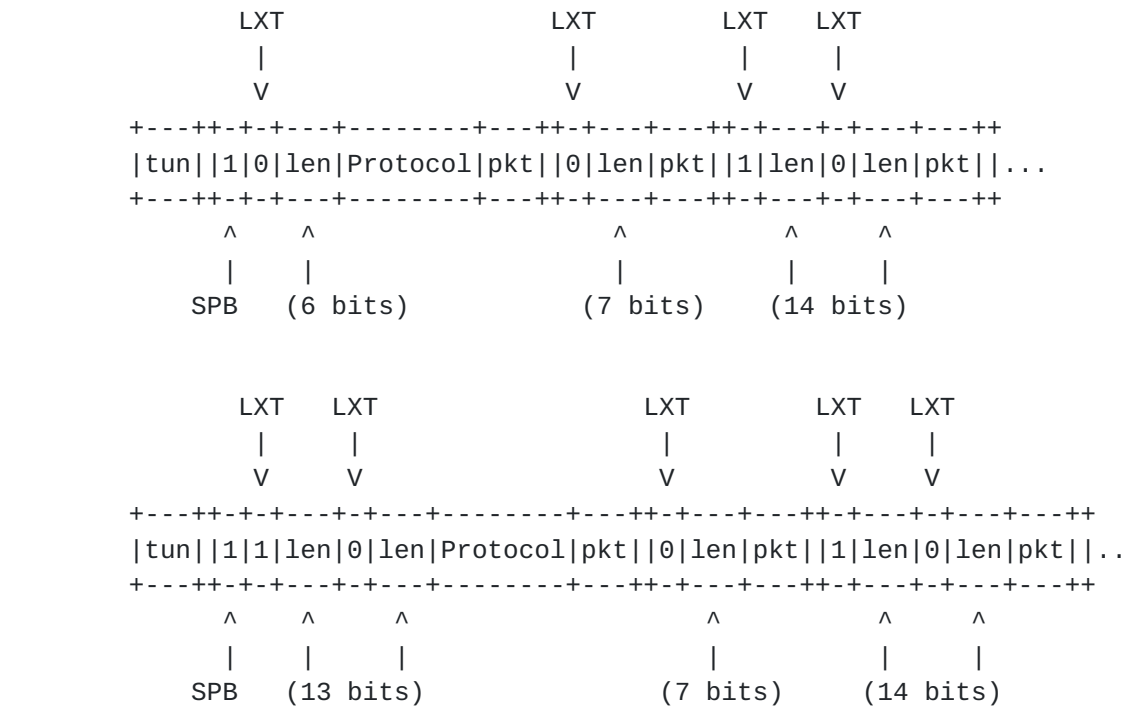


Figure 18

Case 2: Each packet MAY belong to a different protocol: All the Simplemux headers will be 2 or 3 bytes (for usual packet sizes).

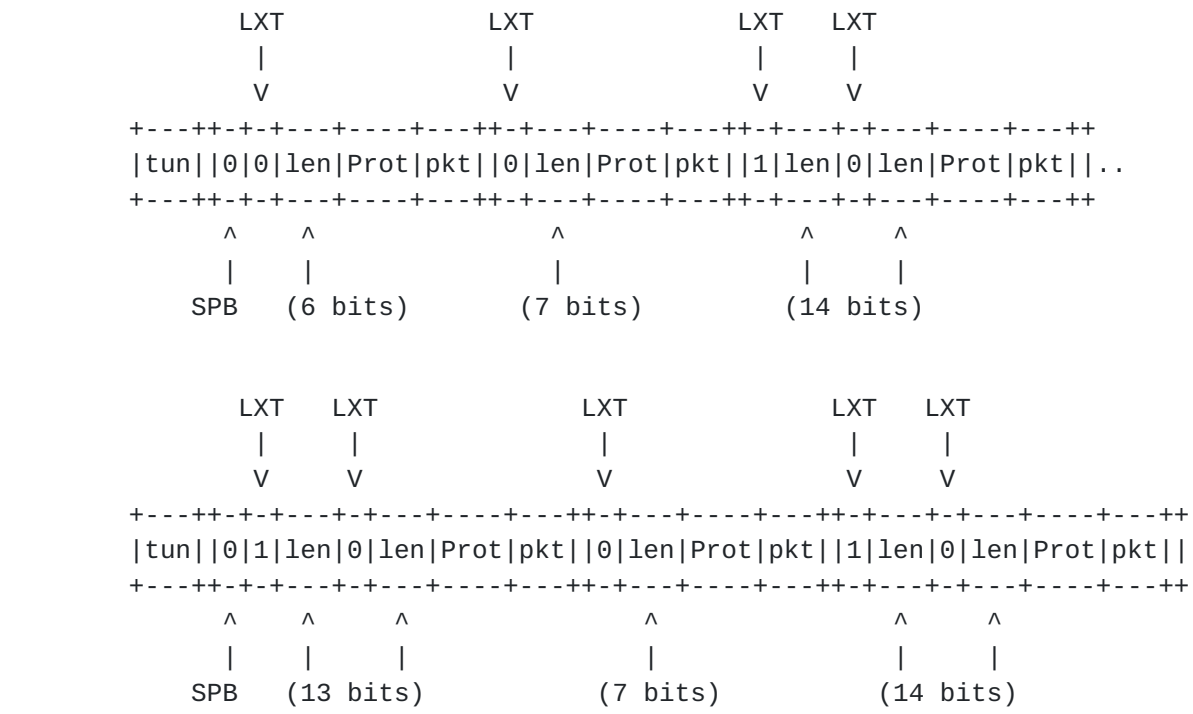


Figure 19

7. IANA Considerations

A protocol number for Simplemux should be requested to IANA.

As a provisional solution for IP networks, the ingress and the egress optimizers may agree on a UDP port, and use IP/UDP as the multiplexing protocol.

8. Security Considerations

Simplemux protocol has been developed in such a way that packet aggregation and security can be simultaneously applied to the same traffic flows, i.e. a single security header could protect a number of packets belonging to different flows.

As a consequence, the overall efficiency could be improved, as the number of security headers could be reduced from N to 1 (being N the number of multiplexed packets).

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC1570] Simpson, W., Ed., "PPP LCP Extensions", RFC 1570, DOI 10.17487/RFC1570, January 1994, <https://www.rfc-editor.org/info/rfc1570>.

[RFC3153] Pazhyannur, R., Ali, I., and C. Fox, "PPP Multiplexing", RFC 3153, DOI 10.17487/RFC3153, August 2001, <https://www.rfc-editor.org/info/rfc3153>.

[RFC4170] Thompson, B., Koren, T., and D. Wing, "Tunneling Multiplexed Compressed RTP (TCRTP)", BCP 110, RFC 4170, DOI 10.17487/RFC4170, November 2005, <https://www.rfc-editor.org/info/rfc4170>.

9.2. Informative References

[RFC1692] Cameron, P., Crocker, D., Cohen, D., and J. Postel, "Transport Multiplexing Protocol (TMux)", RFC 1692, DOI 10.17487/RFC1692, August 1994, <https://www.rfc-editor.org/info/rfc1692>.

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

[chab] Chabarek, J., Sommers, J., Barford, P., Estan, C., Tsiang, D., and S. Wright, "Power Awareness in Network Design and Routing", INFOCOM 2008. The 27th Conference on Computer Communications. IEEE pp.457,465, 2008.

[Saldana] Saldana, J., Forcen, I., Fernandez-Navajas, J., and J. Ruiz-Mas, "Improving Network Efficiency with Simplemux", IEEE CIT 2015, International Conference on Computer and Information Technology pp. 446-453, 26-28 October 2015, Liverpool, UK., 2015.

Author's Address

Jose Saldana
CIRCE Technology Center
Email: jmsaldana@fcirce.es