

loops  
Internet-Draft  
Intended status: Standards Track  
Expires: May 7, 2020

C. Bormann  
Universitaet Bremen TZI  
November 04, 2019

Embedding LOOPS in Geneve  
draft-bormann-loops-geneve-binding-00

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local in-network loss recovery. It can be used with tunneling protocols to efficiently recover lost packets on a single segment of an end-to-end path instead of leaving recovery to the end-to-end protocol, traversing the entire path.

[I-D.welzl-loops-gen-info] defines the information to be carried between LOOPS ingress and egress nodes in a generic way, giving a guideline on defining the common elements to embed LOOPS functions in various tunnel protocols. The present document specifies how to embed LOOPS in a specific overlay tunnel protocol, Geneve [I-D.ietf-nvo3-geneve].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Geneve LOOPS Frame Format . . . . .	3
3.1. Flags and Flag Based Data . . . . .	5
4. Security Considerations . . . . .	6
5. IANA Considerations . . . . .	6
5.1. Geneve Option Class . . . . .	6
5.2. LOOPS Geneve Type Numbers . . . . .	6
6. References . . . . .	7
6.1. Normative References . . . . .	7
6.2. Informative References . . . . .	7
Acknowledgements . . . . .	8
Author's Address . . . . .	8

## 1. Introduction

LOOPS (Local Optimizations on Path Segments) aims to provide local in-network loss recovery. The LOOPS problems and opportunities draft [I-D.li-tsvwg-loops-problem-opportunities] illustrates some typical scenarios where LOOPS are applicable. One way to use LOOPS is to map it onto a tunnel protocol. The path segment on which LOOPS is applied then is a tunnel, which can be an existing one or created on purpose.

LOOPS allows the packet loss recovery to be performed over specific segments instead of end-to-end, enabling faster and more reliable data delivery. [I-D.welzl-loops-gen-info] defines the information to be carried between LOOPS ingress and egress nodes in a generic way, giving a guideline on defining the common elements to embed LOOPS functions in various tunnel protocols.

Geneve [I-D.ietf-nvo3-geneve] is an encapsulation protocol that can be used to create overlay tunnels. It defines an extensible TLV structure to carry so-called "tunnel options". The present document employs this flexibility, specifying how to embed LOOPS in Geneve. This specification covers the format and Geneve-specific procedures only: the actual LOOPS function and procedures are defined in [I-D.welzl-loops-gen-info].

LOOPS has two modes of loss recovery, retransmission and forward error correction (FEC). The current version of the present document covers retransmission only.

## 2. Terminology

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.

This document makes use of the terminology defined in [I-D.welzl-loops-gen-info].

## 3. Geneve LOOPS Frame Format

Figure 1 shows the format of the Geneve Header and a single Geneve Option, as defined in [I-D.ietf-nvo3-geneve]. Geneve LOOPS defines a new Option class called LOOPS to carry forward and backward information.

Geneve Header and Option:

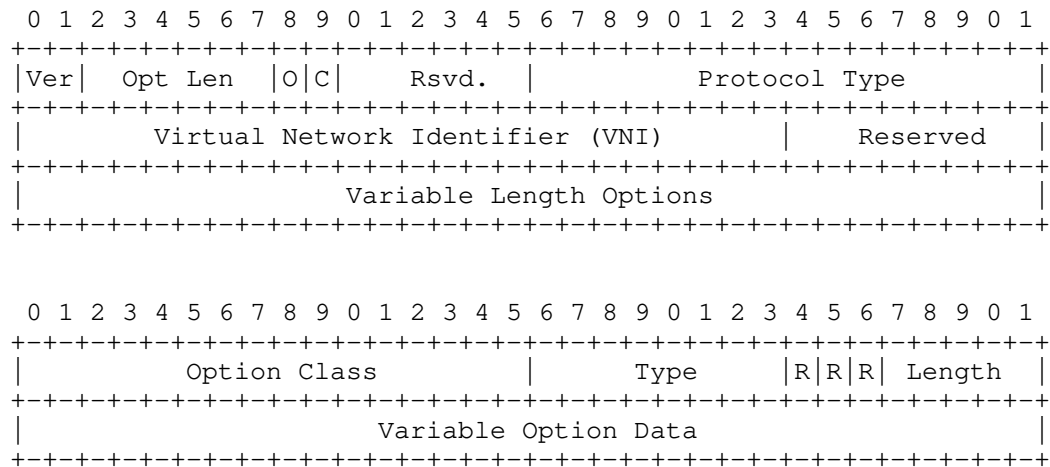


Figure 1: Geneve Header and Option Format

In the Geneve Option structure, a Geneve LOOPS option uses the following values:

- o Option Class: TBD1 for LOOPS (see Section 5).

- o Type: Further to the substructure already defined in Geneve, which uses bit 0 (the most significant bit) to indicate a critical option, LOOPS defines bit 1 as the M bit to indicate the LOOPS retransmission mode, see Figure 2. [[\_5: So what do we use the remaining six bits for? Could we move over some of the flags?]] The present document only addresses messages with M=0.

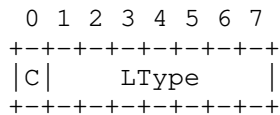


Figure 2: Type Field Format in Geneve LOOPS Option

- o C: Critical bit as defined in [I-D.ietf-nvo3-geneve].
- o LType: LOOPS Mode.
  - \* 0: Retransmission mode. In this mode, the LOOPS option format and operations follow this document.
  - \* 64: FEC mode
  - \* Further mode values can be assigned in an IANA registry (see Section 5.2).
- o Length: Length of Variable Option Data field, expressed in four byte multiples excluding the option header, ranging from 0 to 31. As the option header is another four bytes, the total length of the option in bytes is therefore  $4 * (1 + \text{Length})$ , yielding a maximum total length of 128 bytes.
- o Variable option data: consists of two parts, Flags and Flag Based Data, as shown in Figure 3.
  - \* Flags: 16 bits, as described in next subsection. Some of the flags indicate the presence of additional data in the field of Flag Based Data.
  - \* Flag Based Data: This field consists of one or multiple optional data blocks whose presence is indicated by the corresponding flag bits. Any remaining bytes needed to reach a multiple of four bytes are filled with zeroes.

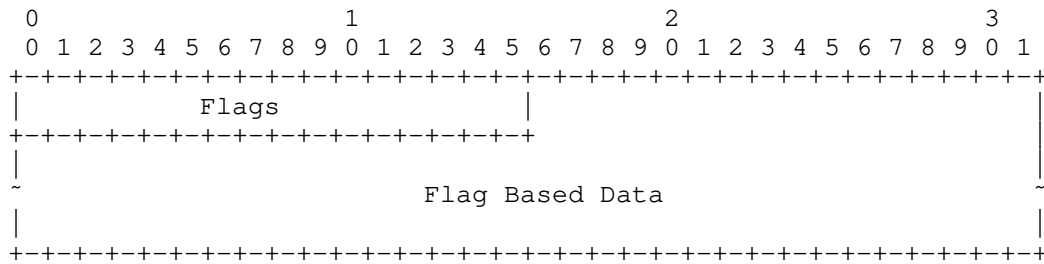


Figure 3: Variable Option Data Format in Geneve LOOPS Option

### 3.1. Flags and Flag Based Data

Flags for LOOPS Tunnel Options are defined in Figure 4. Some flags cause additional data blocks to occur in the Flags Based Data field. Those additional data blocks are placed in the order of the flags causing them.



Figure 4: Flags in Variable Option Data in Geneve LOOPS Option

A number of the flag bits are used on their own and do not cause carrying additional data:

- o I: Initial Packet Sequence Number (PSN) flag; may be set by the LOOPS ingress to notify the egress about using a new initial PSN.
- o R: Initial PSN Received flag; echo of I flag provided by the LOOPS egress.
- o D: ACK Desired flag; set by the LOOPS ingress if it wants the egress to generate an acknowledgement immediately upon receiving a particular packet.

These flag bits cause the addition of a single 32-bit number each:

- o S: PSN flag; indicates a PSN data block is carried in the Flag Based Data field. It must be set when a packet payload is present. It must not be set if the packet is a pure LOOPS ACK packet, i.e. when no payload is included in the packet.
- o T: Timestamp flag. When set, it indicates a Timestamp data block is carried in the Flag Based Data field. [[\_9: Might want to have "timestamp" and "echo" fields of less or more than 4 bytes.]]

- o E: Echoed Timestamp flag. When set, it indicates an Echoed Timestamp data block is carried in the Flag Based Data field.
- o A: ACK number flag. When set, it indicates the presence of a Block 1 ACK information block.
- o R: Reception time flag: May only be set if A is set. Indicates that an absolute reception time is given (Format TBD).

Finally, a single flag bit is defined that causes the addition of a variable-length block (therefore this flag is put as the least significant bit of Flags):

- o B: Block 2 flag. When set, it indicates the presence a Block 2 ACK information block, with the following format: TBD [[\_6: copy over the structure we have in gen-info.]]

Acknowledgement information can be sent as a pure ACK packet without payload or piggybacked in a data packet.

#### 4. Security Considerations

The security considerations of [I-D.welzl-loops-gen-info] and [I-D.ietf-nvo3-geneve] apply.

#### 5. IANA Considerations

##### 5.1. Geneve Option Class

IANA is requested to assign a new option class for LOOPS from the "Geneve Option Class" registry.

Option Class	Description
TBD1	LOOPS (Local Optimizations on Path Segments) [RFCthis]

##### 5.2. LOOPS Geneve Type Numbers

IANA is requested to create a registry for type numbers ("LType") as used in the TBD1 option class for LOOPS from the "Geneve Option Class" registry, with the following three columns:

Type Number: Integer between 0 and 127

Description: Short Description

Reference: Reference to Specification

The initial contents of the registry is:

Type Number	Description	Reference
0	Retransmission mode	[RFCthis]
64	FEC mode	[RFCthis]

(Registry policy TBD, probably Specification Required.)

## 6. References

### 6.1. Normative References

- [I-D.welzl-loops-gen-info]  
Welzl, M. and C. Bormann, "LOOPS Generic Information Set", draft-welzl-loops-gen-info-01 (work in progress), September 2019.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-14 (work in progress), September 2019.
- [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>>.

### 6.2. Informative References

- [I-D.li-tsvwg-loops-problem-opportunities]  
Yizhou, L., Zhou, X., Boucadair, M., and J. Wang, "LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement", draft-li-tsvwg-loops-problem-opportunities-03 (work in progress), July 2019.

#### Acknowledgements

Sami Boutros provided some advice on the use of Geneve in this protocol binding.

#### Author's Address

Carsten Bormann  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63921  
Email: [cabo@tzi.org](mailto:cabo@tzi.org)



loops  
Internet-Draft  
Intended status: Standards Track  
Expires: 14 December 2020

C. Bormann  
Universitaet Bremen TZI  
12 June 2020

Embedding LOOPS in Geneve  
draft-bormann-loops-geneve-binding-01

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local in-network loss recovery. It can be used with tunneling protocols to efficiently recover lost packets on a single segment of an end-to-end path instead of leaving recovery to the end-to-end protocol, traversing the entire path.

[I-D.welzl-loops-gen-info] defines the information to be carried between LOOPS ingress and egress nodes in a generic way, giving a guideline on defining the common elements to embed LOOPS functions in various tunnel protocols. The present document specifies how to embed LOOPS in the overlay tunnel protocol chosen for the initial LOOPS specification, Geneve [I-D.ietf-nvo3-geneve].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 December 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Geneve LOOPS Frame Format . . . . .	3
3.1. Flags and Flag Based Data . . . . .	5
4. Security Considerations . . . . .	6
5. IANA Considerations . . . . .	6
5.1. Geneve Option Class . . . . .	6
5.2. LOOPS Geneve Type Numbers . . . . .	7
6. References . . . . .	7
6.1. Normative References . . . . .	7
6.2. Informative References . . . . .	8
Acknowledgements . . . . .	8
Author's Address . . . . .	8

## 1. Introduction

LOOPS (Local Optimizations on Path Segments) aims to provide local in-network loss recovery. The LOOPS problems and opportunities draft [I-D.li-tsvwg-loops-problem-opportunities] illustrates some typical scenarios where LOOPS are applicable. One way to use LOOPS is to map it onto a tunnel protocol. The path segment on which LOOPS is applied then is a tunnel, which can be an existing one or created on purpose.

LOOPS allows the packet loss recovery to be performed over specific segments instead of end-to-end, enabling faster and more reliable data delivery. [I-D.welzl-loops-gen-info] defines the information to be carried between LOOPS ingress and egress nodes in a generic way, giving a guideline on defining the common elements to embed LOOPS functions in various tunnel protocols.

Geneve [I-D.ietf-nvo3-geneve] is an encapsulation protocol that can be used to create overlay tunnels. It defines an extensible TLV structure to carry so-called "tunnel options". The present document employs this flexibility, specifying how to embed LOOPS in Geneve. This specification covers the format and Geneve-specific procedures only: the actual LOOPS function and procedures are defined in [I-D.welzl-loops-gen-info].

LOOPS has two modes of loss recovery, retransmission and forward error correction (FEC). The current version of the present document covers retransmission only.

## 2. Terminology

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.

This document makes use of the terminology defined in [I-D.welzl-loops-gen-info].

## 3. Geneve LOOPS Frame Format

Figure 1 shows the format of the Geneve Header and a single Geneve Option, as defined in [I-D.ietf-nvo3-geneve]. Geneve LOOPS defines a new Option class called LOOPS to carry LOOPS forward and backward information.

Geneve Header and Option:

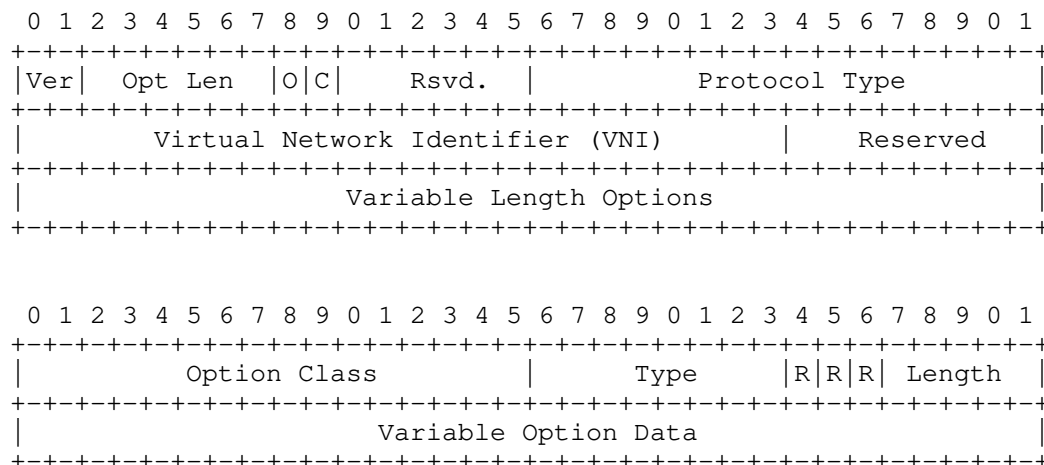


Figure 1: Geneve Header and Option Format

In the Geneve Option structure, a Geneve LOOPS option uses the following values:

- \* Option Class: TBD1 for LOOPS (see Section 5).
- \* Type: Based on the substructure already defined in Geneve, which uses bit 0 (the most significant bit) to indicate a critical option (see Figure 2), LOOPS defines two type numbers: 0 for LOOPS retransmission mode, and 64 for FEC mode. The present document only addresses messages with LType=0.

TBD: Additional type numbers could be defined, possibly obviating the need for some of the flags in the current option structure.

```

 0 1 2 3 4 5 6 7
+-+--+--+--+--+--+
|C|      LType      |
+-+--+--+--+--+--+

```

Figure 2: Type Field Format in Geneve LOOPS Option

- \* C: Critical bit as defined in [I-D.ietf-nvo3-geneve].
- \* LType: LOOPS Mode.
  - 0: Retransmission mode. In this mode, the LOOPS option format and operations follow this document.
  - 64: FEC mode
  - Further mode values can be assigned in an IANA registry (see Section 5.2).
- \* Length: Length of Variable Option Data field, expressed in four byte multiples excluding the option header, ranging from 0 to 31. As the option header is another four bytes, the total length of the option in bytes is therefore  $4 * (1 + \text{Length})$ , yielding a maximum total length of 128 bytes.
- \* Variable option data: consists of two parts, Flags and Flag Based Data, as shown in Figure 3.
  - Flags: 16 bits, as described in next subsection. Some of the flags indicate the presence of additional data in the field of Flag Based Data.

- **Flag Based Data:** This field consists of one or multiple optional data blocks whose presence is indicated by the corresponding flag bits. Any remaining bytes needed to reach a multiple of four bytes are filled with zeroes.

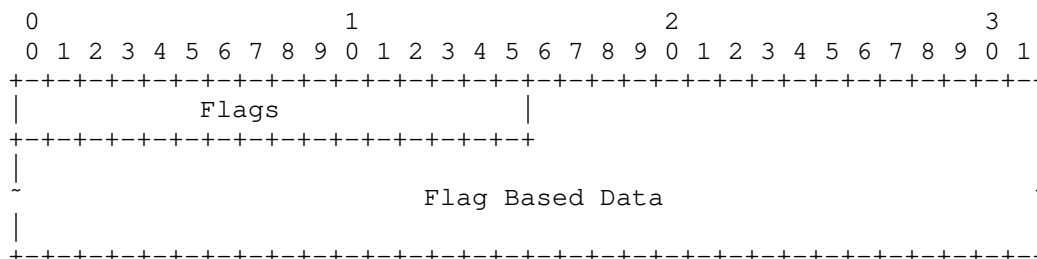


Figure 3: Variable Option Data Format in Geneve LOOPS Option

### 3.1. Flags and Flag Based Data

Flags for LOOPS Tunnel Options are defined in Figure 4. Some flags cause additional data blocks to occur in the Flags Based Data field. Those additional data blocks are placed in the order of the flags causing them.

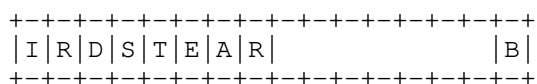


Figure 4: Flags in Variable Option Data in Geneve LOOPS Option

A number of the flag bits are used on their own and do not cause carrying additional data:

- \* **I:** Initial Packet Sequence Number (PSN) flag; may be set by the LOOPS ingress to notify the egress about using a new initial PSN.
- \* **R:** Initial PSN Received flag; echo of I flag provided by the LOOPS egress.
- \* **D:** ACK Desired flag; set by the LOOPS ingress if it wants the egress to generate an acknowledgement immediately upon receiving a particular packet.

These flag bits cause the addition of a single 32-bit number each:

- \* S: PSN flag; indicates a PSN data block is carried in the Flag Based Data field. It must be set when a packet payload is present. It must not be set if the packet is a pure LOOPS ACK packet, i.e. when no payload is included in the packet.
- \* T: Timestamp flag. When set, it indicates a Timestamp data block is carried in the Flag Based Data field.  
// TBD: Might want to have "timestamp" and "echo" fields of less or  
// more than 4 bytes.
- \* E: Echoed Timestamp flag. When set, it indicates an Echoed Timestamp data block is carried in the Flag Based Data field.
- \* A: ACK number flag. When set, it indicates the presence of a Block 1 ACK information block.
- \* R: Reception time flag: May only be set if A is set. Indicates that an absolute reception time is given (Format TBD).

Finally, a single flag bit is defined that causes the addition of a variable-length block (therefore this flag is put as the least significant bit of Flags):

- \* B: Block 2 flag. When set, it indicates the presence a Block 2 ACK information block, with the following format: TBD  
// copy over the structure we have in gen-info.

Acknowledgement information can be sent as a pure ACK packet without payload or piggybacked in a data packet.

#### 4. Security Considerations

The security considerations of [I-D.welzl-loops-gen-info] and [I-D.ietf-nvo3-geneve] apply.

#### 5. IANA Considerations

##### 5.1. Geneve Option Class

IANA is requested to assign a new option class for LOOPS from the "Geneve Option Class" registry.

Option Class	Description
TBD1	LOOPS (Local Optimizations on Path Segments) [RFCthis]

Table 1

## 5.2. LOOPS Geneve Type Numbers

IANA is requested to create a registry for type numbers ("LType") as used in the TBD1 option class for LOOPS from the "Geneve Option Class" registry, with the following three columns:

Type Number: Integer between 0 and 127

Description: Short Description

Reference: Reference to Specification

The initial contents of the registry is:

Type Number	Description	Reference
0	Retransmission mode	[RFCthis]
64	FEC mode	[RFCthis]

Table 2

(Registry policy TBD, probably Specification Required.)

## 6. References

### 6.1. Normative References

[I-D.welzl-loops-gen-info]

Welzl, M. and C. Bormann, "LOOPS Generic Information Set", Work in Progress, Internet-Draft, draft-welzl-loops-gen-info-03, 9 March 2020, <<http://www.ietf.org/internet-drafts/draft-welzl-loops-gen-info-03.txt>>.

[I-D.ietf-nvo3-geneve]

Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", Work in Progress,

Internet-Draft, draft-ietf-nvo3-geneve-16, 7 March 2020,  
<<http://www.ietf.org/internet-drafts/draft-ietf-nvo3-geneve-16.txt>>.

- [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>>.

## 6.2. Informative References

- [I-D.li-tsvwg-loops-problem-opportunities]  
Yizhou, L., Zhou, X., Boucadair, M., and J. Wang, "LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement", Work in Progress, Internet-Draft, draft-li-tsvwg-loops-problem-opportunities-04, 6 January 2020, <<http://www.ietf.org/internet-drafts/draft-li-tsvwg-loops-problem-opportunities-04.txt>>.

## Acknowledgements

Sami Boutros provided some advice on the use of Geneve in this protocol binding.

## Author's Address

Carsten Bormann  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63921  
Email: [cabo@tzi.org](mailto:cabo@tzi.org)



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 16, 2020

F. Maino, Ed.  
Cisco Systems  
L. Kreeger, Ed.  
Arrcus  
U. Elzur, Ed.  
Intel  
October 14, 2019

Generic Protocol Extension for VXLAN  
draft-ietf-nvo3-vxlan-gpe-08

Abstract

This draft describes extending Virtual eXtensible Local Area Network (VXLAN), via changes to the VXLAN header, with three new capabilities: support for multi-protocol encapsulation, operations, administration and management (OAM) signaling and explicit versioning.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 16, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction	2
2. VXLAN Without Protocol Extension	3
3. Generic Protocol Extension for VXLAN (VXLAN GPE)	4
3.1. VXLAN GPE Header	4
3.2. Multi Protocol Support	5
3.3. Replicated BUM Traffic	7
3.4. OAM Support	7
3.5. Version Bits	8
4. Outer Encapsulations	8
4.1. Inner VLAN Tag Handling	12
4.2. Fragmentation Considerations	12
5. Backward Compatibility	12
5.1. VXLAN VTEP to VXLAN GPE VTEP	12
5.2. VXLAN GPE VTEP to VXLAN VTEP	12
5.3. VXLAN GPE UDP Ports	13
5.4. VXLAN GPE and Encapsulated IP Header Fields	13
6. VXLAN GPE Examples	13
7. Security Considerations	14
8. Contributors	14
9. Acknowledgments	15
10. IANA Considerations	16
10.1. UDP Port	16
10.2. VXLAN GPE Next Protocol	16
10.3. VXLAN GPE Flag and Reserved Bits	16
11. References	17
11.1. Normative References	17
11.2. Informative References	18
Authors' Addresses	18

## 1. Introduction

Virtual eXtensible Local Area Network VXLAN [RFC7348] defines an encapsulation format that encapsulates Ethernet frames in an outer UDP/IP transport. As data centers evolve, the need to carry other protocols encapsulated in an IP packet is required, as well as the need to provide increased visibility and diagnostic capabilities

within the overlay. The VXLAN header does not specify the protocol being encapsulated and therefore is currently limited to encapsulating only Ethernet frame payload, nor does it provide the ability to define OAM protocols. In addition, [RFC6335] requires that new transports not use transport layer port numbers to identify tunnel payload, rather it encourages encapsulations to use their own identifiers for this purpose. VXLAN GPE is intended to extend the existing VXLAN protocol to provide protocol typing, OAM, and versioning capabilities.

The Version and OAM bits are introduced in Section 3, and the choice of location for these fields is driven by minimizing the impact on existing deployed hardware.

In order to facilitate deployments of VXLAN GPE with hardware currently deployed to support VXLAN, changes from legacy VXLAN have been kept to a minimum. Section 5 provides a detailed discussion about how VXLAN GPE addresses the requirement for backward compatibility with VXLAN.

## 2. VXLAN Without Protocol Extension

VXLAN provides a method of creating multi-tenant overlay networks by encapsulating packets in IP/UDP along with a header containing a network identifier which is used to isolate tenant traffic in each overlay network from each other. This allows the overlay networks to run over an existing IP network.

Through this encapsulation, VXLAN creates stateless tunnels between VXLAN Tunnel End Points (VTEPs) which are responsible for adding/removing the IP/UDP/VXLAN headers and providing tenant traffic isolation based on the VXLAN Network Identifier (VNI). Tenant systems are unaware that their networking service is being provided by an overlay.

When encapsulating packets, a VTEP must know the IP address of the proper remote VTEP at the far end of the tunnel that can deliver the inner packet to the Tenant System corresponding to the inner destination address. In the case of tenant multicast or broadcast, the outer IP address may be an IP multicast group address, or the VTEP may replicate the packet and send it to all known VTEPs. If multicast is used in the underlay network to send encapsulated packets to remote VTEPs, Any Source Multicast is used and each VTEP serving a particular VNI must perform a (\*, G) join to the same group IP address.

Inner to outer address mapping can be determined in two ways. One is source based learning in the data plane, and the other is distribution via a control plane.

Source based learning requires a receiving VTEP to create an inner to outer address mapping by gleaning the information from the received packets by correlating the inner source address to the outer source IP address. When a mapping does not exist, a VTEP forwards the packets to all remote VTEPs participating in the VNI by using IP multicast in the IP underlay network. Each VTEP must be configured with the IP multicast address to use for each VNI. How this occurs is out of scope.

The control plane used to distribute inner to outer mappings is also out of scope. It could use a centralized authority or be distributed, or use a hybrid.

The VXLAN Network Identifier (VNI) provides scoping for the addresses in the header of the encapsulated PDU. If the encapsulated packet is an Ethernet frame, this means the Ethernet MAC addresses are only unique within a given VNI and may overlap with MAC addresses within a different VNI. If the encapsulated packet is an IP packet, this means the IP addresses are only unique within that VNI.

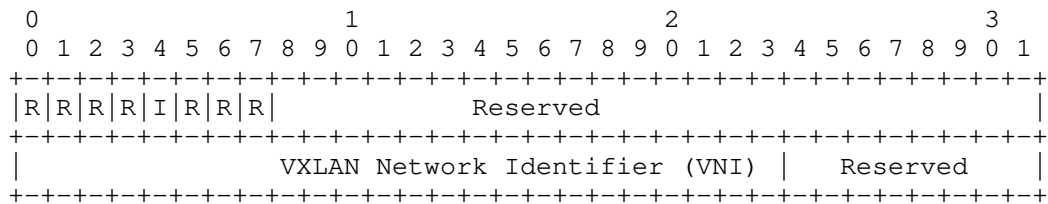


Figure 1: VXLAN Header

### 3. Generic Protocol Extension for VXLAN (VXLAN GPE)

#### 3.1. VXLAN GPE Header

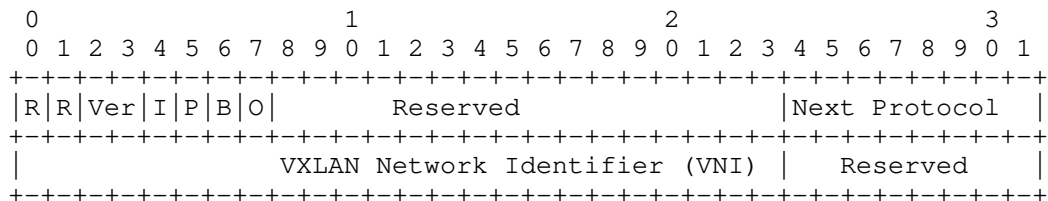


Figure 2: VXLAN GPE Header

Flags (8 bits): The first 8 bits of the header are the flag field. The bits designated "R" above are reserved flags. These MUST be set to zero on transmission and ignored on receipt.

Version (Ver): Indicates VXLAN GPE protocol version. The initial version is 0. If a receiver does not support the version indicated it MUST drop the packet.

Instance Bit (I bit): The I bit MUST be set to indicate a valid VNI.

Next Protocol Bit (P bit): The P bit is set to indicate that the Next Protocol field is present.

BUM Traffic Bit (B bit): The B bit is set to indicate that this is ingress-replicated BUM Traffic (ie, Broadcast, Unknown unicast, or Multicast).

OAM Flag Bit (O bit): The O bit is set to indicate that the packet is an OAM packet.

Next Protocol: This 8 bit field indicates the protocol header immediately following the VXLAN GPE header.

VNI: This 24 bit field identifies the VXLAN overlay network the inner packet belongs to. Inner packets belonging to different VNIs cannot communicate with each other (unless explicitly allowed by policy).

Reserved: Reserved fields MUST be set to zero on transmission and ignored on receipt.

### 3.2. Multi Protocol Support

This draft defines the following two changes to the VXLAN header in order to support multi-protocol encapsulation:

P Bit: Flag bit 5 is defined as the Next Protocol bit. The P bit MUST be set to 1 to indicate the presence of the 8 bit next protocol field.

When UDP dest port=4790, P = 0 the "Next Protocol" field must be set to zero and the payload MUST be ETHERNET(L2) as defined by [RFC7348].

Flag bit 5 was chosen as the P bit because this flag bit is currently reserved in VXLAN.

**Next Protocol Field:** The lower 8 bits of the first word are used to carry a next protocol. This next protocol field contains the protocol of the encapsulated payload packet. A new protocol registry will be requested from IANA, see section 10.2.

This draft defines the following Next Protocol values:

0x1 : IPv4

0x2 : IPv6

0x3 : Ethernet

0x4 : Network Service Header [RFC8300]

0x5 : Multiprotocol Label Switching [RFC3031]. See [I-D.ietf-idr-tunnel-encaps] for more details.

0x6: Unassigned.

0x7: virtual Broadband Network Gateway (vBNG)  
[I-D.huang-nvo3-vxlan-gpe-extension-for-vbng].

0x8 to 0x7F: Unassigned.

0x80: Group-Based Policy (GBP) [I-D.lemon-vxlan-lisp-gpe-gbp]

0x81: In-situ OAM Data (iOAM) [I-D.brockners-ippm-ioam-vxlan-gpe]

0x82 to 0xFF: Unassigned.

Next protocol values from 0x80 to 0xFF are assigned to protocols encoded as generic "shim" headers. These protocols, when present, MUST be encapsulated before protocols identified by next protocol values from 0x0 to 0x7F.

Implementations that are not aware of a given shim header MUST ignore the header and proceed to parse the next protocol. Shim protocols MUST have the first 32 bits defined as:

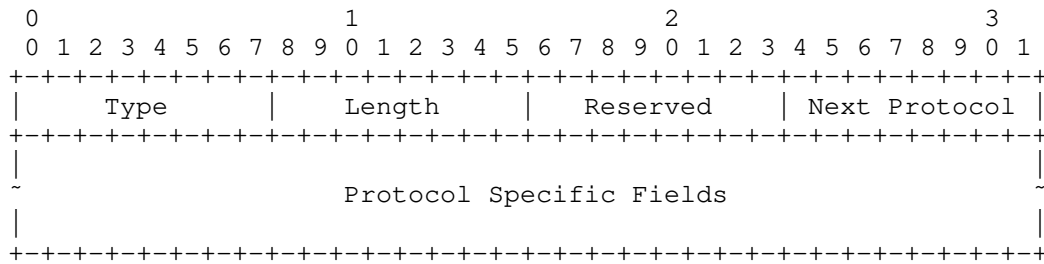


Figure 3: Shim Header

Where:

**Type:** This field MAY be used to identify different messages of this protocol.

**Length:** The length, in 4-octet units, of this protocol message not including the first 4 octets.

**Reserved:** The use of this field is reserved to the protocol defined in this message.

**Next Protocol Field:** This next protocol field contains the protocol of the encapsulated payload. The protocol registry will be requested from IANA as per section 10.2.

### 3.3. Replicated BUM Traffic

Flag bit 6 is defined as the B bit. When the B bit is set to 1, the packet is marked as an ingress-replicated BUM Traffic (i.e. Broadcast, Unknown unicast, or Multicast) to help egress VTEP to differentiate between known and unknown unicast. The details of using the B bit are out of scope for this document, but please see [RFC8365] for an example in the EVPN context. As with the P-bit, bit 6 is currently a reserved flag in VXLAN.

### 3.4. OAM Support

Flag bit 7 is defined as the O bit. When the O bit is set to 1, the packet is an OAM packet and OAM processing MUST occur. Other header fields including Next Protocol MUST adhere to the definitions in Section 3. The OAM protocol details are out of scope for this document. As with the P-bit, bit 7 is currently a reserved flag in VXLAN.

### 3.5. Version Bits

VXLAN GPE bits 2 and 3 are defined as version bits. These bits are reserved in VXLAN. The version field is used to ensure backward compatibility going forward with future VXLAN GPE updates.

The initial version for VXLAN GPE is 0.

## 4. Outer Encapsulations

In addition to the VXLAN GPE header, the packet is further encapsulated in UDP and IP. Data centers based on Ethernet, will then send this IP packet over Ethernet.

Outer UDP Header:

Destination UDP Port: IANA has assigned the value 4790 for the VXLAN GPE UDP port. This well-known destination port is used when sending VXLAN GPE encapsulated packets.

Source UDP Port: The source UDP port is used as entropy for devices forwarding encapsulated packets across the underlay (ECMP for IP routers, or load splitting for link aggregation by bridges). Tenant traffic flows should all use the same source UDP port to lower the chances of packet reordering by the underlay for a given flow. It is recommended for VTEPs to generate this port number using a hash of the inner packet headers. Implementations MAY use the entire 16 bit source UDP port for entropy.

UDP Checksum: Source VTEPs MAY either calculate a valid checksum, or if this is not possible, set the checksum to zero. When calculating a checksum, it MUST be calculated across the entire packet (outer IP header, UDP header, VXLAN GPE header and payload packet). All receiving VTEPs must accept a checksum value of zero. If the receiving VTEP is capable of validating the checksum, it MAY validate a non-zero checksum and MUST discard the packet if the checksum is determined to be invalid.

Outer IP Header:

This is the header used by the underlay network to deliver packets between VTEPs. The destination IP address can be a unicast or a multicast IP address. The source IP address must be the source VTEP IP address which can be used to return tenant packets to the tenant system source address within the inner packet header.

When the outer IP header is IPv4, VTEPs MUST set the DF bit.



## Outer Ethernet Header:

Most data centers networks are built on Ethernet. Assuming the outer IP packet is being sent across Ethernet, there will be an Ethernet header used to deliver the IP packet to the next hop, which could be the destination VTEP or be a router used to forward the IP packet towards the destination VTEP. If VLANs are in use within the data center, then this Ethernet header would also contain a VLAN tag.

The following figures show the entire stack of protocol headers that would be seen on an Ethernet link carrying encapsulated packets from a VTEP across the underlay network for both IPv4 and IPv6 based underlay networks.

```

      0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Outer Ethernet Header:
+++++
|                               Outer Destination MAC Address                               |
+++++
| Outer Destination MAC Address | Outer Source MAC Address |
+++++
|                               Outer Source MAC Address                               |
+++++
| Opt Ethertype = C-Tag 802.1Q | Outer VLAN Tag |
+++++
| Ethertype = 0x0800 |
+++++

```

## Outer IPv4 Header:

```

+++++
|Version| IHL |Type of Service| Total Length |
+++++
| Identification |Flags| Fragment Offset |
+++++
| Time to Live |Protocol=17(UDP)| Header Checksum |
+++++
|                               Outer Source IPv4 Address                               |
+++++
|                               Outer Destination IPv4 Address                           |
+++++

```

## Outer UDP Header:

```

+++++
|                               Source Port | Dest Port = 4790 |
+++++

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|               UDP Length               |               UDP Checksum               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### VXLAN GPE Header:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|R|R|Ver|I|P|R|O|               Reserved               |Next Protocol|
+-----+-----+-----+-----+-----+-----+-----+-----+
|               VXLAN Network Identifier (VNI) |       Reserved       |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### Payload:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|               Depends on VXLAN GPE Next Protocol field above.               |
|               Note that if the payload is Ethernet, then the original         |
|               Ethernet Frame's FCS is not included.                           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### Frame Check Sequence:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|               New FCS (Frame Check Sequence) for Outer Ethernet Frame               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 4: Outer Headers for VXLAN GPE over IPv4

```

0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

```

#### Outer Ethernet Header:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|               Outer Destination MAC Address               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Outer Destination MAC Address | Outer Source MAC Address |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Outer Source MAC Address               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Ethertype = C-Tag 802.1Q |       Outer VLAN Tag       |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Ethertype = 0x86DD |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### Outer IPv6 Header:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|Version| Traffic Class |               Flow Label               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

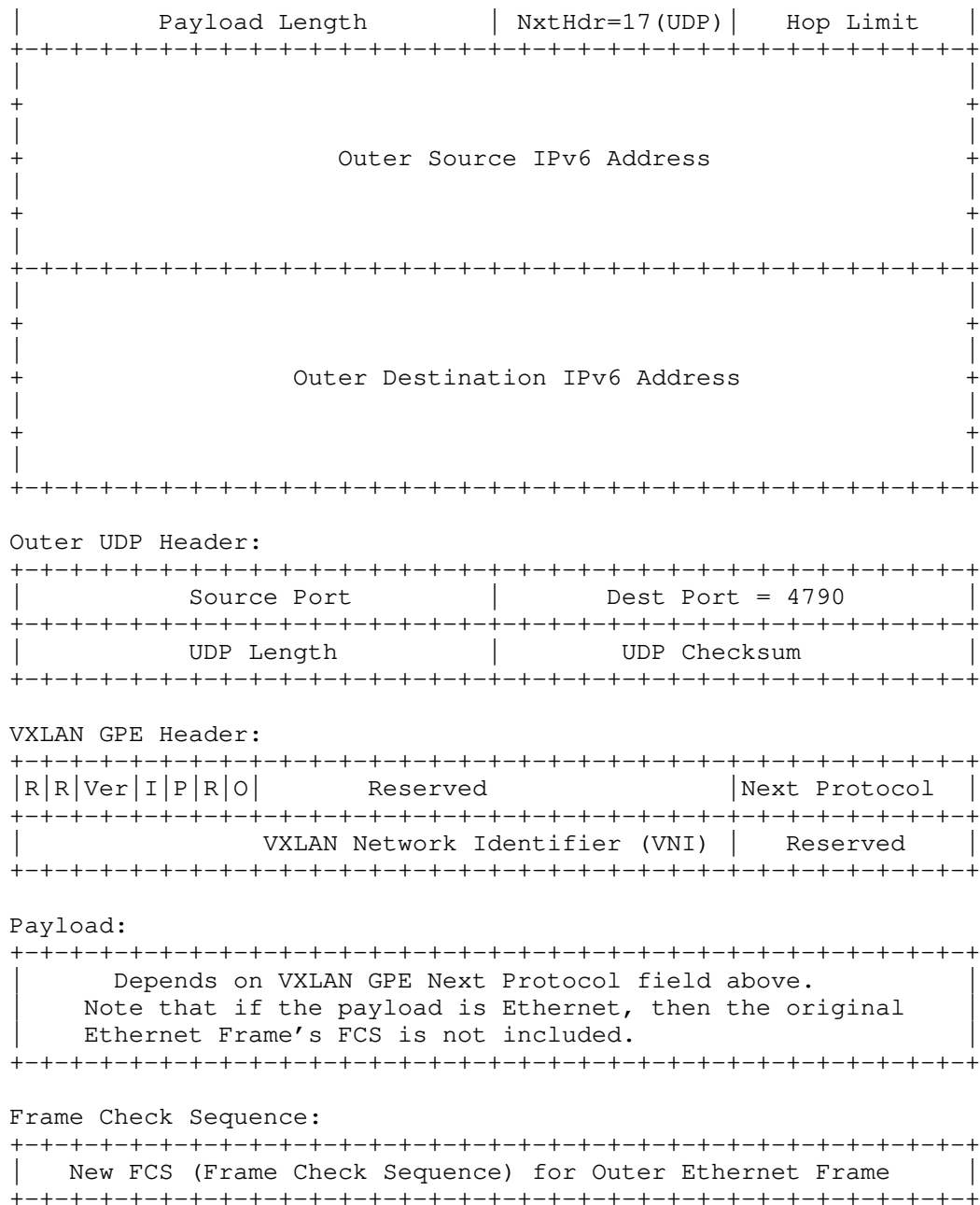


Figure 5: Outer Headers for VXLAN GPE over IPv6

#### 4.1. Inner VLAN Tag Handling

If the inner packet (as indicated by the VXLAN GPE Next Protocol field) is an Ethernet frame, it is recommended that it does not contain a VLAN tag. In the most common scenarios, the tenant VLAN tag is translated into a VXLAN Network Identifier. In these scenarios, VTEPs should never send an inner Ethernet frame with a VLAN tag, and a VTEP performing decapsulation should discard any inner frames received with a VLAN tag. However, if the VTEPs are specifically configured to support it for a specific VXLAN Network Identifier, a VTEP may support transparent transport of the inner VLAN tag between all tenant systems on that VNI. The VTEP never looks at the value of the inner VLAN tag, but simply passes it across the underlay.

#### 4.2. Fragmentation Considerations

VTEPs MUST never fragment an encapsulated VXLAN GPE packet, and when the outer IP header is IPv4, VTEPs MUST set the DF bit in the outer IPv4 header. It is recommended that the underlay network be configured to carry an MTU at least large enough to accommodate the added encapsulation headers. It is recommended that VTEPs perform Path MTU discovery [RFC1191] [RFC1981] to determine if the underlay network can carry the encapsulated payload packet.

### 5. Backward Compatibility

#### 5.1. VXLAN VTEP to VXLAN GPE VTEP

A VXLAN VTEP conforms to VXLAN frame format and uses UDP destination port 4789 when sending traffic to VXLAN GPE VTEP. As per VXLAN, reserved bits 5 and 7, VXLAN GPE P and O-bits respectively must be set to zero. The remaining reserved bits must be zero, including the VXLAN GPE version field, bits 2 and 3. The encapsulated payload MUST be Ethernet.

#### 5.2. VXLAN GPE VTEP to VXLAN VTEP

A VXLAN GPE VTEP MUST NOT encapsulate non-Ethernet frames to a VXLAN VTEP. When encapsulating Ethernet frames to a VXLAN VTEP, the VXLAN GPE VTEP MUST conform to VXLAN frame format and hence will set the P bit to 0, the Next Protocol to 0 and use UDP destination port 4789. A VXLAN GPE VTEP MUST also set O = 0 and Ver = 0 when encapsulating Ethernet frames to VXLAN VTEP. The receiving VXLAN VTEP will treat this packet as a VXLAN packet.

A method for determining the capabilities of a VXLAN VTEP (GPE or non-GPE) is out of the scope of this draft.

### 5.3. VXLAN GPE UDP Ports

VXLAN GPE uses a IANA assigned UDP destination port, 4790, when sending traffic to VXLAN GPE VTEPs.

### 5.4. VXLAN GPE and Encapsulated IP Header Fields

When encapsulating and decapsulating IPv4 and IPv6 packets, certain fields, such as IPv4 Time to Live (TTL) from the inner IP header need to be considered. VXLAN GPE IP encapsulation and decapsulation utilizes the techniques described in [RFC6830], section 5.3.

## 6. VXLAN GPE Examples

This section provides three examples of protocols encapsulated using the Generic Protocol Extension for VXLAN described in this document.

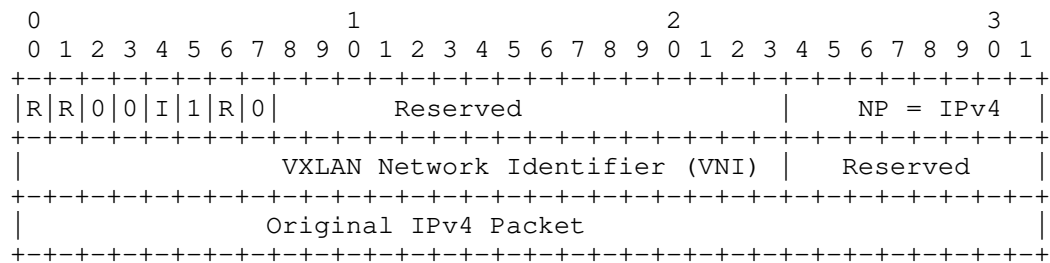


Figure 6: IPv4 and VXLAN GPE

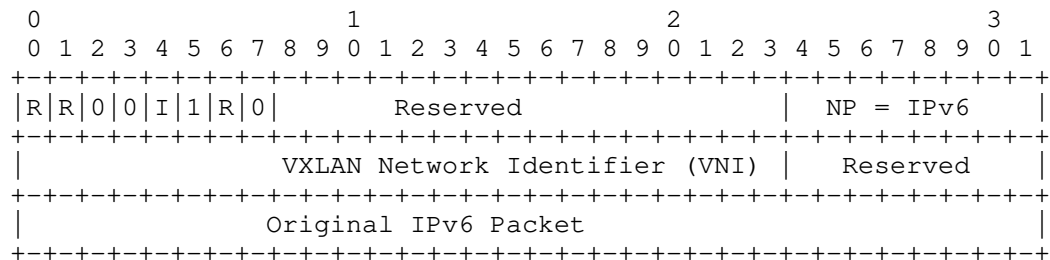


Figure 7: IPv6 and VXLAN GPE

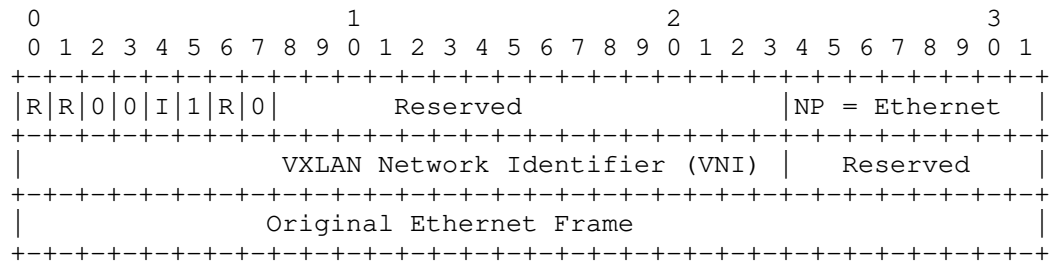


Figure 8: Ethernet and VXLAN GPE

## 7. Security Considerations

VXLAN's security is focused on issues around L2 encapsulation into L3. With VXLAN GPE, issues such as spoofing, flooding, and traffic redirection are dependent on the particular protocol payload encapsulated.

## 8. Contributors

Paul Quinn  
Cisco Systems  
paulq@cisco.com

Rajeev Manur  
Broadcom  
rmanur@broadcom.com

Michael Smith  
Cisco Systems  
michsmi@cisco.com

Darrel Lewis  
Cisco Systems  
darlewis@cisco.com

Puneet Agarwal  
Innovium, Inc  
puneet@acm.org

Lucy Yong  
Huawei USA  
lucy.yong@huawei.com

Xiaohu Xu  
Huawei Technologies  
xuxiaohu@huawei.com

Pankaj Garg  
Microsoft  
pankajg@microsoft.com

David Melman  
Marvell  
davidme@marvell.com

Jennifer Lemon  
Broadcom Limited  
jennifer.lemon@broadcom.com

## 9. Acknowledgments

A special thank you goes to Dino Farinacci for his guidance and detailed review.

## 10. IANA Considerations

### 10.1. UDP Port

UDP 4790 port has been assigned by IANA for VXLAN GPE.

### 10.2. VXLAN GPE Next Protocol

IANA is requested to set up a registry of "Next Protocol". These are 8-bit values. Next Protocol values in the table below are defined in this draft. New values are assigned via Standards Action [RFC5226].

Next Protocol	Description	Reference
0x0	Reserved	This Document
0x1	IPv4	This Document
0x2	IPv6	This Document
0x3	Ethernet	This Document
0x4	NSH	This Document
0x5	MPLS	This Document
0x6	Unassigned	
0x7	vBNG	This Document
0x8..0x7F	Unassigned	
0x80	GBP	This Document
0x81	iOAM	This Document
0x82..0xFF	Unassigned	

### 10.3. VXLAN GPE Flag and Reserved Bits

There are ten flag bits at the beginning of the VXLAN GPE header, followed by 16 reserved bits and an 8-bit reserved field at the end of the header. New bits are assigned via Standards Action [RFC5226].

Bits 0-1 - Reserve6

Bits 2-3 - Version

Bit 4 - Instance ID (I bit)

Bit 5 - Next Protocol (P bit)

Bit 6 - Reserved

Bit 7 - OAM (O bit)

Bit 8-23 - Reserved



Bits 24-31 in the 2nd Word -- Reserved

Reserved bits/fields MUST be set to 0 by the sender and ignored by the receiver.

## 11. References

### 11.1. Normative References

- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <<https://www.rfc-editor.org/info/rfc1981>>.
- [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>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.

- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.

## 11.2. Informative References

- [I-D.brockners-ippm-ioam-vxlan-gpe]  
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., and M. Spiegel, "VXLAN-GPE Encapsulation for In-situ OAM Data", draft-brockners-ippm-ioam-vxlan-gpe-02 (work in progress), July 2019.
- [I-D.huang-nvo3-vxlan-gpe-extension-for-vbng]  
Huang, L., Hu, S., Wang, Z., and T. Ao, "VXLAN GPE Extension for Packets Exchange Between Control and User Plane of vBNG", draft-huang-nvo3-vxlan-gpe-extension-for-vbng-01 (work in progress), October 2017.
- [I-D.ietf-idr-tunnel-encaps]  
Patel, K., Velde, G., and S. Ramachandra, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-14 (work in progress), September 2019.
- [I-D.lemon-vxlan-lisp-gpe-gbp]  
Lemon, J., Maino, F., Smith, M., and A. Isaac, "Group Policy Encoding with VXLAN-GPE and LISP-GPE", draft-lemon-vxlan-lisp-gpe-gbp-02 (work in progress), April 2019.

## Authors' Addresses

Fabio Maino (Editor)  
Cisco Systems

Email: [fmaino@cisco.com](mailto:fmaino@cisco.com)

Larry Kreeger (editor)  
Arrcus

Email: [lkreeger@gmail.com](mailto:lkreeger@gmail.com)

Uri Elzur (editor)  
Intel

Email: [uri.elzur@intel.com](mailto:uri.elzur@intel.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 26, 2022

F. Maino, Ed.  
Cisco Systems  
L. Kreeger, Ed.  
Arrcus  
U. Elzur, Ed.  
Intel  
September 22, 2021

Generic Protocol Extension for VXLAN (VXLAN-GPE)  
draft-ietf-nvo3-vxlan-gpe-12

## Abstract

This document describes extending Virtual eXtensible Local Area Network (VXLAN), via changes to the VXLAN header, with four new capabilities: support for multi-protocol encapsulation, support for operations, administration and maintenance (OAM) signaling, support for ingress-replicated BUM Traffic (i.e. Broadcast, Unknown unicast, or Multicast), and explicit versioning. New protocol capabilities can be introduced via shim headers.

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 26, 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. VXLAN Without Protocol Extension . . . . .	3
3. Generic Protocol Extension for VXLAN (VXLAN-GPE) . . . . .	4
3.1. VXLAN-GPE Header . . . . .	4
3.2. Multi Protocol Support . . . . .	5
3.3. Replicated BUM Traffic . . . . .	7
3.4. OAM Support . . . . .	7
3.5. Version Bits . . . . .	8
4. Outer Encapsulations . . . . .	8
4.1. Inner VLAN Tag Handling . . . . .	11
4.2. Fragmentation Considerations . . . . .	12
5. Implementation and Deployment Considerations . . . . .	12
5.1. Applicability Statement . . . . .	12
5.2. Congestion Control Functionality . . . . .	13
5.3. UDP Checksum . . . . .	13
5.3.1. UDP Zero Checksum Handling with IPv6 . . . . .	13
6. Backward Compatibility . . . . .	15
6.1. VXLAN VTEP to VXLAN-GPE VTEP . . . . .	15
6.2. VXLAN-GPE VTEP to VXLAN VTEP . . . . .	15
6.3. VXLAN-GPE UDP Ports . . . . .	15
6.4. VXLAN-GPE and Encapsulated IP Header Fields . . . . .	15
7. VXLAN-GPE Examples . . . . .	16
8. Security Considerations . . . . .	17
9. Contributors . . . . .	17
10. Acknowledgments . . . . .	18
11. IANA Considerations . . . . .	19
11.1. UDP Port . . . . .	19
11.2. VXLAN-GPE Next Protocol . . . . .	19
11.3. VXLAN-GPE Flag and Reserved Bits . . . . .	19
12. References . . . . .	20
12.1. Normative References . . . . .	20

12.2. Informative References . . . . .	22
Authors' Addresses . . . . .	22

## 1. Introduction

Virtual eXtensible Local Area Network VXLAN [RFC7348] defines an encapsulation format that encapsulates Ethernet frames in an outer UDP/IP transport. As data centers evolve, the need to carry other protocols encapsulated in an IP packet is required, as well as the need to provide increased visibility and diagnostic capabilities within the overlay. The VXLAN header does not specify the protocol being encapsulated and therefore is currently limited to encapsulating only Ethernet frame payload, nor does it provide the ability to define OAM protocols. In addition, [RFC6335] requires that new transports not use transport layer port numbers to identify tunnel payload, rather it encourages encapsulations to use their own identifiers for this purpose. VXLAN-GPE is intended to extend the existing VXLAN protocol to provide protocol typing, OAM, and versioning capabilities.

The Version and OAM bits are introduced in Section 3, and the choice of location for these fields is driven by minimizing the impact on existing deployed hardware.

In order to facilitate deployments of VXLAN-GPE with hardware currently deployed to support VXLAN, changes from legacy VXLAN have been kept to a minimum. Section 6 provides a detailed discussion about how VXLAN-GPE addresses the requirement for backward compatibility with VXLAN.

The capabilities of the VXLAN-GPE protocol can be extended by defining next protocol "shim" headers that are used to implement new data plane functions. For example, Group-Based Policy (GBP) or In-situ Operations, Administration, and Maintenance (IOAM) metadata functionalities can be added as specified in [I-D.lemon-vxlan-lisp-gpe-gbp] and [I-D.brockners-ippm-ioam-vxlan-gpe].

## 2. VXLAN Without Protocol Extension

VXLAN provides a method of creating multi-tenant overlay networks by encapsulating packets in IP/UDP along with a header containing a network identifier which is used to isolate tenant traffic in each overlay network from each other. This allows the overlay networks to run over an existing IP network.

Through this encapsulation, VXLAN creates stateless tunnels between VXLAN Tunnel End Points (VTEPs) which are responsible for adding/

removing the IP/UDP/VXLAN headers and providing tenant traffic isolation based on the VXLAN Network Identifier (VNI). Tenant systems are unaware that their networking service is being provided by an overlay.

When encapsulating packets, a VTEP must know the IP address of the proper remote VTEP at the far end of the tunnel that can deliver the inner packet to the Tenant System corresponding to the inner destination address. The control plane used to distribute inner to outer mappings is out of the scope of this document.

The VXLAN Network Identifier (VNI) provides scoping for the addresses in the header of the encapsulated PDU. If the encapsulated packet is an Ethernet frame, this means the Ethernet MAC addresses are only unique within a given VNI and may overlap with MAC addresses within a different VNI. If the encapsulated packet is an IP packet, this means the IP addresses are only unique within that VNI.

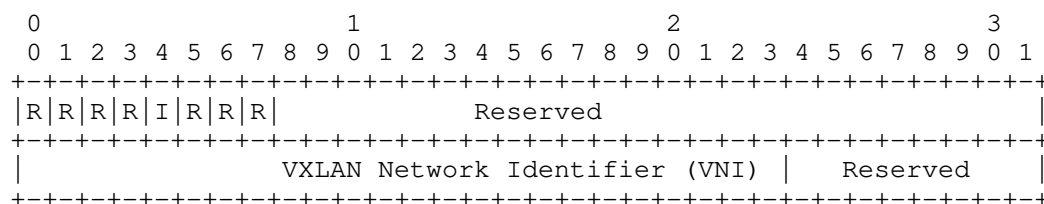


Figure 1: VXLAN Header

### 3. Generic Protocol Extension for VXLAN (VXLAN-GPE)

#### 3.1. VXLAN-GPE Header

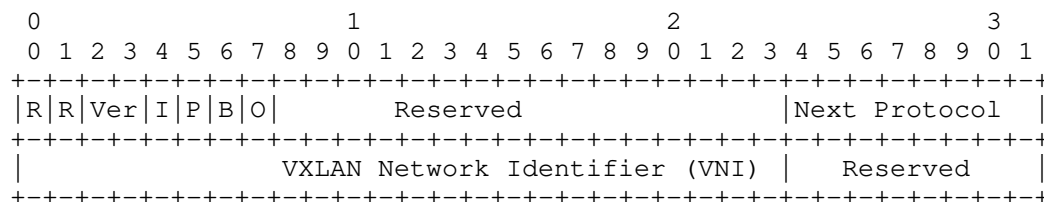


Figure 2: VXLAN-GPE Header

Flags (8 bits): The first 8 bits of the header are the flag field.

The bits designated "R" above are reserved flags. These MUST be set to zero on transmission and ignored on receipt.

Version (Ver): Indicates VXLAN-GPE protocol version. The initial version is 0. If a receiver does not support the version indicated it MUST drop the packet.

Instance Bit (I bit): The I bit MUST be set to indicate a valid VNI.

Next Protocol Bit (P bit): The P bit is set to indicate that the Next Protocol field is present.

BUM Traffic Bit (B bit): The B bit is set to indicate that this is ingress-replicated BUM Traffic (ie, Broadcast, Unknown unicast, or Multicast).

OAM Flag Bit (O bit): The O bit is set to indicate that the packet is an OAM packet.

Next Protocol: This 8 bit field indicates the protocol header immediately following the VXLAN-GPE header.

VNI: This 24 bit field identifies the VXLAN overlay network the inner packet belongs to. Inner packets belonging to different VNIs cannot communicate with each other (unless explicitly allowed by policy).

Reserved: Reserved fields MUST be set to zero on transmission and ignored on receipt.

### 3.2. Multi Protocol Support

This draft defines the following two changes to the VXLAN header in order to support multi-protocol encapsulation:

P Bit: Flag bit 5 is defined as the Next Protocol bit. The P bit MUST be set to 1 to indicate the presence of the 8 bit next protocol field.

When UDP dest port=4790, P = 0 the "Next Protocol" field must be set to zero and the payload MUST be ETHERNET(L2) as defined by [RFC7348].

Flag bit 5 was chosen as the P bit because this flag bit is currently reserved in VXLAN.

Next Protocol Field: The lower 8 bits of the first word are used to carry a next protocol. This next protocol field contains the protocol of the encapsulated payload packet. A new protocol registry will be requested from IANA, see section 10.2.



This draft defines the following Next Protocol values:

0x00 : Reserved

0x01 : IPv4

0x02 : IPv6

0x03 : Ethernet

0x04 : Network Service Header [RFC8300]

0x05 to 0x7D: Unassigned

0x7E, 0x7F: Experimentation and testing

0x80 to 0xFD: Unassigned (shim headers)

0xFE, 0xFF: Experimentation and testing (shim headers)

Next protocol values 0x7E, 0x7F and 0xFE, 0xFF are assigned for experimentation and testing as per [RFC3692].

Next protocol values from 0x80 to 0xFD are assigned to protocols encoded as generic "shim" headers. All shim protocols MUST use the header structure in Figure 3, which includes a Type, a Length, and a Next Protocol field. When shim headers are used with other protocols identified by next protocol values from 0x0 to 0x7F, all the shim headers MUST come first.

Shim headers can be used to incrementally deploy new GPE features without updating the implementation of each transit node between two tunnel endpoints, and without punting the packet with shim headers of unknown type to the 'slow' path. Transit nodes that are not aware of a given shim header type MUST ignore that shim header and proceed to parse the next protocol.

VTEP implementations can keep the processing of known shim headers in the 'fast' path (typically an ASIC), while punting the processing of the remaining new GPE features to the 'slow' path.

Shim protocols MUST have the first 32 bits defined as:

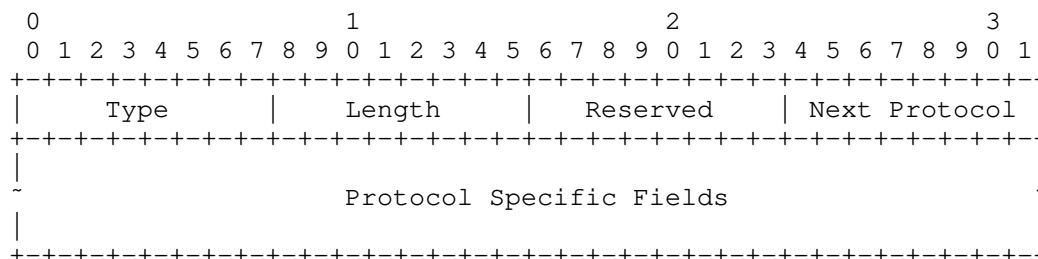


Figure 3: Shim Header

Where:

**Type:** This field MAY be used to identify different messages of this protocol.

**Length:** The length, in in 4-octet units, of this protocol message not including the first 4 octets.

**Reserved:** The use of this field is reserved to the protocol defined in this message.

**Next Protocol Field:** This next protocol field contains the protocol of the encapsulated payload. The protocol registry will be requested from IANA as per section 10.2.

### 3.3. Replicated BUM Traffic

Flag bit 6 is defined as the B bit. When the B bit is set to 1, the packet is marked as an ingress-replicated BUM Traffic (i.e. Broadcast, Unknown unicast, or Multicast) to help egress VTEP to differentiate between known and unknown unicast. The details of using the B bit are out of scope for this document, but please see [RFC8365] for an example in the EVPN context. As with the P-bit, bit 6 is currently a reserved flag in VXLAN.

### 3.4. OAM Support

Flag bit 7 is defined as the O bit. When the O bit is set to 1, the packet is an OAM packet and OAM processing MUST occur. Other header fields including Next Protocol MUST adhere to the definitions in Section 3. The OAM protocol details are out of scope for this document. As with the P-bit, bit 7 is currently a reserved flag in VXLAN.

### 3.5. Version Bits

VXLAN-GPE bits 2 and 3 are defined as version bits. These bits are reserved in VXLAN. The version field is used to ensure backward compatibility going forward with future VXLAN-GPE updates.

The initial version for VXLAN-GPE is 0.

## 4. Outer Encapsulations

In addition to the VXLAN-GPE header, the packet is further encapsulated in UDP and IP. Data centers based on Ethernet, will then send this IP packet over Ethernet.

Outer UDP Header:

Destination UDP Port: IANA has assigned the value 4790 for the VXLAN-GPE UDP port. This well-known destination port is used when sending VXLAN-GPE encapsulated packets.

Source UDP Port: The source UDP port is used as entropy for devices forwarding encapsulated packets across the underlay (ECMP for IP routers, or load splitting for link aggregation by bridges). Tenant traffic flows should all use the same source UDP port to lower the chances of packet reordering by the underlay for a given flow. It is recommended for VTEPs to generate this port number using a hash of the inner packet headers. Implementations MAY use the entire 16 bit source UDP port for entropy.

UDP Checksum: see Section 5.3 for considerations related to UDP Checksum processing.

Outer IP Header:

This is the header used by the underlay network to deliver packets between VTEPs. The destination IP address can be a unicast or a multicast IP address. The source IP address must be the source VTEP IP address which can be used to return tenant packets to the tenant system source address within the inner packet header.

When the outer IP header is IPv4, VTEPs MUST set the DF bit.

Outer Ethernet Header:

Most data centers networks are built on Ethernet. Assuming the outer IP packet is being sent across Ethernet, there will be an Ethernet header used to deliver the IP packet to the next hop, which could be the destination VTEP or be a router used to forward the IP packet

towards the destination VTEP. If VLANs are in use within the data center, then this Ethernet header would also contain a VLAN tag.

The following figures show the entire stack of protocol headers that would be seen on an Ethernet link carrying encapsulated packets from a VTEP across the underlay network for both IPv4 and IPv6 based underlay networks.

```

      0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

```

Outer Ethernet Header:

```

+-----+
|               Outer Destination MAC Address               |
+-----+
| Outer Destination MAC Address | Outer Source MAC Address |
+-----+
|               Outer Source MAC Address                   |
+-----+
| Opt Ethertype = C-Tag 802.1Q |       Outer VLAN Tag       |
+-----+
| Ethertype = 0x0800          |
+-----+

```

Outer IPv4 Header:

```

+-----+
|Version|  IHL  |Type of Service|          Total Length          |
+-----+
|      Identification      |Flags|      Fragment Offset      |
+-----+
| Time to Live |Protocol=17(UDP)|      Header Checksum      |
+-----+
|          Outer Source IPv4 Address          |
+-----+
|          Outer Destination IPv4 Address     |
+-----+

```

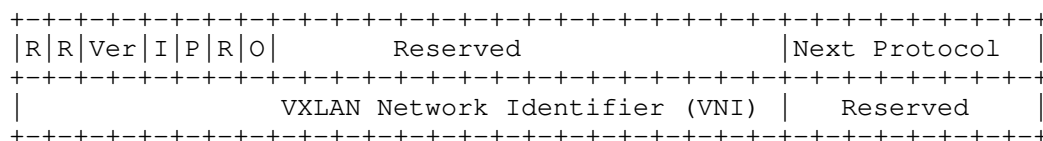
Outer UDP Header:

```

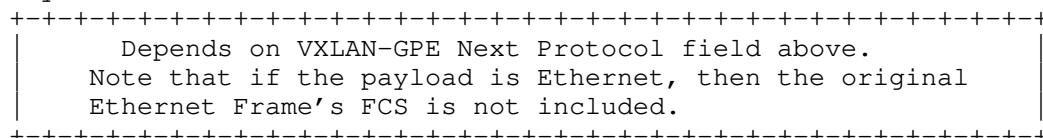
+-----+
|          Source Port          |      Dest Port = 4790      |
+-----+
|          UDP Length          |      UDP Checksum          |
+-----+

```

VXLAN-GPE Header:



Payload:



Frame Check Sequence:

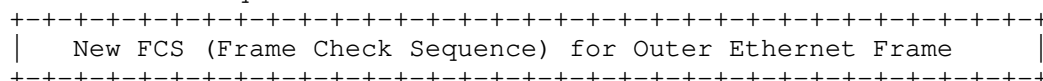
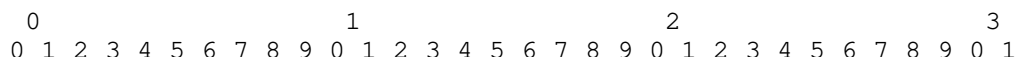
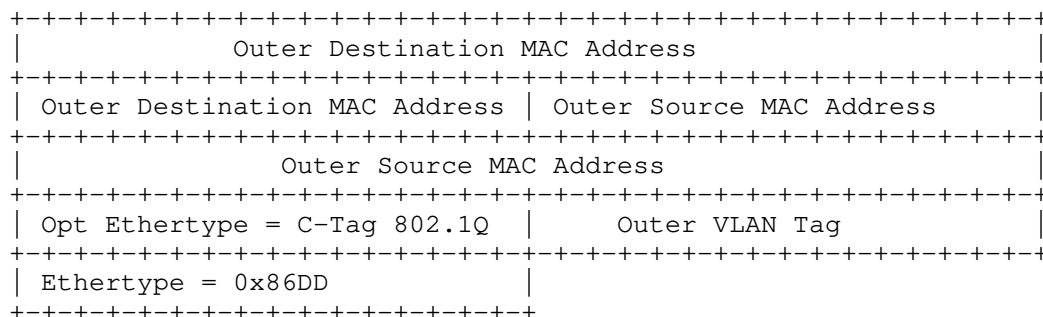


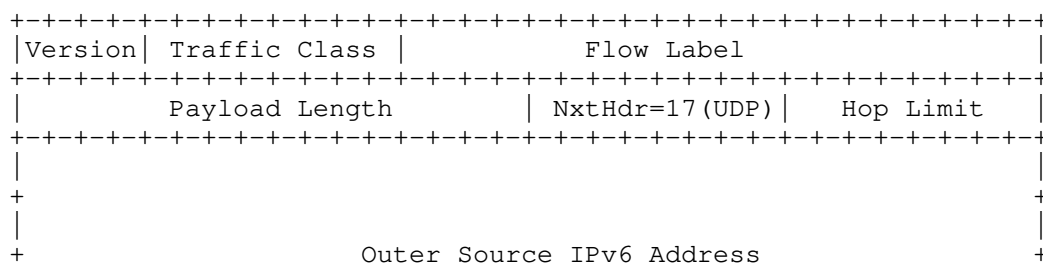
Figure 4: Outer Headers for VXLAN-GPE over IPv4



Outer Ethernet Header:



Outer IPv6 Header:



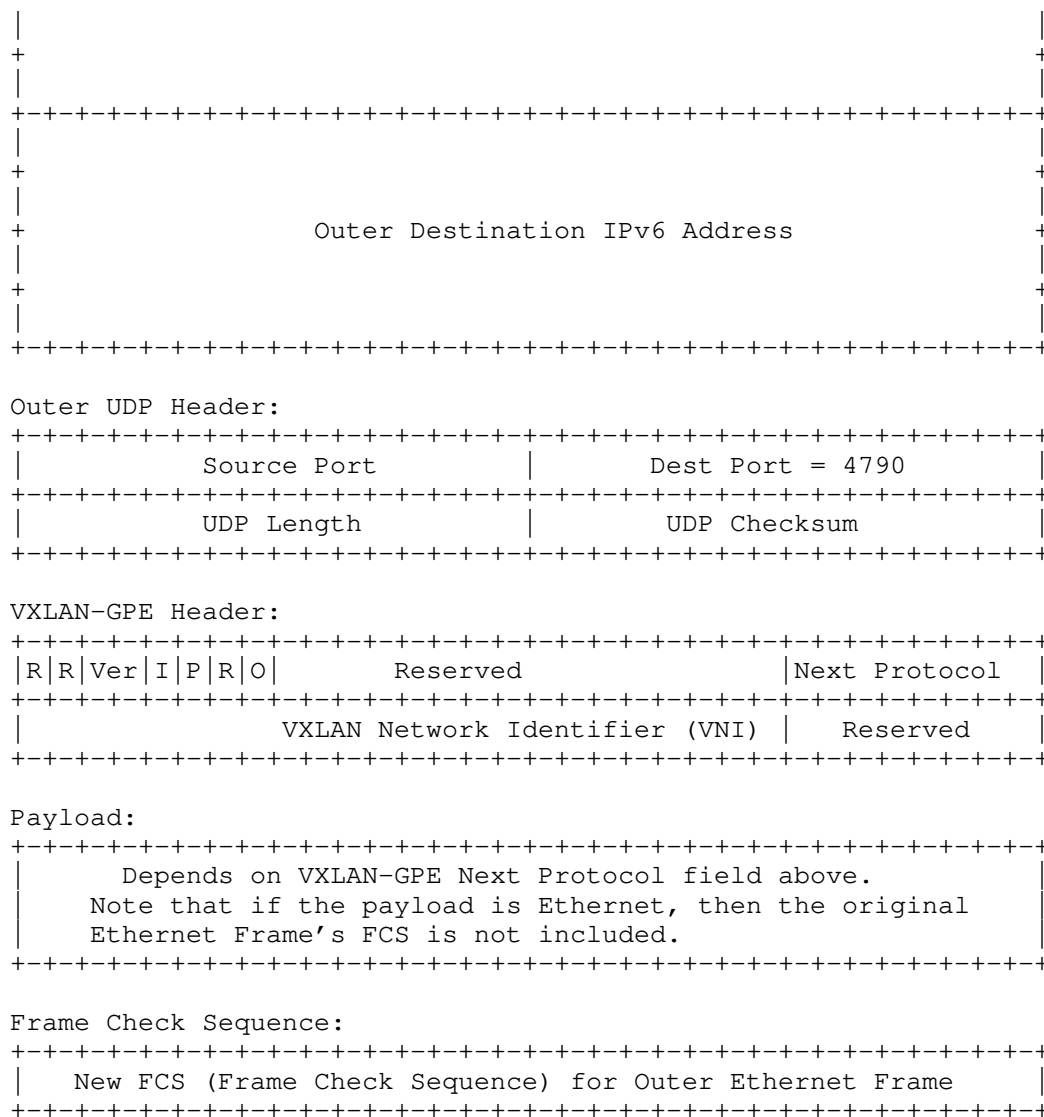


Figure 5: Outer Headers for VXLAN-GPE over IPv6

#### 4.1. Inner VLAN Tag Handling

If the inner packet (as indicated by the VXLAN-GPE Next Protocol field) is an Ethernet frame, it is recommended that it does not contain a VLAN tag. In the most common scenarios, the tenant VLAN tag is translated into a VXLAN Network Identifier. In these scenarios, VTEPs should never send an inner Ethernet frame with a

VLAN tag, and a VTEP performing decapsulation should discard any inner frames received with a VLAN tag. However, if the VTEPs are specifically configured to support it for a specific VXLAN Network Identifier, a VTEP may support transparent transport of the inner VLAN tag between all tenant systems on that VNI. The VTEP never looks at the value of the inner VLAN tag, but simply passes it across the underlay.

#### 4.2. Fragmentation Considerations

VTEPs MUST never fragment an encapsulated VXLAN-GPE packet, and when the outer IP header is IPv4, VTEPs MUST set the DF bit in the outer IPv4 header. It is recommended that the underlay network be configured to carry an MTU at least large enough to accommodate the added encapsulation headers. It is recommended that VTEPs perform Path MTU discovery [RFC1191] [RFC1981] to determine if the underlay network can carry the encapsulated payload packet.

### 5. Implementation and Deployment Considerations

#### 5.1. Applicability Statement

VXLAN-GPE conforms, as an UDP-based encapsulation protocol, to the UDP usage guidelines as specified in [RFC8085]. The applicability of these guidelines are dependent on the underlay IP network and the nature of the encapsulated payload.

[RFC8085] outlines two applicability scenarios for UDP applications, 1) general Internet and 2) controlled environment. The controlled environment means a single administrative domain or adjacent set of cooperating domains. A network in a controlled environment can be managed to operate under certain conditions whereas in general Internet this cannot be done. Hence requirements for a tunnel protocol operating under a controlled environment can be less restrictive than the requirements of general internet.

VXLAN-GPE is intended to be deployed in a data center network environment operated by a single operator or adjacent set of cooperating network operators that fits with the definition of controlled environments in [RFC8085].

For the purpose of this document, a traffic-managed controlled environment (TMCE), outlined in [RFC8086], is defined as an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion. Significant portions of text in this Section are based on [RFC8086].

It is the responsibility of the network operators to ensure that the guidelines/requirements in this section are followed as applicable to their VXLAN-GPE deployments

## 5.2. Congestion Control Functionality

VXLAN-GPE does not natively provide congestion control functionality and relies on the payload protocol traffic for congestion control. As such VXLAN-GPE MUST be used with congestion controlled traffic or within a network that is traffic managed to avoid congestion (TMCE). An operator of a traffic managed network (TMCE) may avoid congestion by careful provisioning of their networks, rate-limiting of user data traffic and traffic engineering according to path capacity.

## 5.3. UDP Checksum

In order to provide integrity of VXLAN-GPE headers and payload, for example to avoid mis-delivery of payload to different tenant systems in case of data corruption, outer UDP checksum SHOULD be used with VXLAN-GPE when transported over IPv4. The UDP checksum provides a statistical guarantee that a payload was not corrupted in transit. These integrity checks are not strong from a coding or cryptographic perspective and are not designed to detect physical-layer errors or malicious modification of the datagram (see Section 3.4 of [RFC8085]). In deployments where such a risk exists, an operator SHOULD use additional data integrity mechanisms such as offered by IPsec.

An operator MAY choose to disable UDP checksum and use zero checksum if VXLAN-GPE packet integrity is provided by other data integrity mechanisms such as IPsec or additional checksums or if one of the conditions in Section 5.3.1 a, b, c are met.

### 5.3.1. UDP Zero Checksum Handling with IPv6

By default, UDP checksum MUST be used when VXLAN-GPE is transported over IPv6. A tunnel endpoint MAY be configured for use with zero UDP checksum if additional requirements described in this section are met.

When VXLAN-GPE is used over IPv6, UDP checksum is used to protect IPv6 headers, UDP headers and VXLAN-GPE headers and payload from potential data corruption. As such by default VXLAN-GPE MUST use UDP checksum when transported over IPv6. An operator MAY choose to configure to operate with zero UDP checksum if operating in a traffic managed controlled environment as stated in Section 5.1 if one of the following conditions are met:



- a. It is known that the packet corruption is exceptionally unlikely (perhaps based on knowledge of equipment types in their underlay network) and the operator is willing to take a risk of undetected packet corruption
- b. It is judged through observational measurements (perhaps through historic or current traffic flows that use non zero checksum) that the level of packet corruption is tolerably low and where the operator is willing to take the risk of undetected corruption
- c. VXLAN-GPE payload is carrying applications that are tolerant of misdelivered or corrupted packets (perhaps through higher layer checksum validation and/or reliability through retransmission)

In addition VXLAN-GPE tunnel implementations using Zero UDP checksum MUST meet the following requirements:

1. Use of UDP checksum over IPv6 MUST be the default configuration for all VXLAN-GPE tunnels
2. If VXLAN-GPE is used with zero UDP checksum over IPv6 then such VTEP implementation MUST meet all the requirements specified in section 4 of [RFC6936] and requirements 1 as specified in section 5 of [RFC6936]
3. The VTEP that decapsulates the packet SHOULD check the source and destination IPv6 addresses are valid for the VXLAN-GPE tunnel that is configured to receive Zero UDP checksum and discard other packets for which such check fails
4. The VTEP that encapsulates the packet MAY use different IPv6 source addresses for each VXLAN-GPE tunnel that uses Zero UDP checksum mode in order to strengthen the decapsulator's check of the IPv6 source address (i.e the same IPv6 source address is not to be used with more than one IPv6 destination address, irrespective of whether that destination address is a unicast or multicast address). When this is not possible, it is RECOMMENDED to use each source address for as few VXLAN-GPE tunnels that use zero UDP checksum as is feasible
5. Measures SHOULD be taken to prevent VXLAN-GPE traffic over IPv6 with zero UDP checksum from escaping into the general Internet. Examples of such measures include employing packet filters at the gateways or edge of a VXLAN-GPE network, and/or keeping logical or physical separation of VXLAN network from networks carrying General Internet

The above requirements do not change either the requirements specified in [RFC2460] as modified by [RFC6935] or the requirements specified in [RFC6936].

The requirement to check the source IPv6 address in addition to the destination IPv6 address, plus the recommendation against reuse of source IPv6 addresses among VXLAN-GPE tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. A traffic-managed controlled environment that satisfies at least one of three conditions listed at the beginning of this section provides additional assurance.

## 6. Backward Compatibility

### 6.1. VXLAN VTEP to VXLAN-GPE VTEP

A VXLAN VTEP conforms to VXLAN frame format and uses UDP destination port 4789 when sending traffic to VXLAN-GPE VTEP. As per VXLAN, reserved bits 5 and 7, VXLAN-GPE P and O-bits respectively must be set to zero. The remaining reserved bits must be zero, including the VXLAN-GPE version field, bits 2 and 3. The encapsulated payload MUST be Ethernet.

### 6.2. VXLAN-GPE VTEP to VXLAN VTEP

A VXLAN-GPE VTEP MUST NOT encapsulate non-Ethernet frames to a VXLAN VTEP. When encapsulating Ethernet frames to a VXLAN VTEP, the VXLAN-GPE VTEP MUST conform to VXLAN frame format and hence will set the P bit to 0, the Next Protocol to 0 and use UDP destination port 4789. A VXLAN-GPE VTEP MUST also set O = 0 and Ver = 0 when encapsulating Ethernet frames to VXLAN VTEP. The receiving VXLAN VTEP will treat this packet as a VXLAN packet.

A method for determining the capabilities of a VXLAN VTEP (GPE or non-GPE) is out of the scope of this draft.

### 6.3. VXLAN-GPE UDP Ports

VXLAN-GPE uses a IANA assigned UDP destination port, 4790, when sending traffic to VXLAN-GPE VTEPs.

### 6.4. VXLAN-GPE and Encapsulated IP Header Fields

When encapsulating IP (including over Ethernet) packets [RFC2983] provides guidance for mapping DSCP between inner and outer IP headers. The Pipe model typically fits better Network virtualization. The DSCP value on the tunnel header is set based on a policy (which may be a fixed value, one based on the inner traffic

class, or some other mechanism for grouping traffic). Some aspects of the Uniform model (which treats the inner and outer DSCP value as a single field by copying on ingress and egress) may also apply, such as the ability to remark the inner header on tunnel egress based on transit marking. However, the Uniform model is not conceptually consistent with network virtualization, which seeks to provide strong isolation between encapsulated traffic and the physical network.

[RFC6040] describes the mechanism for exposing ECN capabilities on IP tunnels and propagating congestion markers to the inner packets. This behavior **MUST** be followed for IP packets encapsulated in VXLAN-GPE.

Though Uniform or Pipe models could be used for TTL (or Hop Limit in case of IPv6) handling when tunneling IP packets, Pipe model is more aligned with network virtualization. [RFC2003] provides guidance on handling TTL between inner IP header and outer IP tunnels; this model is more aligned with the Pipe model and is recommended for use with VXLAN-GPE for network virtualization applications.

When a VXLAN-GPE router performs Ethernet encapsulation, the inner 802.1Q 3-bit priority code point (PCP) field **MAY** be mapped from the encapsulated frame to the DSCP codepoint of the DS field defined in [RFC2474].

When a VXLAN-GPE router performs Ethernet encapsulation, the inner header 802.1Q VLAN Identifier (VID) **MAY** be mapped to, or used to determine the VXLAN Network Identifier (VNI) field.

## 7. VXLAN-GPE Examples

This section provides three examples of protocols encapsulated using the Generic Protocol Extension for VXLAN described in this document.

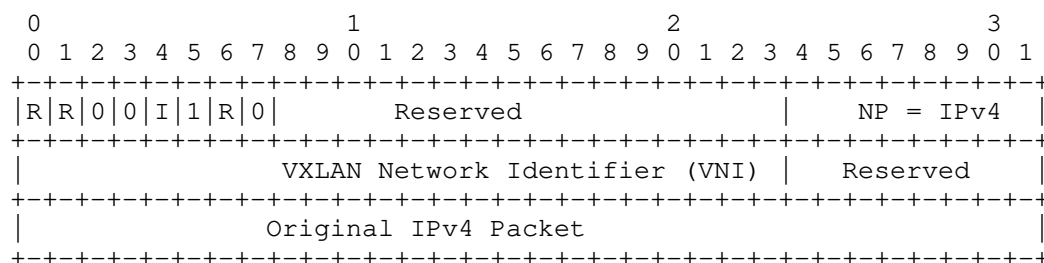


Figure 6: IPv4 and VXLAN-GPE

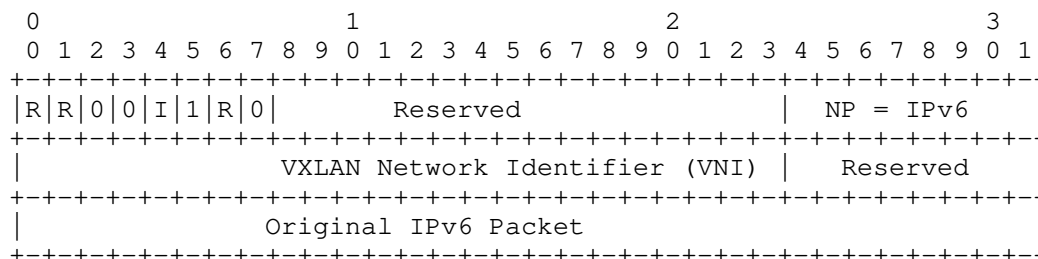


Figure 7: IPv6 and VXLAN-GPE

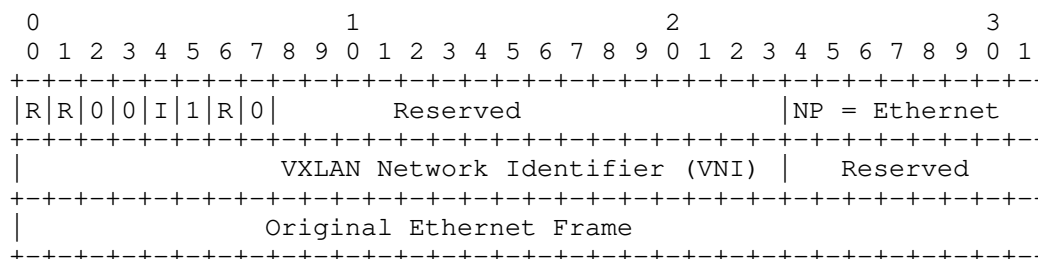


Figure 8: Ethernet and VXLAN-GPE

## 8. Security Considerations

VXLAN-GPE encapsulation does not affect security for the payload protocol. The security considerations for VXLAN applies to VXLAN-GPE, see [RFC7348].

When crossing an untrusted link, such as the public Internet, IPsec [RFC4301] may be used to provide authentication and/or encryption of the IP packets formed as part of VXLAN-GPE encapsulation.

Operators have to make an assessment based on their network environment and determine the risks that are applicable to their specific environment and use appropriate mitigation approaches as applicable.

## 9. Contributors

Paul Quinn  
Cisco Systems  
paulq@cisco.com

Rajeev Manur  
Broadcom  
rmanur@broadcom.com

Michael Smith  
Cisco Systems  
michsmit@cisco.com

Darrel Lewis  
Cisco Systems  
darlewis@cisco.com

Puneet Agarwal  
Innovium, Inc  
puneet@acm.org

Lucy Yong  
Huawei USA  
lucy.yong@huawei.com

Xiaohu Xu  
Alibaba Inc  
xiaohu.xxh@alibaba-inc.com

Pankaj Garg  
Microsoft  
pankajg@microsoft.com

David Melman  
Marvell  
davidme@marvell.com

Jennifer Lemon  
Broadcom Limited  
jennifer.lemon@broadcom.com

## 10. Acknowledgments

A special thank you goes to Dino Farinacci for his guidance and detailed review. Thanks to Tom Herbert for the suggestion to assign codepoints for experimentations and testing.

## 11. IANA Considerations

### 11.1. UDP Port

UDP 4790 port has been assigned by IANA for VXLAN-GPE.

### 11.2. VXLAN-GPE Next Protocol

IANA is requested to set up a registry of "Next Protocol". These are 8-bit values. Next Protocol values in the table below are defined in this draft. New values are assigned via Standards Action [RFC5226].

Next Protocol	Description	Reference
0x0	Reserved	This Document
0x1	IPv4	This Document
0x2	IPv6	This Document
0x3	Ethernet	This Document
0x4	NSH	This Document
0x05..0x7D 0x7E, 0x7F	Unassigned Experimentation and testing	This Document
0x80..0xFD 0x8E, 0x8F	Unassigned (shim headers) Experimentation and testing (shim headers)	This Document

### 11.3. VXLAN-GPE Flag and Reserved Bits

There are ten flag bits at the beginning of the VXLAN-GPE header, followed by 16 reserved bits and an 8-bit reserved field at the end of the header. New bits are assigned via Standards Action [RFC5226].

Bits 0-1 - Reserve6

Bits 2-3 - Version

Bit 4 - Instance ID (I bit)

Bit 5 - Next Protocol (P bit)

Bit 6 - Reserved

Bit 7 - OAM (0 bit)

Bit 8-23 - Reserved

Bits 24-31 in the 2nd Word -- Reserved

Reserved bits/fields MUST be set to 0 by the sender and ignored by the receiver.

## 12. References

### 12.1. Normative References

- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <<https://www.rfc-editor.org/info/rfc1981>>.
- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.
- [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>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.

- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <<https://www.rfc-editor.org/info/rfc3692>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <<https://www.rfc-editor.org/info/rfc6935>>.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.



- [RFC8086] Yong, L., Ed., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", RFC 8086, DOI 10.17487/RFC8086, March 2017, <<https://www.rfc-editor.org/info/rfc8086>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.

## 12.2. Informative References

- [I-D.brockners-ippm-ioam-vxlan-gpe]  
Brockners, F., Bhandari, S., Govindan, V. P., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., and M. Spiegel, "VXLAN-GPE Encapsulation for In-situ OAM Data", draft-brockners-ippm-ioam-vxlan-gpe-03 (work in progress), November 2019.
- [I-D.lemon-vxlan-lisp-gpe-gbp]  
Lemon, J., Maino, F., Smith, M., and A. Isaac, "Group Policy Encoding with VXLAN-GPE and LISP-GPE", draft-lemon-vxlan-lisp-gpe-gbp-02 (work in progress), April 2019.

## Authors' Addresses

Fabio Maino (Editor)  
Cisco Systems

Email: [fmaino@cisco.com](mailto:fmaino@cisco.com)

Larry Kreeger (editor)  
Arrcus

Email: [lkreeger@gmail.com](mailto:lkreeger@gmail.com)

Uri Elzur (editor)  
Intel

Email: [uri.elzur@intel.com](mailto:uri.elzur@intel.com)

INTERNET-DRAFT  
Intended Status: Standards Track

B. Liu, Ed.  
Huawei  
R. Chen  
ZTE  
F. Qin  
China Mobile  
R. Rahman  
Cisco

Expires: May 7, 2020

November 4, 2019

Base YANG Data Model for NVO3 Protocols  
draft-ietf-nvo3-yang-cfg-01.txt

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage Network Virtualization Overlay protocols. The model is focused on the common configuration requirement of various encapsulation options, such as VXLAN, NVGRE, GENEVE and VXLAN-GPE. Using this model as a starting point, incremental work can be done to satisfy the requirement of a specific encapsulation.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Acronyms and Terminology . . . . .	3
2.1. Acronyms . . . . .	3
2.2. Terminology . . . . .	3
3. The YANG Data Model for NVO3 . . . . .	3
3.1 Mapping to the NVO3 architecture . . . . .	4
3.2. The Configuration Parameters . . . . .	4
3.2.1. NVE as an interface . . . . .	4
3.2.2. Virtual Network Instance . . . . .	5
3.2.3. BUM Mode . . . . .	5
3.3. Statistics . . . . .	5
3.3. Model Structure . . . . .	5
3.4. YANG Module . . . . .	8
4. Security Considerations . . . . .	24
5. IANA Considerations . . . . .	24
6. Contributors . . . . .	24
7. Acknowledgements . . . . .	25
8. References . . . . .	25
8.1. Normative References . . . . .	25
8.2. Informative References . . . . .	26
Author's Addresses . . . . .	27

## 1. Introduction

Network Virtualization Overlays (NVO3), such as VXLAN, NVGRE, GENEVE and VXLAN-GPE, enable network virtualization for data center networks environment that assumes an IP-based underlay.

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document specifies a YANG data model that can be used to configure and manage NVO3 protocols. The model covers the configuration of NVO3 instances as well as their operation states, which are the basic common requirements of the different tunnel encapsulations. Thus it is called "the base model for NVO3" in this document.

As the Network Virtualization Overlay evolves, newly defined tunnel encapsulation may require extra configuration. For example, GENEVE may require configuration of TLVs at the NVE. The base module can be augmented to accommodate these new solutions.

## 2. Acronyms and Terminology

### 2.1. Acronyms

NVO3: Network Virtualization Overlays  
VNI: Virtual Network Instance  
BUM: Broadcast, Unknown Unicast, Multicast traffic

### 2.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Familiarity with [RFC7348], [RFC7364], [RFC7365] and [RFC8014] is assumed in this document.

## 3. The YANG Data Model for NVO3

The NVO3 base YANG model defined in this document is used to configure the NVEs. It is divided into three containers. The first container contains the configuration of the virtual network instances, e.g. the VNI, the NVE that the instance is mounted, the peer NVEs which can be determined dynamically via a control plane or given statically, and the statistical states of the instance. The other two containers are separately the statistical states of the

peer NVEs and the tunnels.

### 3.1 Mapping to the NVO3 architecture

The NVO3 base YANG model is defined according to the NVO3 architecture [RFC8014]. As shown in Figure 3.1, the reference model of the NVE defined in [RFC8014], multiple instances can be mounted under a NVE. The key of the instance is VNI. The source NVE of the instance is the NVE configured by the base YANG. An instance can have several peer NVEs. A NVO3 tunnel can be determined by the VNI, the source NVE and the peer NVE. The tunnel can be built statically by manually indicate the addresses of the peer NVEs, or dynamically via a control plane, e.g. EVPN [RFC8365]. An enabler is defined in the NVO3 base YANG to choose from these two modes.

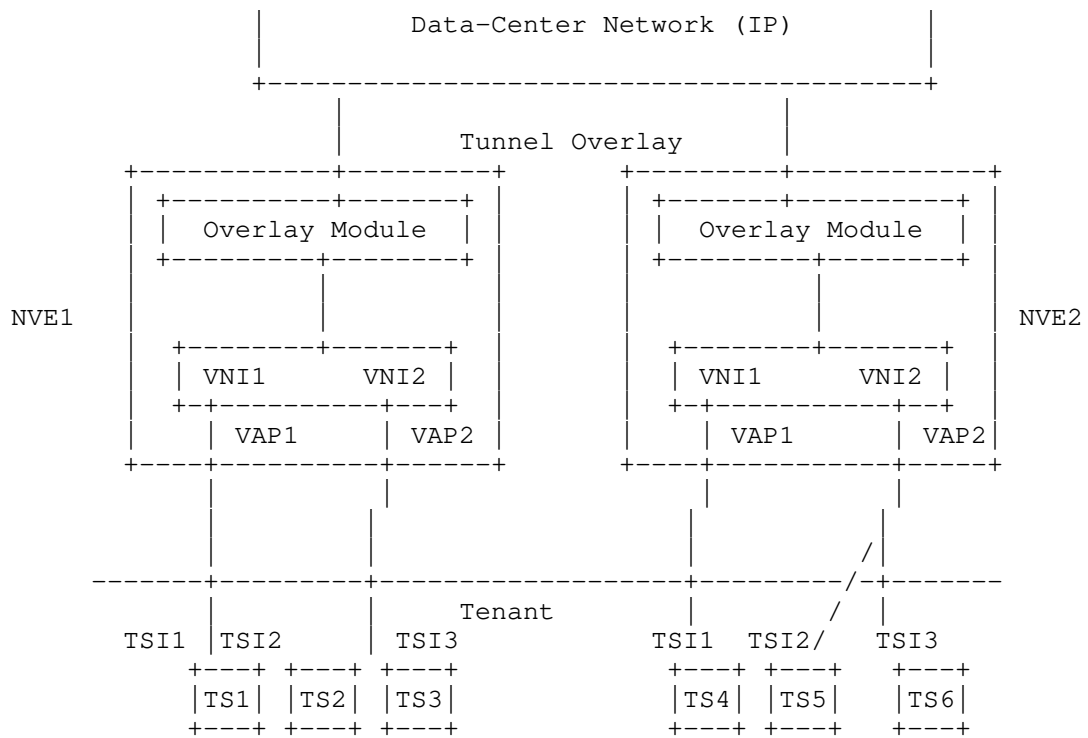


Figure 3.1. NVE Reference model in RFC 8014

### 3.2. The Configuration Parameters

#### 3.2.1. NVE as an interface

A NVE in the NVO3 base YANG is defined via augmenting the IETF

interface YANG. If anycast gateway is enabled, the source VTEP address is the address of the anycast gateway, and a bypass address is used to uniquely identify the NVE. Otherwise, the source VTEP address is the NVE interface's own IP address.

### 3.2.2. Virtual Network Instance

A Virtual Network Instance ('VNI') is a specific VN instance on an NVE [RFC7365]. At each NVE, a Tenant System is connect to VNIs through Virtual Access Points (VAP). VAPs can be physical ports or virtual ports identified by the bridge domain Identifier ('bdId'). The mapping between VNI and bdId is managed by the operator.

As defined in [draft-ietf-bess-evpn-inter-subnet-forwarding], a tenant can have multiple bridge domains, and each domain has its own VNI. Thus these VNIs are used as L2VPN. Besides, a dedicated VNI can be used for routing between the bridge domains, i.e. used as L3VPN. The mapping relationship between VNI and L2VPN (respectively, L3VPN) is given by augmenting the IETF YANG of L2VPN (respectively L3VPN).

### 3.2.3. BUM Mode

An NVE SHOULD support either ingress replication, or multicast proxy, or point to multipoint tunnels on a per-VNI basis. It is possible that both modes be used simultaneously in one NVO3 network by different NVEs.

If ingress replication is used, the receiver addresses are listed in 'peers'. If multicast proxy [RFC8293] is used, the proxy's address is given in "flood-proxy". If the choice is point to multipoint tunnels, the multicast address is given as 'multiAddr'.

### 3.3. Statistics

Operators can determine whether a NVE should gather statistic values on a per-VNI basis. An enabler is contained in the 'static' list as 'statistic-enable' leaf. If the gathering for a VNI is enabled, the statistical information about the local NVEs, the remote NVEs, the flows and the MAC addresses will be collected by the NVEs in this VNI.

### 3.3. Model Structure

```

module: ietf-nvo3-base
  +--rw nvo3
  |   +--rw vni-instances
  |   |   +--rw vni-instance* [vni-id]
  |   |   |   +--rw vni-id                uint32
  |   |   |   +--rw vni-mode                enumeration

```

```

+--rw source-nve                    if:interface-ref
+--rw protocol-bgp?                boolean
+--ro status?                      vni-status-type
+--rw static-ipv4-peers
|   +--rw static-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
|   |   +--rw out-vni-id?            uint32
+--rw static-ipv6-peers
|   +--rw static-ipv6-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv6-address-no-zone
+--rw flood-proxys
|   +--rw flood-proxy* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
+--rw mcast-groups
|   +--rw mcast-group* [mcast-ip]
|   |   +--rw mcast-ip                inet:ipv4-address-no-zone
+--rw statistic
|   +--rw statistic-enable?         boolean
|   +--ro statistic-info
|   |   +--ro rx-bits-per-sec?       uint64
|   |   +--ro rx-pkt-per-sec?        uint64
|   |   +--ro tx-bits-per-sec?       uint64
|   |   +--ro tx-pkt-per-sec?        uint64
|   |   +--ro rx-pkts?               uint64
|   |   +--ro rx-bytes?               uint64
|   |   +--ro tx-pkts?               uint64
|   |   +--ro tx-bytes?               uint64
|   |   +--ro rx-unicast-pkts?       uint64
|   |   +--ro rx-multicast-pkts?     uint64
|   |   +--ro rx-broadcast-pkts?     uint64
|   |   +--ro drop-unicast-pkts?     uint64
|   |   +--ro drop-multicast-pkts?   uint64
|   |   +--ro drop-broadcast-pkts?   uint64
|   |   +--ro tx-unicast-pkts?       uint64
|   |   +--ro tx-multicast-pkts?     uint64
|   |   +--ro tx-broadcast-pkts?     uint64
+--ro vni-peer-infos
|   +--ro peers
|   |   +--ro peer* [vni-id source-ip peer-ip]
|   |   |   +--ro vni-id            uint32
|   |   |   +--ro source-ip         inet:ip-address-no-zone
|   |   |   +--ro peer-ip           inet:ip-address-no-zone
|   |   |   +--ro tunnel-type?      peer-type
|   |   |   +--ro out-vni-id?       uint32
+--ro tunnel-infos
|   +--ro tunnel-info* [tunnel-id]
|   |   +--ro tunnel-id             uint32
|   |   +--ro source-ip?            inet:ip-address-no-zone

```

```

    +--ro peer-ip?          inet:ip-address-no-zone
    +--ro status?           tunnel-status
    +--ro type?             tunnel-type
    +--ro up-time?          string
    +--ro vrf-name?         -> /ni:network-instances/network-instance/name

augment /if:interfaces/if:interface:
  +--rw nvo3-nve
    +--rw nvo3-config
      +--rw source-vtep-ip?    inet:ipv4-address-no-zone
      +--rw source-vtep-ipv6?  inet:ipv6-address-no-zone
      +--rw bypass-vtep-ip?    inet:ipv4-address-no-zone
      +--rw statistics
        +--rw statistic* [vni-id mode peer-ip direction]
          +--rw vni-id          uint32
          +--rw mode            vni-type
          +--rw peer-ip         inet:ipv4-address-no-zone
          +--rw direction       direction-type
          +--ro info
            +--ro rx-pkts?      uint64
            +--ro rx-bytes?     uint64
            +--ro tx-pkts?      uint64
            +--ro tx-bytes?     uint64
            +--ro rx-unicast-pkts? uint64
            +--ro rx-multicast-pkts? uint64
            +--ro rx-broadcast-pkts? uint64
            +--ro tx-unicast-pkts? uint64
            +--ro tx-multicast-pkts? uint64
            +--ro tx-broadcast-pkts? uint64
            +--ro drop-unicast-pkts? uint64
            +--ro drop-multicast-pkts? uint64
            +--ro drop-broadcast-pkts? uint64
            +--ro rx-bits-per-sec? uint64
            +--ro rx-pkt-per-sec? uint64
            +--ro tx-bits-per-sec? uint64
            +--ro tx-pkt-per-sec? uint64
      +--rw nvo3-gateway
        +--rw nvo3-gateway
          +--rw vxlan-anycast-gateway? boolean
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l3vpn:l3vpn/l3
    vpn:l3vpn:
      +--rw vni-lists
        +--rw vni* [vni-id]
          +--rw vni-id          uint32
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l2vpn:l2vpn:
    +--rw vni-lists
      +--rw vni* [vni-id]
        +--rw vni-id          uint32
        +--rw split-horizon-mode? vni-bind-type

```



```

        +---rw split-group?          string

    rpcs:
      +---x reset-vni-instance-statistic
      |   +---w input
      |   |   +---w vni-id          uint32
      +---x reset-vni-peer-statistic
      |   +---w input
      |   |   +---w vni-id          uint32
      |   |   +---w mode            vni-type
      |   |   +---w peer-ip         inet:ipv4-address-no-zone
      |   |   +---w direction       direction-type

```

Figure 3.2. The tree structure of YANG module for NVO3 configuration

### 3.4. YANG Module

```

<CODE BEGINS> file "ietf-nvo3-base@2019-11-04.yang"
module ietf-nvo3-base {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3";
  prefix "nvo3";

  import ietf-network-instance {
    prefix "ni";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  import ietf-bgp-l3vpn {
    prefix "l3vpn";
  }

  organization "ietf";
  contact "ietf";
  description "Yang model for NVO3";

  revision 2019-11-04 {

```

```
    description
      "Cleaning non ietf-bgp-l3vpn related errors";
    reference
      "";
  }

  revision 2019-04-01 {
    description
      "Init revision";
    reference
      "";
  }

  typedef vni-status-type {
    type enumeration {
      enum "up" {
        description
          "Vni status up.";
      }
      enum "down" {
        description
          "Vni status down.";
      }
    }
    description
      "Vni status";
  }

  typedef vni-type {
    type enumeration {
      enum "l2" {
        description
          "layer 2 mode";
      }
      enum "l3" {
        description
          "layer 3 mode";
      }
    }
    description
      "vni type";
  }

  typedef peer-type {
    type enumeration {
      enum "static" {
        description
          "Static.";
      }
    }
  }
```

```

    }
    enum "dynamic" {
        description
            "Dynamic.";
    }
}
description
    "Peer type";
}

typedef tunnel-status {
    type enumeration {
        enum "up" {
            description
                "The tunnel is up.";
        }
        enum "down" {
            description
                "The tunnel is down.";
        }
    }
    description
        "Tunnel status";
}

typedef tunnel-type {
    type enumeration {
        enum "dynamic" {
            description
                "The tunnel is dynamic.";
        }
        enum "static" {
            description
                "The tunnel is static.";
        }
        enum "invalid" {
            description
                "The tunnel is invalid.";
        }
    }
    description
        "Tunnel type";
}

typedef direction-type {
    type enumeration {
        enum "inbound" {
            description
                "Inbound.";
        }
    }
}

```

```

    }
    enum "outbound" {
        description
            "Outbound.";
    }
    enum "bidirection" {
        description
            "Bidirection.";
    }
}
description
    "Bound direction";
}

typedef vni-bind-type {
    type enumeration {
        enum "hub-mode" {
            description
                "Hub mode.";
        }
        enum "spoke-mode" {
            description
                "Spoke mode.";
        }
    }
}
description
    "bdBindVniType";
}

container nvo3 {
    description
        "Management of NVO3.";

    container vni-instances {
        description
            "The confiuration and information table of the VNI.";
        list vni-instance {
            key "vni-id";
            must "(/if:interfaces/if:interface[if:name=current()/source-nve]/if:t
ype='Nve')";
            description
                "The confiuration and information of the VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
            }
            description
                "The id of VNI.";
        }
    }
}

```

```

    leaf vni-mode {
      type enumeration {
        enum "Local" {
          description
            "Local mode";
        }
        enum "Global" {
          description
            "Global mode";
        }
      }
      description
        "The mode of the VNI instance.";
    }
    leaf source-nve {
      type if:interface-ref;
      mandatory true;
      description
        "The name of the nve interface .";
    }
    leaf protocol-bgp {
      type boolean;
      default "false";
      description
        "Whether use bgp as vxlan's protocol.";
    }
    leaf status {
      type vni-status-type;
      config false;
      description
        "The status of the VNI.";
    }
    container static-ipv4-peers {
      description
        "The remote NVE address table in a same VNI.";
      list static-peer {
        key "peer-ip";
        description
          "The remote NVE address in a same VNI.";
        leaf peer-ip {
          type inet:ipv4-address-no-zone;
          description
            "The address of the NVE.";
        }
        leaf out-vni-id {
          type uint32 {
            range "1..16777215";
          }
        }
      }
    }

```

```

        description
            "The ID of the out VNI. Do not support separate deletion.";
    }
}
}
container static-ipv6-peers {
    description
        "The remote NVE ipv6 address table in a same VNI.";
    list static-ipv6-peer {
        key "peer-ip";
        description
            "The remote NVE ipv6 address in a same VNI.";
        leaf peer-ip {
            type inet:ipv6-address-no-zone;
            description
                "The ipv6 address of the NVE.";
        }
    }
}
container flood-proxys {
    description
        "The flood proxys for this VNI";
    list flood-proxy {
        key "peer-ip";
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "peer ip address";
        }
        description
            "List of the flood proxys";
    }
}
container mcast-groups {
    description
        "The mcast address table.";
    list mcast-group {
        key "mcast-ip";
        description
            "The mcast address.";
        leaf mcast-ip {
            type inet:ipv4-address-no-zone;
            description
                "The mcast address of NVO3.";
        }
    }
}
container statistic {

```

```

description
  "The VNI member in a same NVE.";
leaf statistic-enable {
  type boolean;
  default "false";
  description
    "To determine whether to enable the statistics for a VNI.";
}
container statistic-info {
  config false;
  description
    "The vni instance traffic statistics information.";
  leaf rx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits received per second.";
  }
  leaf rx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets received per second.";
  }
  leaf tx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits sent per second.";
  }
  leaf tx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets sent per second.";
  }
  leaf rx-pkts {
    type uint64;
    config false;
    description
      "Total number of received packets.";
  }
  leaf rx-bytes {
    type uint64;
    config false;
    description
      "Total number of received bytes.";
  }
}

```

```
leaf tx-pkts {
    type uint64;
    config false;
    description
        "Total number of sent packets.";
}
leaf tx-bytes {
    type uint64;
    config false;
    description
        "Total number of sent bytes.";
}
leaf rx-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of received unicast packets.";
}
leaf rx-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of received multicast packets.";
}
leaf rx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of received broadcast packets.";
}
leaf drop-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded unicast packets.";
}
leaf drop-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded multicast packets.";
}
leaf drop-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded broadcast packets.";
}
```





```

        description
            "The source address of the NVE interface.";
    }
    leaf peer-ip {
        type inet:ip-address-no-zone;
        config false;
        description
            "The remote NVE address.";
    }
    leaf tunnel-type {
        type peer-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf out-vni-id {
        type uint32 {
            range "1..16777215";
        }
        config false;
        description
            "The ID of the out VNI.";
    }
}

}

}

container tunnel-infos {
    config false;
    description
        "VxLAN tunnel information.";
    list tunnel-info {
        key "tunnel-id";
        config false;
        description
            "VxLAN tunnel information list.";
        leaf tunnel-id {
            type uint32 {
                range "1..4294967295";
            }
            config false;
            description
                "The ID of Vxlan tunnel.";
        }
        leaf source-ip {
            type inet:ip-address-no-zone;
            config false;
            description

```

```

        "Local NVE interface address.";
    }
    leaf peer-ip {
        type inet:ip-address-no-zone;
        config false;
        description
            "Remote NVE interface address.";
    }
    leaf status {
        type tunnel-status;
        config false;
        description
            "Tunnel status.";
    }
    leaf type {
        type tunnel-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf up-time {
        type string {
            length "1..10";
        }
        config false;
        description
            "Vxlan tunnel up time.";
    }
    leaf vrf-name {
        type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
        }
        default "_public_";
        config false;
        description
            "The name of VPN instance.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Augment the interface, NVE as an interface.";
    when "if:type = 'Nve'";
    container nvo3-nve {
        description
            "Network virtualization edge.";
        leaf source-vtep-ip {

```

```

    type inet:ipv4-address-no-zone;
    description
        "The source address of the NVE interface.";
}
leaf source-vtep-ipv6 {
    type inet:ipv6-address-no-zone;
    description
        "The source ipv6 address of the NVE interface.";
}
leaf bypass-vtep-ip {
    type inet:ipv4-address-no-zone;
    description
        "The source address of bypass VXLAN tunnel.";
}
container statistics {
    description
        "VXLAN Tunnel Traffic Statistical Configuration Table.";
    list statistic {
        key "vni-id mode peer-ip direction";
        description
            "VXLAN Tunnel Traffic Statistics Configuration.";
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
                "ID of the VNI.";
        }
        leaf mode {
            type vni-type;
            description
                "The type of the NVE interface.";
        }
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "IP address of the remote VTEP.";
        }
        leaf direction {
            type direction-type;
            must "(../mode='l3' and ../direction!='bidirection')";
            description
                "Traffic statistics type about the VXLAN tunnel.";
        }
    }
    container info {
        config false;
        description
            "Traffic statistics about the peer.";
    }
}

```

```
leaf rx-pkts {
    type uint64;
    config false;
    description
        "Total number of received packets.";
}
leaf rx-bytes {
    type uint64;
    config false;
    description
        "Total number of received bytes.";
}
leaf tx-pkts {
    type uint64;
    config false;
    description
        "Total number of sent packets.";
}
leaf tx-bytes {
    type uint64;
    config false;
    description
        "Total number of sent bytes.";
}
leaf rx-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of received unicast packets.";
}
leaf rx-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of received multicast packets.";
}
leaf rx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of received broadcast packets.";
}
leaf tx-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of sent unicast packets.";
}
```

```
leaf tx-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of sent multicast packets.";
}
leaf tx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of sent broadcast packets.";
}
leaf drop-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded unicast packets.";
}
leaf drop-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded multicast packets.";
}
leaf drop-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded broadcast packets.";
}
leaf rx-bits-per-sec {
    type uint64;
    config false;
    description
        "Number of bits received per second.";
}
leaf rx-pkt-per-sec {
    type uint64;
    config false;
    description
        "Number of packets received per second.";
}
leaf tx-bits-per-sec {
    type uint64;
    config false;
    description
        "Number of bits sent per second.";
}
```

```

        leaf tx-pkt-per-sec {
            type uint64;
            config false;
            description
                "Number of packets sent per second.";
        }
    }
}

}
container nvo3-gateway {
    when "/if:interfaces/if:interface/if:type = 'Vbdif'";
    description
        "Enable anycast gateway.";
    leaf vxlan-anycast-gateway {
        type boolean;
        default "false";
        description
            "Enable vxlan anycast gateway.";
    }
}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
    "/l3vpn:l3vpn/l3vpn:l3vpn" {
    description "Augment for l3vpn instance";
    container vni-lists {
        description "Vni list for l3vpn";
        list vni {
            key "vni-id";
            description
                "Vni for current l3vpn instance";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
            }
            description
                "The id of VNI.";
        }
    }
}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
    "/l2vpn:l2vpn" {
    description "Augment for l2vpn instance";
    container vni-lists {

```

```

description "Vni list for l2vpn";
list vni {
  key "vni-id";
  description
    "Vni for current l2vpn instance";
  leaf vni-id {
    type uint32 {
      range "1..16777215";
    }
    description
      "The id of VNI.";
  }
  leaf split-horizon-mode {
    type vni-bind-type;
    default "hub-mode";
    description
      "Split horizon mode.";
  }
  leaf split-group {
    type string {
      length "1..31";
    }
    description
      "Split group name.";
  }
}
}

rpc reset-vni-instance-statistic {
  description
    "Clear traffic statistics about the VNI.";
  input {
    leaf vni-id {
      type uint32 {
        range "1..16777215";
      }
      mandatory true;
      description
        "ID of the VNI.";
    }
  }
}

rpc reset-vni-peer-statistic {
  description
    "Clear traffic statistics about the VXLAN tunnel.";
  input {
    leaf vni-id {

```



```
        type uint32 {
            range "1..16777215";
        }
        mandatory true;
        description
            "ID of the VNI.";
    }
    leaf mode {
        type vni-type;
        mandatory true;
        description
            "The type of vni memeber statistic.";
    }
    leaf peer-ip {
        type inet:ipv4-address-no-zone;
        mandatory true;
        description
            "IP address of the remote NVE interface.";
    }
    leaf direction{
        type direction-type;
        must "(../mode='l3' and ../direction!='bidirection')";
        mandatory true;
        description
            "Traffic statistics type about the VXLAN tunnel.";
    }
}
}
```

<CODE ENDS>

#### 4. Security Considerations

This document raises no new security issues.

#### 5. IANA Considerations

The namespace URI defined in Section 3.3 need be registered in the IETF XML registry [RFC3688].

This document need to register the 'ietf-nvo3-base' YANG module in the YANG Module Names registry [RFC6020].

#### 6. Contributors

Haibo Wang  
Huawei

Email: rainsword.wang@huawei.com

Yuan Gao  
Huawei  
Email: sean.gao@huawei.com

Gang Yan  
Huawei  
Email: yangang@huawei.com

Mingui Zhang  
Huawei  
Email: zhangmingui@huawei.com

Yubao (Bob) Wang  
ZTE Corporation  
Email: yubao.wang2008@hotmail.com

Ruixue Wang  
China Mobile  
Email: wangruixue@chinamobile.com

Sijun Weng  
China Mobile  
Email: wengsijun@chinamobile.com

## 7. Acknowledgements

Authors would like to thank the comments and suggestions from Tao Han, Weilian Jiang.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7364] T. Narten, E. Gray, et al, "Problem Statement: Overlays for Network Virtualization", draft-ietf-nvo3-overlay-problem-statement, working in progress.
- [RFC7365] Marc Lasserre, Florin Balus, et al, "Framework for DC Network Virtualization", draft-ietf-nvo3-framework, working in progress.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger,

L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

[I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-10 (work in progress), March 2019.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten, An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3), RFC8014, December 2016.

## 8.2. Informative References

[RFC7637] M. Sridharan, A. Greenberg, et al, "NVGRE: Network Virtualization using Generic Routing Encapsulation", RFC7637, September 2015.

[I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.

[I-D.draft-ietf-bess-evpn-inter-subnet-forwarding] A. Sajassi, S. Salam, S. Thoria, J. Drake, J. Rabadan, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-08, March 4, 2019.

[RFC8293] A. Ghanwani, L. Dunbar, V. Bannai, M. McBride, R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC8293, January 2018.

Author's Addresses

Bing Liu  
Huawei Technologies  
No. 156 Beiqing Rd. Haidian District,  
Beijing 100095  
P.R. China

Email: remy.liubing@huawei.com

Ran Chen  
ZTE Corporation

Email: chen.ran@zte.com.cn

Fengwei Qin  
China Mobile  
32 Xuanwumen West Ave, Xicheng District  
Beijing, Beijing 100053  
China

Email: qinfengwei@chinamobile.com

Reshad Rahman  
Cisco Systems

Email: rrahman@cisco.com

NVO3  
Internet-Draft  
Intended status: Standards Track  
Expires: March 10, 2022

B. Liu, Ed.  
Huawei Technologies  
R. Chen  
ZTE Corporation  
F. Qin  
China Mobile  
R. Rahman  
September 6, 2021

Base YANG Data Model for NVO3 Protocols  
draft-ietf-nvo3-yang-cfg-05

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage Network Virtualization Overlay protocols. The model is focused on the common configuration requirement of various encapsulation options, such as VXLAN, NVGRE, GENEVE and VXLAN-GPE. Using this model as a starting point, incremental work can be done to satisfy the requirement of a specific encapsulation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 10, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Acronyms and Terminology . . . . .	3
2.1. Acronyms . . . . .	3
2.2. Terminology . . . . .	3
3. The YANG Data Model for NVO3 . . . . .	3
3.1. Mapping to the NVO3 architecture . . . . .	3
3.2. The Configuration Parameters . . . . .	4
3.2.1. NVE as an interface . . . . .	4
3.2.2. Virtual Network Instance . . . . .	4
3.2.3. BUM Mode . . . . .	5
3.3. Statistics . . . . .	5
3.4. Model Structure . . . . .	5
3.5. YANG Module . . . . .	8
4. Security Considerations . . . . .	22
5. IANA Considerations . . . . .	22
6. Contributors . . . . .	22
7. References . . . . .	23
7.1. Normative References . . . . .	23
7.2. Informative References . . . . .	25
Authors' Addresses . . . . .	25

## 1. Introduction

Network Virtualization Overlays (NVO3), such as VXLAN [RFC7348], NVGRE [RFC7637], GENEVE [I-D.ietf-nvo3-geneve] and VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], enable network virtualization for data center networks environment that assumes an IP-based underlay.

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document specifies a YANG data model that can be used to configure and manage NVO3 protocols. The model covers the configuration of NVO3 instances as well as their operation states, which are the basic common requirements of the different tunnel encapsulations. Thus it is called "the base model for NVO3" in this document.

As the Network Virtualization Overlay evolves, newly defined tunnel encapsulation may require extra configuration. For example, GENEVE

may require configuration of TLVs at the NVE. The base module can be augmented to accommodate these new solutions.

## 2. Acronyms and Terminology

### 2.1. Acronyms

NVO: Network Virtualization Overlays

VNI: Virtual Network Instance

BUM: Broadcast, Unknown Unicast, Multicast traffic

### 2.2. Terminology

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Familiarity with [RFC7348], [RFC7348], [RFC7364], [RFC7365] and [RFC8014] is assumed in this document.

## 3. The YANG Data Model for NVO3

The NVO3 base YANG model defined in this document is used to configure the NVEs. It is divided into three containers. The first container contains the configuration of the virtual network instances, e.g. the VNI, the NVE that the instance is mounted, the peer NVEs which can be determined dynamically via a control plane or given statically, and the statistical states of the instance. The other two containers are separately the statistical states of the peer NVEs and the tunnels.

### 3.1. Mapping to the NVO3 architecture

The NVO3 base YANG model is defined according to the NVO3 architecture [RFC8014]. As shown in Figure 1, the reference model of the NVE defined in [RFC8014], multiple instances can be mounted under a NVE. The key of the instance is VNI. The source NVE of the instance is the NVE configured by the base YANG. An instance can have several peer NVEs. A NVO3 tunnel can be determined by the VNI, the source NVE and the peer NVE. The tunnel can be built statically by manually indicate the addresses of the peer NVEs, or dynamically via a control plane, e.g. EVPN [RFC8365]. An enabler is defined in the NVO3 base YANG to choose from these two modes.

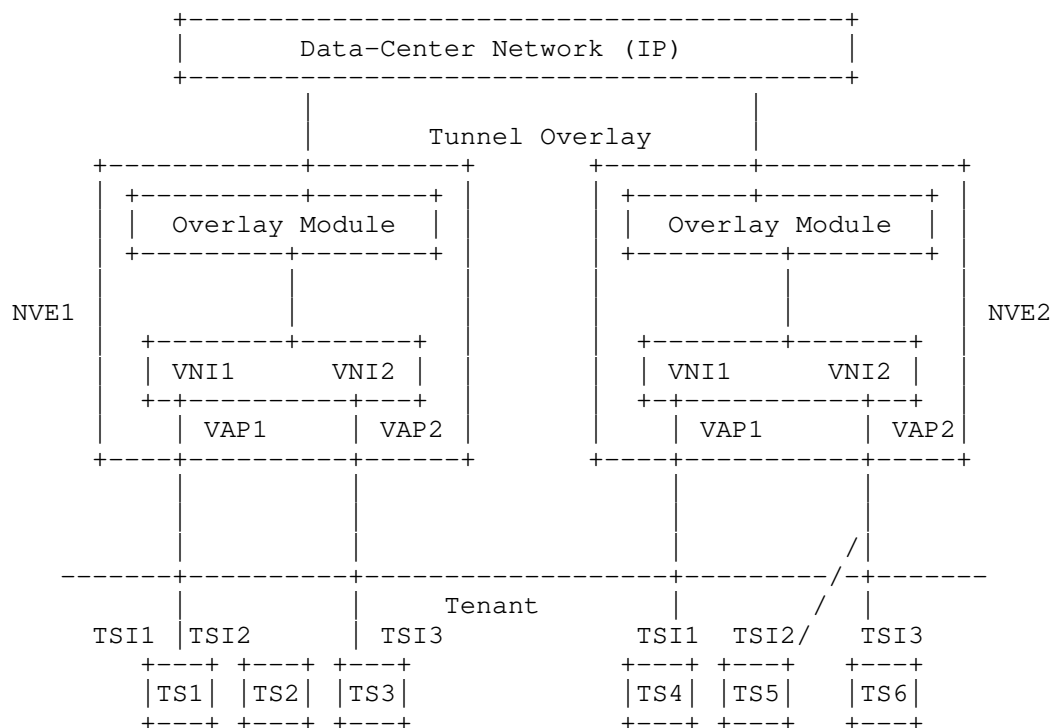


Figure 1: NVE Reference model in RFC8014

### 3.2. The Configuration Parameters

#### 3.2.1. NVE as an interface

A NVE in the NVO3 base YANG is defined via augmenting the IETF interface YANG. If anycast gateway is enabled, the source VTEP address is the address of the anycast gateway, and a bypass address is used to uniquely identify the NVE. Otherwise, the source VTEP address is the NVE interface's own IP address.

#### 3.2.2. Virtual Network Instance

A Virtual Network Instance ('VNI') is a specific VN instance on an NVE [RFC7365]. At each NVE, a Tenant System is connect to VNIs through Virtual Access Points (VAP). VAPs can be physical ports or virtual ports identified by the bridge domain Identifier ('bdId'). The mapping between VNI and bdId is managed by the operator.

As defined in [I-D.ietf-bess-evpn-inter-subnet-forwarding], a tenant can have multiple bridge domains, and each domain has its own VNI.



Thus these VNIs are used as L2VPN. Besides, a dedicated VNI can be used for routing between the bridge domains, i.e. used as L3VPN. The mapping relationship between VNI and L2VPN (respectively, L3VPN) is given by augmenting the IETF YANG of L2VPN (respectively L3VPN).

### 3.2.3. BUM Mode

An NVE SHOULD support either ingress replication, or multicast proxy, or point to multipoint tunnels on a per-VNI basis. It is possible that both modes be used simultaneously in one NVO3 network by different NVEs.

If ingress replication is used, the receiver addresses are listed in 'peers'. If multicast proxy [RFC8293] is used, the proxy's address is given in "flood-proxy". If the choice is point to multipoint tunnels, the multicast address is given as 'multiAddr'.

### 3.3. Statistics

Operators can determine whether a NVE should gather statistic values on a per-VNI basis. An enabler is contained in the 'static' list as 'statistic-enable' leaf. If the gathering for a VNI is enabled, the statistical information about the local NVEs, the remote NVEs, the flows and the MAC addresses will be collected by the NVEs in this VNI.

### 3.4. Model Structure

```
module: ietf-nvo3-base
+--rw nvo3
|   +--rw vni-instances
|       +--rw vni-instance* [vni-id]
|           +--rw vni-id                uint32
|           +--rw vni-mode?              vni-mode
|           +--rw source-nve             if:interface-ref
|           +--rw protocol-bgp?          boolean
|           +--ro status?                vni-status-type
|           +--rw static-ipv4-peers
|               +--rw static-peer* [peer-ip]
|                   +--rw peer-ip        inet:ipv4-address-no-zone
|                   +--rw out-vni-id?    uint32
|           +--rw static-ipv6-peers
|               +--rw static-ipv6-peer* [peer-ip]
|                   +--rw peer-ip        inet:ipv6-address-no-zone
|                   +--rw out-vni-id?    uint32
|           +--rw flood-proxys
|               +--rw flood-proxy* [peer-ip]
|                   +--rw peer-ip        inet:ip-address-no-zone
```

```

+--rw mcast-groups
|   +--rw mcast-group* [mcast-ip]
|       +--rw mcast-ip      inet:ip-address-no-zone
+--rw statistic
    +--rw enable?    boolean
    +--ro info
        +--ro send-bits-rate?      uint64
        +--ro send-pkts-rate?      uint64
        +--ro send-unicast-pkts?   uint64
        +--ro send-multicast-pkts? uint64
        +--ro send-broadcast-pkts? uint64
        +--ro send-total-bytes?    uint64
        +--ro send-total-pkts?     uint64
        +--ro receive-bits-rate?   uint64
        +--ro receive-pkts-rate?   uint64
        +--ro receive-unicast-pkts? uint64
        +--ro receive-multicast-pkts? uint64
        +--ro receive-broadcast-pkts? uint64
        +--ro receive-total-bytes? uint64
        +--ro receive-total-pkts?  uint64
        +--ro drop-unicast-pkts?   uint64
        +--ro drop-multicast-pkts? uint64
        +--ro drop-broadcast-pkts? uint64
+--ro vni-peer-infos
    +--ro peers
        +--ro peer* [vni-id source-ip peer-ip]
            +--ro vni-id      uint32
            +--ro source-ip   inet:ip-address-no-zone
            +--ro peer-ip     inet:ip-address-no-zone
            +--ro type?       tunnel-type
            +--ro out-vni-id? uint32
+--ro tunnel-infos
    +--ro tunnel-info* [tunnel-id]
        +--ro tunnel-id      uint32
        +--ro source-ip?     inet:ip-address-no-zone
        +--ro peer-ip?       inet:ip-address-no-zone
        +--ro status?        tunnel-status
        +--ro type?          tunnel-type
        +--ro up-time?       string
        +--ro vrf-name?      -> /ni:network-instances/network-instance/name

augment /if:interfaces/if:interface:
+--rw nvo3-nve
    +--rw nve-ip?      inet:ipv4-address-no-zone
    +--rw nve-ipv6?    inet:ipv6-address-no-zone
    +--rw bypass-nve-ip? inet:ipv4-address-no-zone
    +--rw bypass-nve-ipv6? inet:ipv6-address-no-zone
    +--rw statistics

```

```

    +---rw statistic* [vni-id peer-ip direction]
      +---rw vni-id      uint32
      +---rw peer-ip     inet:ip-address-no-zone
      +---rw direction   direction-type
      +---ro info
        +---ro send-bits-rate?      uint64
        +---ro send-pkts-rate?      uint64
        +---ro send-unicast-pkts?   uint64
        +---ro send-multicast-pkts? uint64
        +---ro send-broadcast-pkts? uint64
        +---ro send-total-bytes?    uint64
        +---ro send-total-pkts?     uint64
        +---ro receive-bits-rate?   uint64
        +---ro receive-pkts-rate?   uint64
        +---ro receive-unicast-pkts? uint64
        +---ro receive-multicast-pkts? uint64
        +---ro receive-broadcast-pkts? uint64
        +---ro receive-total-bytes? uint64
        +---ro receive-total-pkts?  uint64
        +---ro drop-unicast-pkts?   uint64
        +---ro drop-multicast-pkts? uint64
        +---ro drop-broadcast-pkts? uint64

augment /ni:network-instances/ni:network-instance/ni:ni-type/l3vpn:l3vpn/l3vpn:
l3vpn:
  +---rw vnis
    +---rw vni* [vni-id]
      +---rw vni-id      uint32

augment /ni:network-instances/ni:network-instance/ni:ni-type/l2vpn:l2vpn:
l2vpn:
  +---rw vnis
    +---rw vni* [vni-id]
      +---rw vni-id      uint32
      +---rw split-horizon-mode? vni-bind-type
      +---rw split-group?   string

rpcs:
  +---x reset-vni-instance-statistic
  |   +---w input
  |   |   +---w vni-id      uint32
  +---x reset-vni-peer-statistic
  |   +---w input
  |   |   +---w vni-id      uint32
  |   |   +---w peer-ip     inet:ip-address-no-zone
  |   |   +---w direction   direction-type

```

### 3.5. YANG Module

```
<CODE BEGINS> file "ietf-nvo3-base@2021-03-08.yang"

module ietf-nvo3-base {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3-base";
  prefix "nvo3";

  import ietf-network-instance {
    prefix "ni";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  import ietf-bgp-l3vpn {
    prefix "l3vpn";
  }

  import iana-if-type {
    prefix ianaift;
  }

  organization "ietf";
  contact "ietf";
  description "Yang model for NVO3.";
  revision 2021-03-08 {
    description
      "Fix the keyword 'must' order issue in the leaf source-nve";
    reference
      "";
  }
  revision 2020-08-26 {
    description
      "Clean non ietf-bgp-l3vpn & ietf-l2vpn related errors.";
    reference
      "";
  }
}
```

```
revision 2020-07-22 {
  description
    "Solve syntax and norms issues.";
  reference
    "";
}

revision 2020-03-09 {
  description
    "Revise some design in the statistics.";
  reference
    "";
}

revision 2019-11-04 {
  description
    "Cleaning non ietf-bgp-l3vpn related errors.";
  reference
    "";
}

revision 2019-04-01 {
  description
    "Init revision.";
  reference
    "";
}

typedef vni-status-type {
  type enumeration {
    enum "up" {
      description
        "The state is up.";
    }
    enum "down" {
      description
        "The state is down.";
    }
  }
  description
    "The state for VNI.";
}

typedef tunnel-status {
  type enumeration {
    enum "up" {
      description
        "The tunnel is up.";
    }
  }
}
```

```
    }
    enum "down" {
        description
            "The tunnel is down.";
    }
}
description
    "The status of NVO3 Tunnel.";
}
typedef tunnel-type {
    type enumeration {
        enum "dynamic" {
            description
                "The tunnel is dynamic.";
        }
        enum "static" {
            description
                "The tunnel is static.";
        }
        enum "invalid" {
            description
                "The tunnel is invalid.";
        }
    }
}
description
    "The type of NVO3 Tunnel.";
}

typedef direction-type {
    type enumeration {
        enum "inbound" {
            description
                "Inbound.";
        }
        enum "outbound" {
            description
                "Outbound.";
        }
        enum "bidirection" {
            description
                "Bidirection.";
        }
    }
}
description
    "Bound direction.";
}
typedef vni-bind-type {
    type enumeration {
```

```

        enum "hub-mode" {
            description
                "Hub mode. The vni instance can't communicate with other hub mode vni i
nstances.";
        }
        enum "spoke-mode" {
            description
                "Spoke mode.";
        }
        enum "split-group-mode" {
            description
                "Split group mode.";
        }
    }
    description
        "The binding type of VNI.";
}

typedef vni-mode {
    type enumeration {
        enum "local" {
            description
                "Local mode.";
        }
        enum "global" {
            description
                "Global mode.";
        }
    }
    description
        "The mode of VNI.";
}

grouping nvo3-traffic-statistics {
    description
        "NVO3 tunnel traffic statistics collection.";
    leaf send-bits-rate {
        type uint64;
        units bit/s;
        description
            "Number of send bits per second.";
    }
    leaf send-pkts-rate {
        type uint64;
        units pps;
        description
            "Number of send packets per second.";
    }
    leaf send-unicast-pkts {

```

```
    type uint64;
    units packet;
    description
        "Number of send unicast packets.";
}
leaf send-multicast-pkts {
    type uint64;
    units packet;
    description
        "Number of send multicast packets.";
}
leaf send-broadcast-pkts {
    type uint64;
    units packet;
    description
        "Number of send broadcast packets.";
}
leaf send-total-bytes {
    type uint64;
    units Byte;
    description
        "Total number of send bytes.";
}
leaf send-total-pkts {
    type uint64;
    units packet;
    description
        "Total number of send packets.";
}
leaf receive-bits-rate {
    type uint64;
    units bit/s;
    description
        "Number of receive bits per second.";
}
leaf receive-pkts-rate {
    type uint64;
    units pps;
    description
        "Number of receive packets per second.";
}
leaf receive-unicast-pkts {
    type uint64;
    units packet;
    description
        "Number of receive unicast packets.";
}
leaf receive-multicast-pkts {
```



```

    type uint64;
    units packet;
    description
        "Number of receive multicast packets.";
}
leaf receive-broadcast-pkts {
    type uint64;
    units packet;
    description
        "Number of receive broadcast packets.";
}
leaf receive-total-bytes {
    type uint64;
    units Byte;
    description
        "Total number of receive bytes.";
}
leaf receive-total-pkts {
    type uint64;
    units packet;
    description
        "Total number of receive packets.";
}
leaf drop-unicast-pkts {
    type uint64;
    units packet;
    description
        "Number of discarded unicast packets.";
}
leaf drop-multicast-pkts {
    type uint64;
    units packet;
    description
        "Number of discarded multicast packets.";
}
leaf drop-broadcast-pkts {
    type uint64;
    units packet;
    description
        "Number of discarded broadcast packets.";
}
}

container nvo3 {
    description
        "Management of NV03.";
    container vni-instances {
        description

```

```

    "List of virtual network instances.";
list vni-instance {
    key "vni-id";
    description
        "Configure the information of VNI.";
    leaf vni-id {
        type uint32 {
            range "1..16777215";
        }
        description
            "The id of VNI.";
    }
    leaf vni-mode {
        type vni-mode;
        default "local";
        description
            "The mode of VNI.";
    }
    leaf source-nve {
        type if:interface-ref;
        must "(/if:interfaces/if:interface[if:name=current()]/if:type='Nve')";
        mandatory true;
        description
            "The name of the local NVE.";
    }
    leaf protocol-bgp {
        type boolean;
        default "false";
        description
            "Learn remote NVEs in the same VNI via BGP.";
    }
    leaf status {
        type vni-status-type;
        config false;
        description
            "The status of the VNI.";
    }
    container static-ipv4-peers {
        description
            "List of remote NVE address created by users in a VNI.";
        list static-peer {
            key "peer-ip";
            description
                "Configure remote NVE address in a same VNI.";
            leaf peer-ip {
                type inet:ipv4-address-no-zone;
                description
                    "The address of the remote NVE.";
            }
        }
    }
}

```

```

    }
    leaf out-vni-id {
        type uint32 {
            range "1..16777215";
        }
        description
            "The ID of VNI for outbound. Do not support separate deletion.";
    }
}
}
container static-ipv6-peers {
    description
        "List of remote NVE IPv6 address created by users in a VNI.";
    list static-ipv6-peer {
        key "peer-ip";
        description
            "Configure remote NVE IPv6 address in a same VNI.";
        leaf peer-ip {
            type inet:ipv6-address-no-zone;
            description
                "The IPv6 address of the remote NVE.";
        }
        leaf out-vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
                "The ID of VNI for outbound. Do not support separate deletion.";
        }
    }
}
}
container flood-proxys {
    description
        "List of flood proxys for the VNI.";
    list flood-proxy {
        key "peer-ip";
        description
            "Configure flood proxys for the VNI.";
        leaf peer-ip {
            type inet:ip-address-no-zone;
            description
                "The address of flood proxy.";
        }
    }
}
}
container mcast-groups {
    description
        "List of multicast address for the VNI.";
}

```

```

    list mcast-group {
        key "mcast-ip";
        description
            "Configure multicast address in a same VNI.";
        leaf mcast-ip {
            type inet:ip-address-no-zone;
            description
                "The mcast address of NVO3.";
        }
    }
}
container statistic {
    description
        "Configure VNI traffic statistics.";
    leaf enable {
        type boolean;
        default "false";
        description
            "Enable/disable VNI traffic statistics.";
    }
    container info {
        when "../enable='true'";
        config false;
        description
            "The information of vni instance traffic statistics.";
        uses nvo3-traffic-statistics;
    }
}
}
}
}
container vni-peer-infos {
    config false;
    description
        "List of remote NVE addresses.";
    container peers {
        config false;
        description
            "Operational data of remote NVE address in a VNI.";
        list peer {
            key "vni-id source-ip peer-ip";
            config false;
            description
                "Operational data of remote NVE addresses in a VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
            }
        }
    }
}

```

```

        config false;
        description
            "The ID of VNI.";
    }
    leaf source-ip {
        type inet:ip-address-no-zone;
        config false;
        description
            "Local NVE address, as NVO3 tunnel source point.";
    }
    leaf peer-ip {
        type inet:ip-address-no-zone;
        config false;
        description
            "Remote NVE address, as NVO3 tunnel end point.";
    }
    leaf type {
        type tunnel-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf out-vni-id {
        type uint32 {
            range "1..16777215";
        }
        config false;
        description
            "The ID of VNI for outbound.";
    }
}
}
}

container tunnel-infos {
    config false;
    description
        "List of NVO3 tunnel information.";
    list tunnel-info {
        key "tunnel-id";
        config false;
        description
            "Operational data of NVO3 tunnel information.";
        leaf tunnel-id {
            type uint32 {
                range "1..4294967295";
            }
        }
        config false;
    }
}

```

```

        description
            "The ID of NVO3 tunnel.";
    }
    leaf source-ip {
        type inet:ip-address-no-zone;
        config false;
        description
            "Local NVE address, as NVO3 tunnel source point.";
    }
    leaf peer-ip {
        type inet:ip-address-no-zone;
        config false;
        description
            "Remote NVE address, as NVO3 tunnel end point.";
    }
    leaf status {
        type tunnel-status;
        config false;
        description
            "Tunnel status.";
    }
    leaf type {
        type tunnel-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf up-time {
        type string {
            length "1..10";
        }
        config false;
        description
            "The continuous time as NVO3 tunnel is reachable.";
    }
    leaf vrf-name {
        type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
        }
        default "_public_";
        config false;
        description
            "The name of VPN instance.";
    }
}

identity Nve {

```

```

    base ianaift:iana-interface-type;
    description "A new interface type to be registered to IANA";
}

augment "/if:interfaces/if:interface" {
    when "(/if:interfaces/if:interface/if:type = 'nvo3:Nve')";
    description
        "Augment the interface, NVE as an interface.";
    container nvo3-nve {
        description
            "Local NVE.";
        leaf nve-ip {
            type inet:ipv4-address-no-zone;
            description
                "The address of local NVE.";
        }
        leaf nve-ipv6 {
            type inet:ipv6-address-no-zone;
            description
                "The IPv6 address of the local NVE.";
        }
        leaf bypass-nve-ip {
            type inet:ipv4-address-no-zone;
            description
                "The address of local NVE as bypass.";
        }
        leaf bypass-nve-ipv6 {
            type inet:ipv6-address-no-zone;
            description
                "The IPv6 address of local NVE as bypass.";
        }
    }
    container statistics {
        description
            "List of NVO3 tunnel statistics.";
        list statistic {
            key "vni-id peer-ip direction";
            description
                "Configure NVO3 tunnel statistics information.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "The ID of the VNI.";
            }
            leaf peer-ip {
                type inet:ip-address-no-zone;
                description

```

```

        "The address of remote NVE.";
    }
    leaf direction {
        type direction-type;
        description
            "Traffic statistics direction for the tunnel.";
    }
    container info {
        config false;
        description
            "The information of tunnel traffic statistics.";
        uses nvo3-traffic-statistics;
    }
}
}
}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
    "/l3vpn:l3vpn/l3vpn:l3vpn" {
    description "Augment for l3vpn instance";
    container vnis {
        description "Vni list for l3vpn.";
        list vni {
            key "vni-id";
            description
                "Vni for current l3vpn instance.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "The ID of the VNI.";
            }
        }
    }
}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
    "/l2vpn:l2vpn" {
    description "Augment for l2vpn instance.";
    container vnis {
        description "Vni list for l2vpn.";
        list vni {
            key "vni-id";
            description
                "Vni for current l2vpn instance.";
            leaf vni-id {

```



```

        type uint32 {
            range "1..16777215";
        }
        description
            "The ID of the VNI.";
    }
    container split-horizon {
        description "Configure NVO3 split-horizon information.";
        leaf split-horizon-mode {
            type vni-bind-type;
            default "hub-mode";
            description
                "Split horizon mode.";
        }
        leaf split-group {
            when "(../split-horizon-mode='split-group-mode')";
            type string {
                length "1..31";
            }
            description
                "Split group name.";
        }
    }
}

rpc reset-vni-instance-statistic {
    description
        "Clear traffic statistics about the VNI.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "The ID of the VNI.";
        }
    }
}

rpc reset-vni-peer-statistic {
    description
        "Clear traffic statistics about the VXLAN tunnel.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
        }
    }
}

```

```
    }
    mandatory true;
    description
      "The ID of the VNI.";
  }
  leaf peer-ip {
    type inet:ip-address-no-zone;
    mandatory true;
    description
      "The address of the remote NVE.";
  }
  leaf direction{
    type direction-type;
    mandatory true;
    description
      "Traffic statistics direction for the tunnel.";
  }
}
}
```

<CODE ENDS>

#### 4. Security Considerations

This document raises no new security issues.

#### 5. IANA Considerations

The namespace URI defined in Section 3.4 need to be registered in the IETF XML registry [RFC3688].

This document need to register the 'ietf-nvo3-base' YANG module in the YANG Module Names registry [RFC6020].

#### 6. Contributors

Haibo Wang  
Huawei  
Email: rainsword.wang@huawei.com

Yuan Gao  
Huawei  
Email: sean.gao@huawei.com

Guannan Shi  
Huawei  
Email: shiguannan1@huawei.com

Gang Yan  
Huawei  
Email: yangang@huawei.com

Mingui Zhang  
Huawei  
Email: zhangmingui@huawei.com

Yubao Wang  
ZTE Corporation  
Email: yubao.wang2008@hotmail.com

Ruixue Wang  
China Mobile  
Email: wangruixue@chinamobile.com

Sijun Weng  
China Mobile  
Email: wengsijun@chinamobile.com

This document is part of a plan to make xml2rfc indispensable.

## 7. References

### 7.1. 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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7364] Narten, T., Ed., Gray, E., Ed., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, DOI 10.17487/RFC7364, October 2014, <<https://www.rfc-editor.org/info/rfc7364>>.
- [RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <<https://www.rfc-editor.org/info/rfc7365>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [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>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.

## 7.2. Informative References

- [I-D.ietf-bess-evpn-inter-subnet-forwarding]  
Sajassi, A., Salam, S., Thoria, S., Drake, J. E., and J. Rabadan, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-15 (work in progress), July 2021.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.
- [I-D.ietf-nvo3-vxlan-gpe]  
(Editor), F. M., (editor), L. K., and U. E. (editor), "Generic Protocol Extension for VXLAN (VXLAN-GPE)", draft-ietf-nvo3-vxlan-gpe-11 (work in progress), March 2021.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC8293] Ghanwani, A., Dunbar, L., McBride, M., Bannai, V., and R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC 8293, DOI 10.17487/RFC8293, January 2018, <<https://www.rfc-editor.org/info/rfc8293>>.

## Authors' Addresses

Bing Liu (editor)  
Huawei Technologies  
No. 156 Beiqing Rd. Haidian District  
Beijing 100095  
China

Email: [remy.liubing@huawei.com](mailto:remy.liubing@huawei.com)

Ran Chen  
ZTE Corporation

Email: [chen.ran@zte.com.cn](mailto:chen.ran@zte.com.cn)

Fengwei Qin  
China Mobile  
32 Xuanwumen West Ave, Xicheng District  
Beijing 100053  
China

Email: qinfengwei@chinamobile.com

Reshad Rahman

Email: reshad@yahoo.com

TSVWG  
Internet-Draft  
Intended status: Informational  
Expires: January 8, 2020

Y. Li  
X. Zhou  
Huawei  
M. Boucadair  
Orange  
J. Wang  
China Telecom  
July 07, 2019

LOOPS (Localized Optimizations on Path Segments) Problem Statement and  
Opportunities for Network-Assisted Performance Enhancement  
draft-li-tsvwg-loops-problem-opportunities-03

Abstract

In various network deployments, end to end forwarding paths are partitioned into multiple segments. For example, in some cloud-based WAN communications, stitching multiple overlay tunnels are used for traffic policy enforcement matters such as to optimize traffic distribution or to select paths exposing a lower latency. Likewise, in satellite communications, the communication path is decomposed into two terrestrial segments and a satellite segment. Such long-haul paths are naturally composed of multiple network segments with various encapsulation schemes. Packet loss may show different characteristics on different segments.

Traditional transport protocols (e.g., TCP) respond to packet loss slowly especially in long-haul networks: they either wait for some signal from the receiver to indicate a loss and then retransmit from the sender or rely on sender's timeout which is often quite long. Non-congestive loss may make the TCP sender over-reduce the sending rate unnecessarily. With the increase of end-to-end transport encryption (e.g., QUIC), traditional PEP (performance enhancing proxy) techniques such as TCP splitting are no longer applicable.

LOOPS (Local Optimizations on Path Segments) is a network-assisted performance enhancement over path segment and it aims to provide local in-network recovery to achieve better data delivery by making packet loss recovery faster and by avoiding the senders over-reducing their sending rate. In an overlay network scenario, LOOPS can be performed over a variety of the existing, or purposely created, tunnel-based path segments.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 8, 2020.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. The Problem . . . . .	3
1.2. Sketching a Work Direction: Rationale & Goals . . . . .	4
2. Terminology . . . . .	6
3. Cloud-Internet Overlay Network . . . . .	7
3.1. Tail Loss or Loss in Short Flows . . . . .	9
3.2. Packet Loss in Real Time Media Streams . . . . .	9
3.3. Packet Loss and Congestion Control in Bulk Data Transfer . . . . .	10
3.4. Multipathing . . . . .	10
4. Satellite Communication . . . . .	11
5. Branch Office WAN Connection . . . . .	13
6. Features and Impacts to be Considered for LOOPS . . . . .	14
6.1. Local Recovery and End-to-end Retransmission . . . . .	15
6.1.1. OE to OE Measurement, Recovery, and Multipathing . . . . .	17



6.2. Congestion Control Interaction . . . . .	18
6.3. Overlay Protocol Extensions . . . . .	19
6.4. Summary . . . . .	20
7. Security Considerations . . . . .	20
8. IANA Considerations . . . . .	21
9. Acknowledgements . . . . .	21
10. Informative References . . . . .	21
Authors' Addresses . . . . .	24

## 1. Introduction

### 1.1. The Problem

Tunnels are widely deployed within many networks to achieve various engineering goals, including long-haul WAN interconnection or enterprise wireless access networks. A connection between two endpoints can be decomposed into many connection legs. As such, the corresponding forwarding path can be partitioned into multiple path segments that some of them are using network overlays by means of tunnels. This design serves a number of purposes such as steering the traffic, optimize egress/ingress link utilization, optimize traffic performance metrics (such as delay, delay variation, or loss), optimize resource utilization by invoking resource bonding, provide high-availability, etc.

A reliable transport layer normally employs some end-to-end retransmission mechanisms which also address congestion control [RFC0793] [RFC5681]. The sender either waits for the receiver to send some signals on a packet loss or sets some form of timeout for retransmission. For unreliable transport protocols such as RTP [RFC3550], optional and limited usage of end-to-end retransmission is employed to recover from packet loss [RFC4585] [RFC4588].

End-to-end retransmission to recover lost packets is slow especially when the network is long-haul. When a path is partitioned into multiple path segments that are realized typically as overlay tunnels, LOOPS (Local Optimizations on Path Segments) aims to provide local segment based in-network recovery to achieve better data delivery by making packet loss recovery faster and by avoiding the senders over-reducing their sending rate. In an overlay network scenario, LOOPS can be performed over the existing, or purposely created, overlay tunnel based path segments. Figure 1 show a basic usage scenario of LOOPS.

Some link types (satellite, microwave, drone-based networking, etc.) may exhibit unusually high loss rate in special conditions (e.g., fades due to heavy rain). The traditional TCP sender interprets loss as congestion and over-reduces the sending rate, degrading the

throughput. LOOPS is also applicable to such scenarios to improve the throughput.

Also, multiple paths may be available in the network that may be used for better performance. These paths are not visible to endpoints. Means to make use of these paths while ensuring the overall performance is enhanced would contribute to customer satisfaction. Blindly implementing link aggregation may lead to undesired effects (e.g., underperform compared to single path).

## 1.2. Sketching a Work Direction: Rationale & Goals

This document sketches a proposal that is meant to experimentally investigate to what extent a network-assisted approach can contribute to increase the overall perceived quality of experience in specific situations (e.g., Sections 3.5 and 3.6 of [RFC8517]) without requiring access to internal transport primitives. The rationale beneath this approach is that some information (loss detection, better visibility on available paths and their characteristics, etc.) can be used to trigger local actions while avoiding as much as possible undesired side effects (e.g., expose a behavior that would be interpreted by an endpoint as an anomaly (corrupt data) and which would lead to exacerbate end-to-end recovery. Such local actions would have a faster effect (e.g., faster recovery, used multiple paths simultaneously).

To that aim, the work is structured into two (2) phased stages:

- o Stage 1: Network-assisted optimization. This one assumes that optimizations (e.g., support latency-sensitive applications) can be implemented at the network without requiring defining new interaction with the endpoint. Existing tools such as ECN will be used. Some of these optimizations may be valuable in deployments where communications are established over paths that are not exposing the same performance characteristics.
- o Stage 2: Collaborative networking optimization. This one requires more interaction between the network and an endpoint to implement coordinated and more surgical network-assisted optimizations based on information/instructions shared by an endpoint or sharing locally-visible information with endpoint for better and faster recovery.

The document focuses on the first stage. Effort related to the second stage is out of scope of the initial planned work. Nevertheless, future work will be planned once progress is (hopefully) made on the first stage.

The proposed mechanism is not meant to be applied to all traffic, but only to a subset which is eligible to the network-assisted optimization service.

Which traffic is eligible is deployment-specific and policy-based. For example, techniques for dynamic information of optimization function (e.g., SFC) may be leveraged to unambiguously identify the aggregate of traffic that is eligible to the service. Such identification may be triggered by subscription actions made by customers or be provided by a network provider (e.g., specific-applications, during specific events such as during severe DDoS attack or flash crowds events).

Likewise, whether the optimization function is permanently instantiated or on-demand is deployment-specific.

This document does not intend to provide a comprehensive list of target deployment cases. Sample scenarios are described to illustrate some LOOPS potentials. Similar issues and optimizations may be helpful in other deployments such as enhancing the reliability of data transfer when a fleet of drones are used for specific missions (e.g., site inspection, live streaming, and emergency service). Captured data should be reliably transmitted via paths involving radio connections.

It is not required that all segments are LOOPS-aware to benefit from LOOPS advantages.

Section 3 presents some of the issues and opportunities found in Cloud-Internet overlay networks that require higher performance and more reliable packet transmission over best effort networks. Section 4 discusses applications of LOOPS in satellite communication. Section 6 describes the corresponding solution features and their impact on existing network technologies.

ON=overlay node  
UN=underlay node

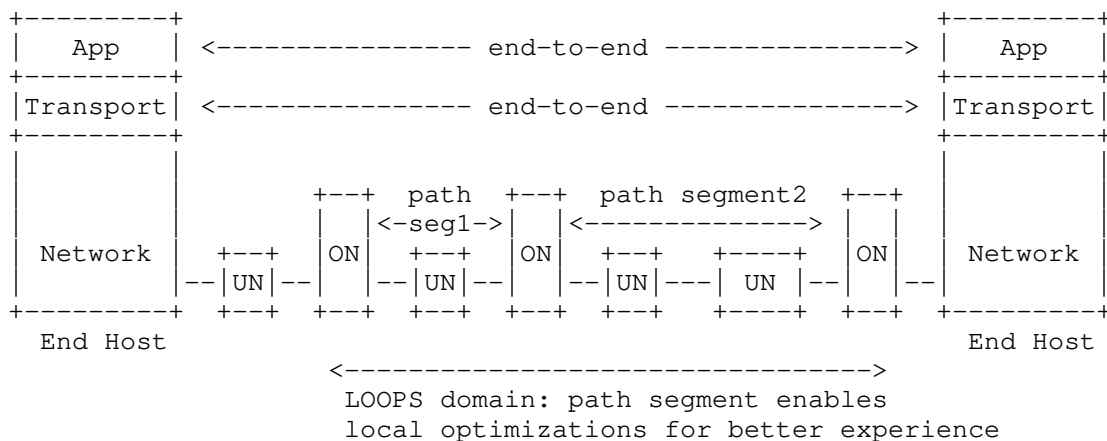


Figure 1: LOOPS in Overlay Network Usage Scenario

## 2. Terminology

This document makes use of the following terms:

LOOPS: Local Optimizations on Path Segments. LOOPS includes to the local in-network (i.e., non end-to-end) recovery functions and other supporting features such as local measurement, loss detection, and congestion feedback.

LOOPS Node: A node supporting LOOPS functions.

Overlay Node (ON): A node having overlay functions (e.g., overlay protocol encapsulation/decapsulation, header modification, TLV inspection) and LOOPS functions in LOOPS overlay network usage scenario.

Overlay Tunnel: A tunnel with designated ingress and egress nodes using some network overlay protocol as encapsulation, optionally with a specific traffic type.

Overlay Edge (OE): Edge node of an overlay tunnel. It can behave as ingress or egress as a function of the traffic direction.

Path segment: A LOOPS enabled tunnel-based network subpath. It is used interchangeably with overlay segment in this document when the context wants to emphasize on its overlay encapsulated nature. It is also called segment for simplicity in this document.

Overlay segment: Refers to path segment.

Underlay Node (UN): A node not participating in the overlay network.

### 3. Cloud-Internet Overlay Network

CSPs (Cloud Service Providers) are connecting their data centers using the Internet or via self-constructed networks/links. This expands the traditional Internet's infrastructure and, together with the original ISP's infrastructure, forms the Internet underlay.

Automation techniques and NFV (Network Function Virtualization) further ambitions to make it easier to dynamically provision a new virtual node/function as a workload in a cloud for CPU/storage intensive functions. With the aid of various mechanisms such as kernel bypassing and Virtual IO, forwarding based on virtual nodes is becoming more and more effective. The interconnection among the purposely positioned virtual nodes and/or the existing nodes with virtualization functions potentially form an overlay infrastructure. It is called the Cloud-Internet Overlay Network (CION) in this document for short.

This architecture scenario makes use of overlay technologies to direct the traffic going through the specific overlay path regardless of the underlying physical topology, in order to achieve better service delivery. It purposely creates or selects overlay nodes (ON) from providers. By continuously measuring the delay of path segments and use them as metrics for path selection, when the number of overlay nodes is sufficiently large, there is a high chance that a better path could be found [DOI\_10.1109\_ICDCS.2016.49] [DOI\_10.1145\_3038912.3052560]. [DOI\_10.1145\_3038912.3052560] further shows all cloud providers experience random loss episodes and random loss accounts for more than 35% of total loss.

Some of the considerations that are discussed below may also apply for interconnecting DCs owned by a network provider.

Figure 2 shows an example of an overlay path over large geographic distances. Three path segments, i.e., ON1-ON2, ON2-ON3, ON3-ON4 are shown. ON is usually a virtual node, though it does not have to be. Each segment transmits packets using some form of network overlay protocol encapsulation. ON has the computing and memory resources that can be used for some functions like packet loss detection, network measurement and feedback, and packet recovery. ONs are managed by a single administrator though they can be workloads created from different CSPs.

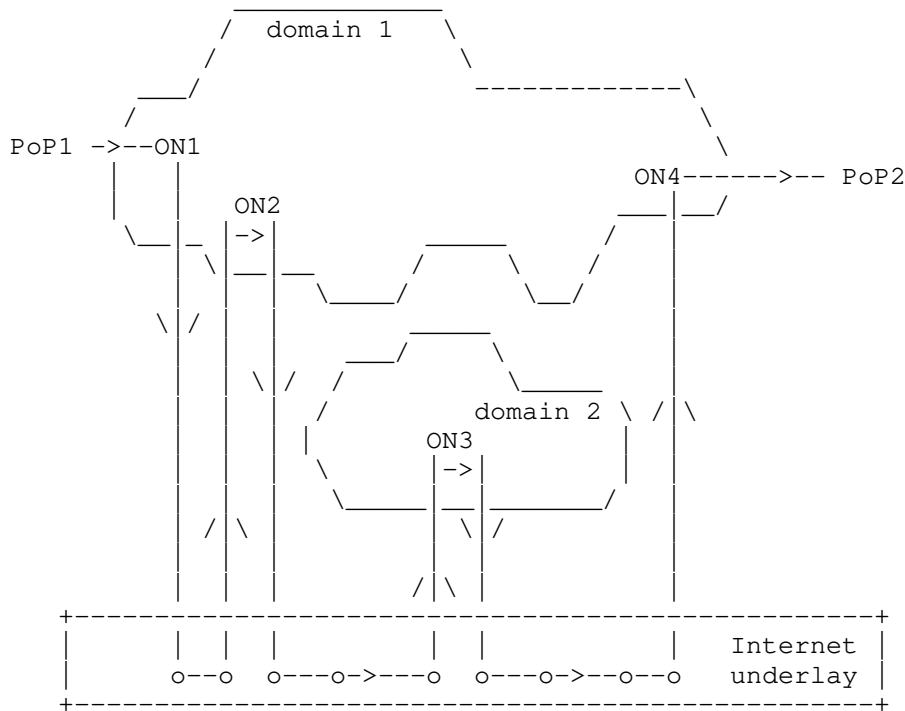


Figure 2: Cloud-Internet Overlay Network (CION)

We tested based on 37 overlay nodes from multiple cloud providers globally. Each pair of the overlay nodes are used as sender and receiver. When the traffic is not intentionally directed to go through any intermediate virtual nodes, we call the path followed by the traffic in the test as the default path. When any of the virtual nodes is intentionally used as an intermediate node to forward the traffic, the path that the traffic takes is called an overlay path. The preliminary experiments showed that the delay of an overlay path is shorter than the one of the default path in 69% of cases at 99% percentile and improvement is 17.5% at 99% percentile when we probe Ping packets every second for a week. More experimental information can be found in [OCN].

Lower delay does not necessarily mean higher throughput. Different path segments may have different packet loss rates. Loss rate is another major factor impacting the overall TCP throughput. From some customer requirements, the target loss rate is set in the test to be less than 1% at 99% percentile and 99.9% percentile, respectively. The loss was measured between any two overlay nodes, i.e., any potential path segment. Two thousand Ping packets were sent every 20

seconds between two overlay nodes for 55 hours. This preliminary experiment showed that the packet loss rate satisfaction are 44.27% and 29.51% at the 99% and 99.9% percentiles, respectively.

Hence packet loss in an overlay segment is a key issue to be solved in such architecture. In long-haul networks, the end-to-end retransmission of lost packet can result in an extra round trip time (RTT). Such extra time is not acceptable in some latency-sensitive applications. As CION naturally consists of multiple overlay segments, LOOPS leverages this to perform local optimizations on a single hop between two overlay nodes. ("Local" here is a concept relative to end-to-end, it does not mean such optimization is limited to LAN networks.)

The following subsections present different scenarios using multiple segment-based overlay paths with a common need of local in-network loss recovery in best effort networks.

### 3.1. Tail Loss or Loss in Short Flows

When the lost segments are at the end of a transaction, TCP's fast retransmit algorithm does not work as there are no ACKs to trigger it. When a sender does not receive an ACK for a given segment within a certain amount of time called retransmission timeout (RTO), it re-sends the segment [RFC6298]. RTO can be as long as several seconds. Hence the recovery of lost segments triggered by RTO is lengthy. [I-D.dukkipati-tcpm-tcp-loss-probe] indicates that large RTOs make a significant contribution to the long tail on the latency statistics of short flows such as loading web pages.

The short flow often completes in one or two RTTs. Even when the loss is not a tail loss, it can possibly add another RTT because of end-to-end retransmission (not enough packets are in flight to trigger fast retransmit). In long-haul networks, it can result in extra time of tens or even hundreds of milliseconds.

An overlay segment transmits the aggregated flows from ON to ON. As short-lived flows are aggregated, the probability of tail loss over this specific overlay segment decreases compared to an individual flow. The overlay segment is much shorter than the end-to-end path in a Cloud- Internet overlay network, hence loss recovery over an overlay segment is faster.

### 3.2. Packet Loss in Real Time Media Streams

The Real-time transport protocol (RTP) is widely used in interactive audio and video. Packet loss degrades the quality of the received media. When the latency tolerance of the application is sufficiently

large, the RTP sender may use RTCP NACK feedback from the receiver [RFC4585] to trigger the retransmission of the lost packets before the playout time is reached at the receiver.

In a Cloud-Internet overlay network, the end-to-end path can be hundreds of milliseconds. End-to-end feedback based retransmission may be not be very useful when applications can not tolerate one more RTT of this length. Loss recovery over an overlay segment can then be used for the scenarios where RTCP NACK triggered retransmission is not appropriate.

### 3.3. Packet Loss and Congestion Control in Bulk Data Transfer

TCP congestion control algorithms such as Reno and CUBIC basically interpret packet loss as congestion experienced somewhere in the path. When a loss is detected, the congestion window will be decreased at the sender to make the sending slower. It has been observed that packet loss is not an accurate way to detect congestion in the current Internet [I-D.cardwell-iccr-g-bbr-congestion-control]. In long-haul links, when the loss is caused by non-persistent burst which is extremely short and pretty random, the sender's reaction of reducing sending rate is not able to respond in time to the instantaneous path situation or to mitigate such bursts. On the contrary, reducing window size at the sender unnecessarily or too aggressively harms the throughput for application's long lasting traffic like bulk data transfer.

The overlay nodes are distributed over the path with computing capability, they are in a better position than the end hosts to quickly deduce the underlying links' instantaneous situation from measuring the delay, loss or other metrics over the segment. Shorter round trip time over a path segment will benefit more accurate and immediate measurements for the maximum recent bandwidth available, the minimum recent latency, or trend of change. ONs can further decide if the sending rate reduction at the sender is necessary when a loss happened. Section 6.2 talks more details on this.

### 3.4. Multipathing

As an overlay path may suffer from an impairment of the underlying network, two or more overlay paths between the same set of ingress and egress overlay nodes can be combined for reliability purpose. During a transient time when a network impairment is detected, sending replicating traffic over two paths can improve reliability.

When two or more disjoint overlay paths are available as shown in Figure 3 from ON1 to ON2, different sets of traffic may use different overlay paths. For instance, one path is for low latency and the



other is for higher bandwidth, or they can be simply used as load balancing for better bandwidth utilization.

Two disjoint paths can be, for example, found by measurement to figure out the segments with very low "mathematical correlation" in latency change. When the number of overlay nodes is large, it is easy to find disjoint or partially disjoint segments. This information may be available if the ONs are managed by the network provider managing the underlying forwarding paths.

Different overlay paths may have varying characteristics, obviously. The overlay tunnel should allow the overlay path to handle the packet loss depending on its own path measurements.

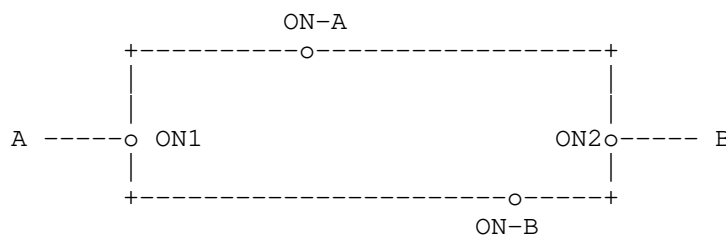


Figure 3: Example of Multiple Overlay Paths

In reference to Figure 3, both A and B are not aware of the existence of these multiple paths. A network-assistance would be valuable for the sake of better resilience and performance. Note that in a collaborative context (a.k.a., stage 2 mentioned in Section 1.2) LOOPS may target means to advertise the available path characteristics to an endpoint A/B, to allow an endpoint A/B to control the traffic distribution policy to be enforced by ON1/ON2, or to let endpoint A/B notify ON1/ON2 with their multipathing preference.

#### 4. Satellite Communication

Traditionally, satellite communications deploy PEP (performance enhancing proxy [RFC3135]) nodes around the satellite link to enhance end-to-end performance. TCP splitting is a common approach employed by such PEPs, where the TCP connection is split into three: the segment before the satellite hop, the satellite section (uplink, downlink), and the segment behind the satellite hop. This requires heavy interactions with the end-to-end transport protocols, usually without the explicit consent of the end hosts. Unfortunately, this is indistinguishable from a man-in-the-middle attack on TCP. With end-to-end encryption moving under the transport (QUIC), this approach is no longer useful.

Geosynchronous Earth Orbit (GEO) satellites have a one-way delay (up to the satellite and back) on the order of 250 milliseconds. This does not include queueing, coding and other delays in the satellite ground equipment. The Round Trip Time for a TCP or QUIC connection going over a satellite hop in both directions, in the best case, will be on the order of 600 milliseconds. And, it may be considerably longer. RTTs on this order of magnitude have significant performance implications.

Packet loss recovery is an area where splitting the TCP connection into different parts helps. Packets lost on the terrestrial links can be recovered at terrestrial latencies. Packet loss on the satellite link can be recovered more quickly by an optimized satellite protocol between the PEPs and/or link layer FEC than they could be end to end. Again, encryption makes TCP splitting no longer applicable. Enhanced error recovery at the satellite link layer helps for the loss on the satellite link but doesn't help for the terrestrial links. Even when the terrestrial segments are short, any loss must be recovered across the satellite link delay. And, there are cases when a satellite ground station connects to the general Internet with a potentially larger terrestrial segment (e.g., to a correspondent host in another country). Faster recovery over such long terrestrial segments is desirable.

Another aspect of recovery is that terrestrial loss is highly likely to be congestion related but satellite loss is more likely to be transmission errors due to link conditions. A transport endpoint slowing down because of mis-interpreting these errors as congestion losses unnecessarily reduces performance. But, at the end points, the difference between the two is not easily distinguished. To elaborate more on the loss recovery for satellite communications, while the error rate on the satellite paths is generally very low most of the time, it might get higher during special link conditions (e.g. fades due to heavy rain). The satellite hop itself does know which losses are due to link conditions as opposed to congestion, but it has no mechanism to signal this difference to the end hosts.

We will need the protocol under QUIC to try to minimize non-congestion packet drop. Specific link layers may have techniques such as satellite FEC to recover. Where the capabilities of that may be exceeded (e.g., rainfade), we can look at LOOPS-like approaches.

There are two high level classes of solutions for making encrypted transport traffic like QUIC work well over satellite:

- o Hooks in the transport protocol which can adapt to large BDPs where both the bandwidth and the latency are large. This would require end to end enhancement.

- o Capabilities (such as LOOPS) under the transport protocol to improve performance over specific segments of the path. In particular, separating the terrestrial from the satellite losses. Fixing the terrestrial loss quickly and keeping throughput high over satellite segment by not causing the end-hosts to over-reduce their sending window in case of non-congestion loss.

This document focuses on the latter.

## 5. Branch Office WAN Connection

Enterprises usually require network connections between the branch offices or between branch offices and cloud data center over geographic distances. With the increasing deployment of vCPE (virtual CPE), some services usually hosted on the CPE are moved to the provider network from the customer site. Such vCPE approach enables some value added service to be provided such as WAN optimization and traffic steering.

Figure 4 shows an example of two branch offices WAN connection via Internet. Figure 5 shows a branch office access to public cloud via a selected PoP (point of presence). vCPE connects to that PoP which can be hundreds of kilometers away via Internet. In both cases, the path segments over Internet is subject to loss. Similar problems presented in subsections of Section 3 should be solved. The GW1 may be reachable via multiple paths.

Requirements to steer traffic through different sub-paths for latency optimization, resource optimization, balancing, or other purposes are increasing. For example, directing the traffic from vCPE to a lightly loaded PoP rather than to the closest one. Mere best effort transport is not sufficient. New technologies like SFC (Service Function Chaining), SRv6 (segment routing over IPv6), and NFV/SDN used together with vCPE to enable the potentials to embed more complicated loss recovery functions at intermediate nodes in end-to-end path.

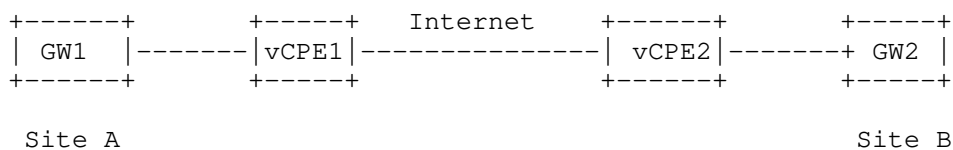


Figure 4: Branch Office WAN Connection via Internet

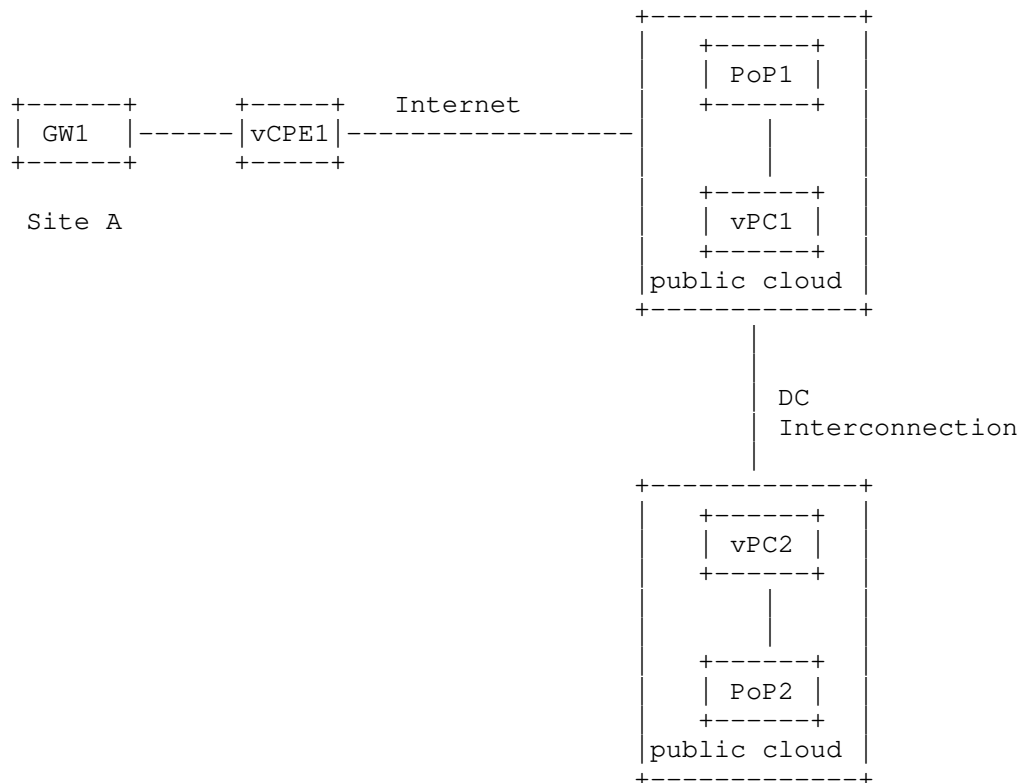


Figure 5: Enterprise Cloud Access

## 6. Features and Impacts to be Considered for LOOPS

This section provides an overview of the proposed LOOPS solution. This section is not meant to document a detailed specification, but it is meant to highlight some design choices that may be followed during the solution design phase.

LOOPS aims to improve the transport performance "locally" in addition to native end-to-end mechanism supported by a given transport protocol. This is possible because LOOPS nodes will be instantiated to partition the path into multiple segments. With the advent of automation and technologies like NFV and virtual IO, it is possible to dynamically instantiate functions to nodes. Some overlay protocols such as VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], LISP [RFC6830] or CAPWAP [RFC5415] may be used in the network. In overlay network usage scenario, LOOPS can extend a specific overlay

protocol header to perform local measurement and local recovery functions, like the example shown in Figure 6.

```

+-----+-----+-----+-----+-----+
|Outer IP hdr|Overlay hdr |LOOPS information|Inner hdr|payload  |
+-----+-----+-----+-----+-----+

```

Figure 6: LOOPS Extension Header Example

LOOPS should be designed to minimize its overhead while increasing the benefit (e.g., reduces the completion time of a video application, reduces the loss). Also, LOOPS should be designed to auto-tune itself in case its overhead is exceeding a threshold.

For example, LOOPS uses packet number space independent from that of the transport layer. Acknowledgment should be generated from ON receiver to ON sender for packet loss detection and local measurement. To reduce overhead, negative ACK over each path segment is a good choice here. A Timestamp echo mechanism, analogous to TCP's Timestamp option, should be employed in-band in LOOPS extension to measure the local RTT and variation for an overlay segment. Local in-network recovery is performed. The measurement over segment is expected to give a hint on whether the lost packet of locally recovered one was caused by congestion. Such a hint could be further feedback, using like by ECN Congestion Experienced (CE) markings, to the end host sender. It directs the end host sender if congestion window adjustment is necessary. LOOPS normally works on the overlay segment which aggregates the same type of traffic, for instance TCP traffic or finer granularity like TCP throughput sensitive traffic. LOOPS does not look into the inner packet (when an encapsulation scheme is used). Elements to be considered in LOOPS are discussed briefly here.

#### 6.1. Local Recovery and End-to-end Retransmission

There are basically two ways to perform local recovery, retransmission and FEC (Forward Error Correction). They are possibly used together in some cases. Such approaches between two overlay nodes recover the lost packet in relatively shorter distance and thus shorter latency. Therefore the local recovery is always faster compared to end-to-end.

At the same time, most transport layer protocols have their own end-to-end retransmission to recover the lost packet. It would be ideal if end-to-end retransmission at the sender was not triggered when the local recovery is successful.

End-to-end retransmission is normally triggered by a NACK as in RTCP or multiple duplicate ACKs as in TCP.

When FEC is used for local recovery, it may come with a buffer to make sure the recovered packets delivered are in order subsequently. Therefore the receiver side is unlikely to see the out-of-order packets and then send a NACK or multiple duplicate ACKs. The side effect to unnecessarily trigger end-to-end retransmit is minimum. When FEC is used, if redundancy and block size are determined, extra latency required to recover lost packets is also bounded. Then RTT variation caused by it is predictable. In some extreme case like a large number of packet loss caused by persistent burst, FEC may not be able to recover it. Then end-to-end retransmit will work as a last resort. In summary, when FEC is used as local recovery, the impact on end-to-end retransmission is limited.

When local retransmission is used, more care is required.

For packet loss in RTP streaming, local retransmission can recover those packets which would not be retransmitted end-to-end otherwise due to long RTT. It would be ideal if the retransmitted packet reaches the receiver before it sends back information that the sender would interpret as a NACK for the lost packet. Therefore when the segment(s) being retransmitted is a small portion of the whole end to end path, the retransmission will have a significant effect of improving the quality at receiver. When the sender also re-transmits the packet based on a NACK received, the receiver will receive the duplicated retransmitted packets and should ignore the duplication.

For packet loss in TCP flows, TCP RENO and CUBIC use duplicate ACKs as a loss signal to trigger the fast retransmit. There are different ways to avoid the sender's end-to-end retransmission being triggered prematurely:

- o The egress overlay node can buffer the out-of-order packets for a while, giving a limited time for a packet being retransmitted somewhere in the overlay path to reach it. The retransmitted packet and the buffered packets caused by it may increase the RTT variation at the sender. When the retransmitted latency is a small portion of RTT or the loss is rare, such RTT variation will be smoothed without much impact. Another possible way is to make the sender exclude such packets from the RTT measurement. The locally recovered packets can be specially marked and this marking is spin back to end host sender. Then RTT measurement should not use that packet.

The buffer management is nontrivial in this case. It has to be determined how many out-of-order packets can be buffered at the

egress overlay node before it gives up waiting for a successful local retransmission. In some extreme case the lost packet is not recovered successfully locally, the sender may invoke end-to-end fast retransmit slower than it would be in classic TCP.

- o If LOOPS network does not buffer the out-of-order packets caused by packet loss, TCP sender can use a time based loss detection like RACK [I-D.ietf-tcpm-rack] to prevent the TCP sender from invoking fast retransmit too early. RACK uses the notion of time to replace the conventional DUPACK threshold approach to detect losses. RACK is required to be tuned to fit the local retransmission better. If there are  $n$  similar segments over the path, segment retransmission will at least add  $RTT/n$  to the reordering window by average when the packet is lost only once over the whole overlay path. This approach is more preferred than one described in previous bullet. On the other hand, if time based loss detection is not supported at the sender, end to end retransmission will be invoked as usual. It wastes some bandwidth.

#### 6.1.1.1. OE to OE Measurement, Recovery, and Multipathing

When multiple segments are stitched, another type of local recovery can be performed between OE (Overlay Edge) to OE. When the segments of an overlay path have similar characteristics and/or only OE has the expected processing capability, OE to OE based local recovery can be used instead of per-segment based recovery.

If there is more than one overlay path between two OEs, multipathing can split and recombine the traffic. Measurements such as RTT and loss rate between OEs have to be specific to each path. The ingress OE can use the feedback measurement to determine the FEC parameter settings for different path. FEC can also be configured to work over the combined path. FEC should not increase redundancy over the path where a congestion is found. The egress OE should be able to remove the duplicated packets when multipathing is available.

OE to OE measurement can help each segment determine its proportion in edge to edge delay. It is useful for ON to decide if it is necessary to turn on the per segment recovery or how to fine tune the parameter settings. When the segment delay ratio is small, the segment retransmission is more effective. Such approach requires nested LOOPS function. This draft does not focus on the nest LOOPS now. More details will be discussed later if comments showing interests in it are received.

## 6.2. Congestion Control Interaction

When a TCP-like transport layer protocol is used, local recovery in LOOPS has to interact with the upper layer transport congestion control. Classic TCP adjusts the congestion window when a loss is detected and fast retransmit is invoked.

The local recovery mechanism breaks the assumption of the necessary and sufficient conditional relationship between detected packet loss and congestion control trigger at the sender in classic TCP. The loss that is locally recovered can be caused by a non-persistent congestion such as a random loss or a microburst, both of which ideally would not let the sender invoke the congestion control mechanism. But then, loss can also possibly caused by a real persistent congestion which should let the sender aware of it and reduces its sending rate.

When a local recovery takes effect, we consider the following two cases. Firstly, the classic TCP sender does not see enough number of duplicate ACKs to trigger fast retransmit. This may be due to the local recovery procedures, which hides the out-of-order packet from receiver using mechanisms like reordering buffer at egress node. Classic TCP sender in this case will not reduce congestion window as no loss is detected. Secondly, if a time based loss detection such as RACK is used, as long as the locally recovered packet's ACK reaches the sender before the reordering window expires, the congestion window will not be reduced.

Such behavior brings the desirable throughput improvement when the recovered packet is lost due to non-persistent congestion. It solves the throughput problem mentioned in Section 3.3 and Section 4. However, it also brings the risk that the sender is not able to detect a real persistent congestion in time, and then overshooting may occur. Eventually a severe congestion that is not recoverable by a local recovery mechanism will be detected by sender. In addition, it may be unfriendly to other flows (possibly pushing them out) if those flows are running over the same underlying bottleneck links.

There is a spectrum of approaches. On one end, each locally recovered packet can be treated exactly as a loss in order to invoke the congestion control at the sender to guarantee the fair sharing as classic TCP by setting its CE (Congestion Experienced) bit. Explicit Congestion Notification (ECN) can be used here as ECN marking was required to be equivalent to a packet drop [RFC3168]. Congestion control at the sender works as usual and no throughput improvement could be achieved (although the benefit of faster recovery is still there). On the other hand, ON can perform its congestion measurement over the segment, for instance local RTT and its variation trend.



Such measurement can help to determine if a lost packet by congestion. It will further decide if it is necessary to set CE marking or even what ratio is set to make the sender adjust the sending rate.

There are possible cases that the sender detects the loss even with local recovery in function. For example, when the re-ordering window in RACK is not optimally adapted, the sender may trigger the congestion control at the same time of end-to-end retransmission. If spurious retransmission detection based on DSACK [RFC3708] is used, such end-to-end retransmission will be found out unnecessary when locally recovered packets reaches the receiver successfully. Then congestion control changes will be undone at the sender. This results in similar pros and cons as described earlier. Pros are preventing the unnecessary window reduction and improving the throughput when the loss is caused by non-congestive loss. Cons are some mechanisms like ECN or its variants should be used wisely to make sure the congestion control is invoked in case of persistent congestion.

An approach where the losses on a path segment are not immediately made known to the end-to-end congestion control can be combined with a "circuit breaker" style congestion control on the path segment. When the usage of path segment by the overlay flow starts to become unfair, the path segment sends congestion signals up to the end-to-end congestion control. This must be carefully tuned to avoid unwanted oscillation.

In summary, local recovery can improve Flow Completion Time (FCT) by eliminating tail loss in small flows. As it may change loss event to out-of-order event in most cases to TCP sender, if TCP sender uses loss based congestion control, there is no much throughput improvement. We suggest ECN and spurious retransmission to be enabled when local recovery is in use, it would give the desirable throughput performance, i.e. when loss is caused by congestion, reduce congestion window; otherwise keep sender's sending rate. We do not suggest to use spurious retransmission alone together with local recovery as it may cause the TCP sender falsely undo window reduction when congestion occurs. If only ECN is enabled or neither ECN nor spurious retransmission is enabled, the throughput with local recovery in use is no much difference from that of the tradition TCP.

### 6.3. Overlay Protocol Extensions

The overlay usually has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for example, the sequence of overlay nodes a message traverses before reaching its destination. LOOPS assumes the overlay protocol can

deliver the packets in such designated sequence. Most forms of overlay networking use some sort of "encapsulation". The whole path taken can be performed by stitching multiple overlay paths, like VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], or it can be a single overlay path with a sequence of intermediate overlay nodes specified, as in SRv6 [I-D.ietf-6man-segment-routing-header]. In either way, LOOPS information is required to be embedded in some form to support the data plane measurement and feedback. Retransmission or FEC based loss recovery can be either per ON-hop or OE to OE based.

LOOPS alone has no setup requirement on control plane. Some overlay protocols, e.g., CAPWAP [RFC5415], has session setup phase, it can be used to exchange the information such as dynamic FEC parameters.

#### 6.4. Summary

LOOPS is expected to extend the existing overlay protocols in data plane. Path selection is assumed a feature provided by the overlay protocols via SDN techniques [RFC7149] or other approaches and is not a part of LOOPS. LOOPS is a set of functions to be implemented on Overlay Nodes, that will be involved in forwarding packets in a long haul overlay network. LOOPS targets the following features.

1. Local recovery: Retransmission, FEC, or combination thereof can be used as local recovery method. Such recovery mechanism is in-network. It is performed by two network nodes with computing and memory resources.
2. Local congestion measurement: Ingress/Egress overlay nodes measure the local segment RTT, loss and/or throughput to immediately get the overlay segment status.
3. Signal to end-to-end congestion control: Strategy to set ECN CE marking or simply not to recover the packet to signal the end host sender about if and/or how to adjust the sending rate is required.

#### 7. Security Considerations

LOOPS does not require access to the traffic payload in clear, so encrypted payload does not affect functionality of LOOPS.

The use of LOOPS introduces some issues which impact security. ON with LOOPS function represents a point in the network where the traffic can be potentially manipulated and intercepted by malicious nodes. Means to ensure that only legitimate nodes are involved should be considered.

Denial of service attack can be launched from an ON. A rogue ON might be able to spoof packets as if it come from a legitimate ON. It may also modify the ECN CE marking in packets to influence the sender's rate. In order to protected from such attacks, the overlay protocol itself should have some build-in security protection which inherently be used by LOOPS. The operator should use some authentication mechanism to make sure ONs are valid and non-compromised.

## 8. IANA Considerations

No IANA action is required.

## 9. Acknowledgements

Thanks to etosat mailing list about the discussion about the SatCom and LOOPS use case.

## 10. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <<https://www.rfc-editor.org/info/rfc3135>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, DOI 10.17487/RFC3708, February 2004, <<https://www.rfc-editor.org/info/rfc3708>>.

- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <<https://www.rfc-editor.org/info/rfc4588>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.

- [I-D.dukkipati-tcpm-tcp-loss-probe]  
Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis,  
"Tail Loss Probe (TLP): An Algorithm for Fast Recovery of  
Tail Losses", draft-dukkipati-tcpm-tcp-loss-probe-01 (work  
in progress), February 2013.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic  
Network Virtualization Encapsulation", draft-ietf-  
nvo3-geneve-13 (work in progress), March 2019.
- [I-D.ietf-tcpm-rack]  
Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK:  
a time-based fast loss detection algorithm for TCP",  
draft-ietf-tcpm-rack-05 (work in progress), April 2019.
- [I-D.ietf-6man-segment-routing-header]  
Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,  
Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment  
Routing Header (SRH)", draft-ietf-6man-segment-routing-  
header-21 (work in progress), June 2019.
- [I-D.cardwell-iccr-g-bbr-congestion-control]  
Cardwell, N., Cheng, Y., Yeganeh, S., and V. Jacobson,  
"BBR Congestion Control", draft-cardwell-iccr-g-bbr-  
congestion-control-00 (work in progress), July 2017.
- [DOI\_10.1109\_ICDCS.2016.49]  
Cai, C., Le, F., Sun, X., Xie, G., Jamjoom, H., and R.  
Campbell, "CRONets: Cloud-Routed Overlay Networks", 2016  
IEEE 36th International Conference on Distributed  
Computing Systems (ICDCS), DOI 10.1109/icdcs.2016.49, June  
2016.
- [DOI\_10.1145\_3038912.3052560]  
Haq, O., Raja, M., and F. Dogar, "Measuring and Improving  
the Reliability of Wide-Area Cloud Paths", Proceedings of  
the 26th International Conference on World Wide Web -  
WWW '17, DOI 10.1145/3038912.3052560, 2017.
- [OCN]  
Xu, Z., Ju, R., Gu, L., Wang, W., Li, J., Li, F., and L.  
Han, "Using Overlay Cloud Network to Accelerate Global  
Communications", INFOCOM ICCN 2019, April 2019,  
<[https://github.com/zhaoguixu/INFOCOM19\\_ICCN/blob/master/  
ocn.pdf](https://github.com/zhaoguixu/INFOCOM19_ICCN/blob/master/ocn.pdf)>.

Authors' Addresses

Yizhou Li  
Huawei Technologies  
101 Software Avenue,  
Nanjing 210012  
China

Phone: +86-25-56624584  
Email: liyizhou@huawei.com

Xingwang Zhou  
Huawei Technologies  
101 Software Avenue,  
Nanjing 210012  
China

Email: zhouxingwang@huawei.com

Mohamed Boucadair  
Orange

Email: mohamed.boucadair@orange.com

Jianglong Wang  
China Telecom

Email: wangjl1.bri@chinatelecom.cn

TSVWG  
Internet-Draft  
Intended status: Informational  
Expires: January 14, 2021

Y. Li  
X. Zhou  
Huawei  
M. Boucadair  
Orange  
J. Wang  
China Telecom  
F. Qin  
China Mobile  
July 13, 2020

LOOPS (Localized Optimizations on Path Segments) Problem Statement and  
Opportunities for Network-Assisted Performance Enhancement  
draft-li-tsvwg-loops-problem-opportunities-06

Abstract

In various network deployments, end to end forwarding paths are partitioned into multiple segments. For example, in some cloud-based WAN communications, stitching multiple overlay tunnels are used for traffic policy enforcement matters such as to optimize traffic distribution or to select paths exposing a lower latency. Likewise, in satellite communications, the communication path is decomposed into two terrestrial segments and a satellite segment. Such long-haul paths are naturally composed of multiple network segments with various encapsulation schemes. Packet loss may show different characteristics on different segments.

Traditional transport protocols (e.g., TCP) respond to packet loss slowly especially in long-haul networks: they either wait for some signal from the receiver to indicate a loss and then retransmit from the sender or rely on sender's timeout which is often quite long. With the increase of end-to-end transport encryption (e.g., QUIC), traditional PEP (performance enhancing proxy) techniques such as TCP splitting are no longer applicable.

LOOPS (Local Optimizations on Path Segments) is a network-assisted performance enhancement over path segment and it aims to provide local in-network recovery to achieve better data delivery by making packet loss recovery faster. In an overlay network scenario, LOOPS can be performed over a variety of the existing, or purposely created, tunnel-based path segments.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. The Problem and Opportunity Overview . . . . .	3
1.2. Sketching a Work Direction: Rationale & Goals . . . . .	4
2. Terminology . . . . .	5
3. Usage Scenarios . . . . .	6
3.1. Cloud-Internet Overlay Network . . . . .	6
3.2. Satellite Communication . . . . .	8
3.3. Branch Office WAN Connection . . . . .	9
4. Impact of Packet loss . . . . .	10
4.1. Tail Loss or Loss in Short Flows . . . . .	10
4.2. Packet Loss in Real Time Media Streams . . . . .	11
5. Features to be Considered for LOOPS . . . . .	11
5.1. Local Recovery . . . . .	11
5.2. Congestion Control Interaction . . . . .	12



5.3. Overlay Protocol Extensions . . . . .	12
6. Local in-network Recovery and End-to-end Retransmission . . .	13
7. Summary . . . . .	14
8. Security Considerations . . . . .	15
9. IANA Considerations . . . . .	15
10. Acknowledgements . . . . .	15
11. Informative References . . . . .	15
Authors' Addresses . . . . .	18

## 1. Introduction

### 1.1. The Problem and Opportunity Overview

Packet loss is ubiquitous in Internet. A reliable transport layer normally employs some end-to-end retransmission mechanisms which also address congestion control [RFC0793] [RFC5681]. The sender either waits for the receiver to send some signals on a packet loss or sets some form of timeout for retransmission. For unreliable transport protocols such as RTP [RFC3550], optional and limited usage of end-to-end retransmission is employed to recover from packet loss [RFC4585] [RFC4588]. End-to-end retransmission to recover lost packets is slow especially when the network is long-haul. For short-lived flows and transactional flows, latency suffers a lot from tail loss.

Tunnels are widely deployed within many networks to achieve various engineering goals, including long-haul WAN interconnection or enterprise wireless access networks. A connection between two endpoints can be decomposed into many connection legs. As such, the corresponding forwarding path can be partitioned into multiple path segments that some of them are using network overlays by means of tunnels. This design serves a number of purposes such as steering the traffic, optimizing egress/ingress link utilization, optimizing traffic performance metrics (such as delay, delay variation, or loss), optimizing resource utilization by invoking resource bonding, provide high-availability, etc.

When a path is partitioned into multiple path segments that are realized typically as overlay tunnels, LOOPS (Local Optimizations on Path Segments) aims to provide in-network recovery over segments to achieve better data delivery by making packet loss recovery faster. In an overlay network scenario, LOOPS can be performed over the existing, or purposely created, overlay tunnel based path segments. Figure 1 show an overall usage scenarios of LOOPS.

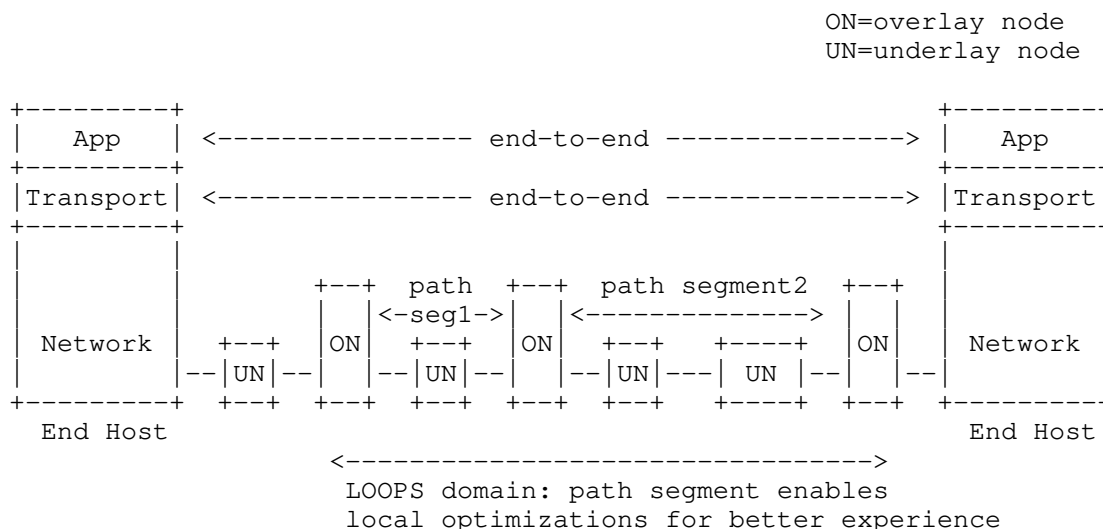


Figure 1: LOOPS Usage Scenario

## 1.2. Sketching a Work Direction: Rationale & Goals

This document sketches a proposal that is meant to experimentally investigate to what extent a network-assisted approach can contribute to increase the overall perceived quality of experience in specific situations (e.g., Sections 3.5 and 3.6 of [RFC8517]) without requiring access to internal transport primitives. The rationale beneath this approach is that some information (loss detection and segment characteristics, etc.) can be used to trigger local in-network recovery actions which have a faster effect while not impacting the end-to-end congestion control loop.

To that aim, the work is structured into two (2) phased stages:

- o Stage 1: Network-assisted optimization. This one assumes that optimizations can be implemented at the network without requiring defining new interaction with the endpoint. Existing tools such as ECN will be used. Loss signal would be converted to CE (congestion experienced) signal to interact with the end-to-end control loop.
- o Stage 2: Collaborative networking optimization. This one requires more interaction between the network and an endpoint to implement coordinated and more surgical network-assisted optimizations based on information/instructions shared by an endpoint or sharing locally-visible information with endpoint for better and faster recovery.

The document focuses on the first stage. Effort related to the second stage is out of scope of the initial planned work.

The proposed mechanism is not meant to be applied to all traffic, but only to a subset which is particularly benefits from, and has been selected for the network-assisted optimization service.

Which traffic is selected is deployment-specific and policy-based. For example, techniques for dynamic information about optimization function (e.g., SFC) may be leveraged to unambiguously identify the aggregate of traffic that is eligible to the service. Such identification may be triggered by subscription actions made by customers or be provided by a network provider (e.g., specific applications, during specific events such as during severe DDoS attack or flash crowds events).

Likewise, whether the optimization function is permanently instantiated or on-demand is deployment-specific.

This document does not intend to provide a comprehensive list of target deployment cases. Sample scenarios are described to illustrate some LOOPS potentials. Similar issues and optimizations may be helpful in other deployments such as enhancing the reliability of data transfer when a fleet of drones are used for specific missions (e.g., site inspection, live streaming, and emergency service). Captured data should be reliably transmitted via paths involving radio connections.

It is not required that all segments are LOOPS-aware to benefit from LOOPS advantages.

Section 3 presents the issues and opportunities found in some multiple path segments scenarios. Section 3 describes the impact of packet loss for different traffic. Section 5 describes the LOOPS desired features and their impact on existing network technologies. Section 6 shows the analysis on local retransmission and end-to-end retransmission. Section 7 summarizes LOOPS key elements.

## 2. Terminology

This document makes use of the following terms:

**LOOPS:** Local Optimizations on Path Segments. LOOPS includes the local in-network (i.e., non end-to-end) recovery functions and other supporting features such as local measurement, loss detection, and congestion feedback.

**LOOPS Node:** A node supporting LOOPS functions.

Overlay Node (ON): A node having overlay functions (e.g., overlay protocol encapsulation/decapsulation, header modification, TLV inspection) and LOOPS functions in the LOOPS overlay network usage scenario.

Overlay Tunnel: A tunnel with designated ingress and egress nodes using some network overlay protocol as encapsulation, optionally with a specific traffic type.

Path segment: A LOOPS enabled tunnel-based network subpath. It is used interchangeably with overlay segment in this document when the context wants to emphasize on its overlay encapsulated nature. It is also called segment for simplicity in this document.

Overlay segment: Refers to path segment.

Underlay Node (UN): A node not participating in the overlay network.

### 3. Usage Scenarios

#### 3.1. Cloud-Internet Overlay Network

CSPs (Cloud Service Providers) are connecting their data centers using the Internet or via self-constructed networks/links. This expands the traditional Internet's infrastructure and, together with the original ISP's infrastructure, forms the Internet underlay.

Automation techniques and NFV (Network Function Virtualization) make it easier to dynamically provision a new virtual node/function as a workload in a cloud for CPU/storage intensive functions. Virtual nodes can be in form of virtual machines or containers hosting the workloads sharing a physical node's infrastructure. With the aid of various mechanisms such as kernel bypassing and Virtual IO, forwarding based on virtual nodes is becoming more and more effective. The interconnection among the purposely positioned virtual nodes and/or the existing nodes with virtualization functions potentially form an overlay infrastructure. It is called the Cloud-Internet Overlay Network (CION) in this document for short.

This architecture scenario makes use of overlay technologies to direct the traffic going through the specific overlay path in order to achieve better service delivery. It purposely creates or selects overlay nodes (ON) from providers. By continuously measuring the delay of path segments and use them as metrics for path selection, when the number of overlay nodes is sufficiently large, there is a high chance that a better path could be found [DOI\_10.1109\_ICDCS.2016.49] [DOI\_10.1145\_3038912.3052560]. [DOI\_10.1145\_3038912.3052560] further shows all cloud providers

experience random loss episodes and random loss accounts for more than 35% of total loss.

Figure 2 shows an example of an overlay path over large geographic distances. An overlay node (ON) is usually a virtual node, though it does not have to be. Three path segments, i.e., ON1-ON2, ON2-ON3, ON3-ON4 are shown. Each segment transmits packets using some form of network overlay protocol encapsulation. ON has the computing and memory resources that can be used for some functions like packet loss detection, network measurement and feedback, packet retransmission and FEC (Forward Error Correction) computation. ONs here are managed by a single administrator though they can be workloads created from different CSPs.

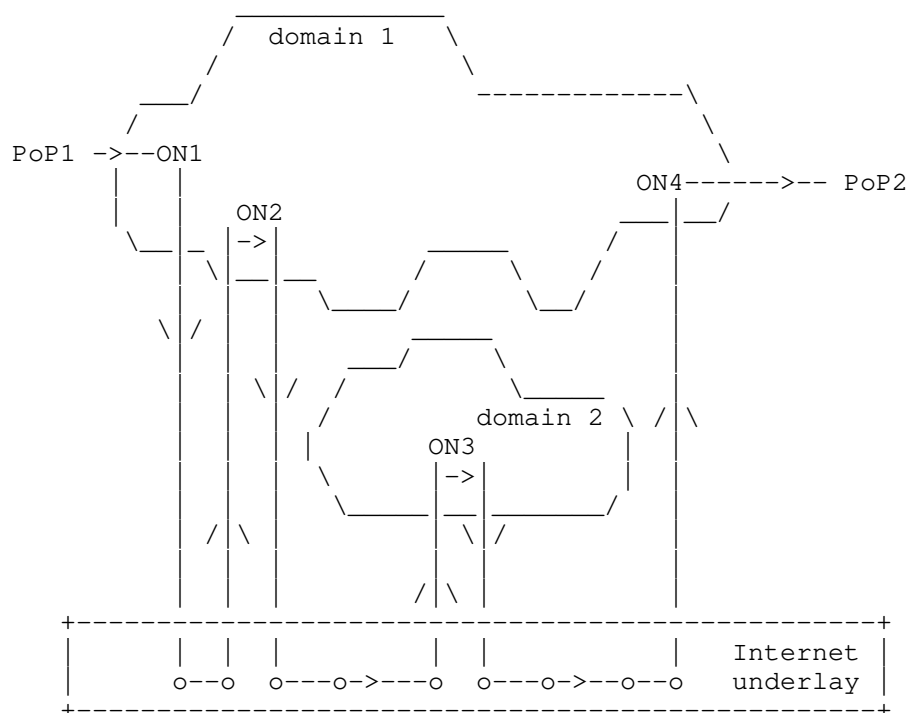


Figure 2: Cloud-Internet Overlay Network (CION)

We tested based on 37 overlay nodes from multiple cloud providers globally. Each pair of the overlay nodes are used as sender and receiver. When the traffic is not intentionally directed to go through any intermediate virtual nodes, we call the path followed by the traffic in the test the default path. When any of the virtual nodes is intentionally used as an intermediate node to forward the

traffic, the path that the traffic takes is called an overlay path. The preliminary experiments showed that the delay of a specifically selected overlay path has lower latency than the one of the default path in 69% of cases at 99% percentile and improvement is 17.5% at 99% percentile when we probe Ping packets every second for a week. The average number of hops for an overlay path is 3.02. More experimental information can be found in [DOI\_10.1109\_INFCOMW.2019.8845208].

Lower average delay does not necessarily mean less or no packet loss. Different path segments have different packet loss rates. Loss rate is another major factor impacting the user experience, especially for the short-lived or transactional flows. From some customer requirements, the target loss rate is set in the test to be less than 1% at 99% percentile and 99.9% percentile, respectively. The loss was measured between any two overlay nodes, i.e., any potential path segment. Two thousand Ping packets were sent every 20 seconds between two overlay nodes for 55 hours. This preliminary experiment showed that the packet loss rate satisfaction are only 44.27% and 29.51% at the 99% and 99.9% percentiles, respectively.

As CION naturally consists of multiple overlay segments, LOOPS can leverage this to perform local optimizations on a single hop between two overlay nodes. ("Local" here is a concept relative to end-to-end, it does not mean such optimization is limited to LAN networks.)

### 3.2. Satellite Communication

Traditionally, satellite communications deploy PEP (performance enhancing proxy [RFC3135]) nodes around the satellite link to enhance end-to-end performance. TCP splitting is a common approach employed by such PEPs, where the TCP connection is split into three: the segment before the satellite hop, the satellite section (uplink, downlink), and the segment behind the satellite hop. This requires heavy interactions with the end-to-end transport protocols, usually without the explicit consent of the end hosts. Unfortunately, this is indistinguishable from a man-in-the-middle attack on TCP. With end-to-end encryption moving under the transport (QUIC), this approach is no longer useful.

Geosynchronous Earth Orbit (GEO) satellites have a one-way delay (up to the satellite and back) on the order of 250 milliseconds. This does not include queueing, coding and other delays in the satellite ground equipment. The Round Trip Time for a TCP or QUIC connection going over a satellite hop in both directions, in the best case, will be on the order of 600 milliseconds. And, it may be considerably longer. RTTs on this order of magnitude have significant performance implications.

Packet loss recovery is an area where splitting the TCP connection into different parts helps. Packets lost on the terrestrial links can be recovered at terrestrial latencies. Packet loss on the satellite link can be recovered more quickly by an optimized satellite protocol between the PEPs and/or link layer FEC than they could be end to end. Again, encryption makes TCP splitting no longer applicable. Enhanced error recovery at the satellite link layer helps for the loss on the satellite link but doesn't help for the terrestrial links. Even when the terrestrial segments are short, any loss must be recovered across the satellite link delay. And, there are cases when a satellite ground station connects to the general Internet with a potentially larger terrestrial segment (e.g., to a correspondent host in another country). Faster recovery over such long terrestrial segments is desirable.

There are two high level classes of solutions for making encrypted transport traffic like QUIC work well over satellite:

- o Hooks in the transport protocol which can adapt to large BDPS where both the bandwidth and the latency are large. This would require end to end enhancement.
- o Capabilities (such as LOOPS) under the transport protocol to improve performance over specific segments of the path. In particular, separating the terrestrial from the satellite losses. Fixing the terrestrial loss quickly.

This document focuses on the latter.

### 3.3. Branch Office WAN Connection

Enterprises usually require network connections between the branch offices, or between branch office and cloud data center over geographic distances. With the increasing deployment of vCPE (virtual CPE), services hosted on the CPE are moved to the provider network from the customer site. Such vCPE approach enables some value added service to be provided such as WAN optimization and traffic steering.

Figure 3 shows a branch office access to public cloud via a selected PoP (point of presence) for service access or reaching another branch office via vPC (Virtual Private Cloud) interconnect. vCPE connects to the PoP which can be hundreds of kilometers away via Internet. From vCPE1 to vCPE2, it can consist of three segments, vCPE1-PoP1, PoP1-PoP2 and PoP2-vCPE2. Packet loss can happen on any of them. Segment based in-network recovery can be employed here to improve the WAN connection quality.

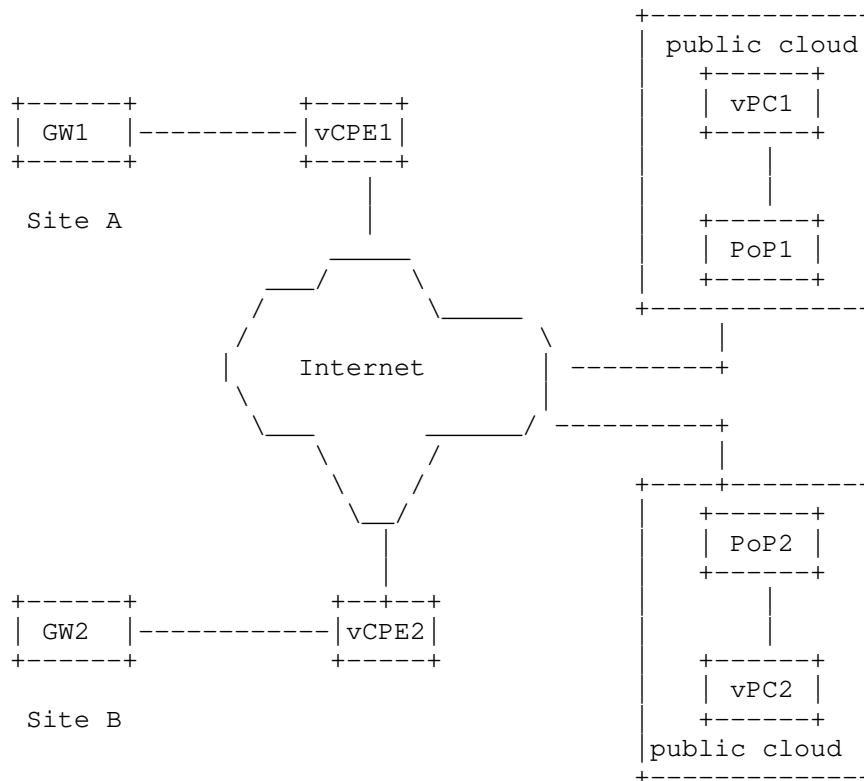


Figure 3: Enterprise Cloud Access

#### 4. Impact of Packet loss

##### 4.1. Tail Loss or Loss in Short Flows

When the lost segments are at the end of a transaction, TCP's fast retransmit algorithm does not work as there are no ACKs to trigger it. When a sender does not receive an ACK for a given segment within a certain amount of time called retransmission timeout (RTO), it re-sends the segment [RFC6298]. RTO can be as long as several seconds. Hence the recovery of lost segments triggered by RTO is lengthy. [I-D.dukkipati-tcpm-tcp-loss-probe] indicates that large RTOs make a significant contribution to the long tail on the latency statistics of short flows such as loading web pages.

The short-lived flows often complete in one or two RTTs. Even when the lost packet is not an exact tail, it can possibly add another RTT



because there may not be enough packets in flight to trigger the fast retransmit). In long-haul networks, it can result in extra time of tens or hundreds of milliseconds. For ant short lived or transactional flows, it affects the latency greatly.

An overlay segment transmits the aggregated flows from ON to ON. As short-lived flows are aggregated, the probability of tail loss over this specific overlay segment decreases compared to an individual flow. The overlay segment is much shorter than the end-to-end path, hence loss recovery over an overlay segment helps to obtain low latency.

#### 4.2. Packet Loss in Real Time Media Streams

The Real-time transport protocol (RTP) is widely used in interactive audio and video. Packet loss degrades the quality of the received media. When the latency tolerance of the application is sufficiently large, the RTP sender may use RTCP NACK feedback from the receiver [RFC4585] to trigger the retransmission of the lost packets before the playout time is reached at the receiver.

The end-to-end path over WAN can be hundreds of milliseconds, so the end-to-end feedback based retransmission may be not be very useful when applications can not tolerate one more RTT. Loss recovery over an overlay segment can then be used for the scenarios in which a shorter delayed retransmission can catch up with the playout time.

### 5. Features to be Considered for LOOPS

This section provides an overview of the LOOPS features. This section is not meant to document a detailed specification, but it is meant to highlight some design choices that may be followed during the solution design phase.

#### 5.1. Local Recovery

LOOPS (Local Optimizations on Path Segments) aims to provide in-network recovery over segments to achieve better data delivery by making packet loss recovery faster. This is viable because LOOPS nodes will be instantiated to partition the path into segments. At the same time, LOOPS does not replace the end-to-end loss recovery (if any). With the advent of automation and technologies like NFV and virtual IO, it is possible to dynamically instantiate functions to nodes. The enabling of LOOPS is expected to be dynamic. When to enable this function is out of scope. The operator or administrator can make the decision based on their historical experience or real-time monitoring.

There are two ways to recover packet, retransmission and Forward Error Correction (FEC). A document to specify the generic elements for loss detection, sequence number space, acknowledgment generation and state transition is available in [I-D.welzl-loops-gen-info].

## 5.2. Congestion Control Interaction

When a TCP-like transport layer protocol is used, local recovery in LOOPS has to interact with the upper layer transport congestion control. Classic TCP adjusts the congestion window when a loss is detected and then fast retransmit is invoked. LOOPS performs in-network recovery which may cause a loss event not being observed by the TCP sender. Then TCP sender may overshoot then.

To solve this issue, LOOPS needs to report the loss to end-to-end congestion control LOOPS. LOOPS can CE (Congestion Experienced) marks its recovered packets as the loss signal to end-to-end. Converting a packet loss signal to CE marking signal brings the benefits of reducing Head-of-Line blocking and probability of RTO expiry [RFC8087] without affecting TCP sender's loss based congestion control behaviour while enjoying the faster local recovery. ECN based indication is equivalent to a loss event at the TCP sender [RFC3168]. In this way, a requirement is set for applying LOOPS. Only ECT (ECN-Capable Transport) flows should be directed to an LOOPS enabled path segment.

## 5.3. Overlay Protocol Extensions

Some tunnel protocols such as VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], LISP [RFC6830] or CAPWAP [RFC5415] are employed in overlay network. They are used in various ways. A path can have single overlay tunnel as a sub-path or stitch multiple segments together, like VXLAN [RFC7348] or GENEVE [I-D.ietf-nvo3-geneve], or specify a sequence of intermediate nodes, as in SRv6 [RFC8754].

LOOPS does not look into the inner packet. LOOPS information is required to be embedded in the overlay protocol header. An example shown in Figure 4. The current protocol focus is GENEVE [I-D.ietf-nvo3-geneve]. The specific information is to be defined in separate documents.

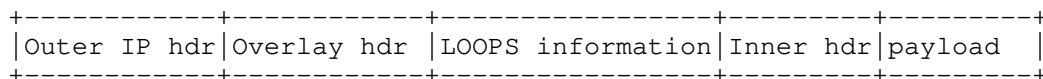


Figure 4: LOOPS Extension Header Example

## 6. Local in-network Recovery and End-to-end Retransmission

Most transport layer protocols have their own end-to-end retransmission to recover the lost packet. When LOOPS is in use, its local recovery can affect the end-to-end one. This section talks about such impacts.

There are two ways to perform local recovery, retransmission and FEC (Forward Error Correction). They are possibly used together in some cases. Such approaches between two overlay nodes recover the lost packet in relatively shorter distance and thus shorter latency. Therefore the local recovery is generally faster compared to end-to-end.

End-to-end retransmission is normally triggered by a NACK as in RTCP, multiple duplicate ACKs as in traditional TCP or time based detection as in RACK [I-D.ietf-tcpm-rack].

When FEC is used for local recovery, it may come with a buffer to make sure the recovered packets delivered are in order subsequently. Therefore the receiver side is unlikely to see the out-of-order packets and then send a NACK or multiple duplicate ACKs. The side effect to unnecessarily trigger end-to-end retransmit is minimum. When FEC is used in this way, if redundancy and block size are determined, extra latency required to recover lost packets is also bounded. Then RTT variation caused by it is predictable. In some extreme case like a large number of packet loss caused by persistent burst, FEC may not be able to recover it. Then end-to-end retransmit will work as a last resort. In summary, when FEC is used as local recovery, the impact on end-to-end retransmission is limited.

When local retransmission is used, it has the following impacts on the end-to-end retransmission.

For packet loss in RTP streaming, local retransmission can recover those packets which would not be retransmitted end-to-end otherwise due to long RTT. Therefore when the segment(s) being retransmitted on is a small portion of the whole end to end path, the retransmission will have a significant effect of improving the quality at receiver.

When the sender also re-transmits the packet based on a NACK received, the receiver may receive the duplicated retransmitted packets.

For packet loss in TCP flows, TCP RENO and CUBIC use duplicate ACKs as a loss signal to trigger the fast retransmit. Though we are not

standardize the buffering feature of a LOOPS egress, an introductory analysis is given as follows.

- o The egress overlay node can buffer the out-of-order packets for a while, giving a limited time for a packet being retransmitted somewhere in the overlay path to reach it. The retransmitted packet and the buffered packets caused by it may increase the RTT variation at the sender. When the retransmitted latency is a small portion of RTT or the loss is rare, such RTT variation will be smoothed without much impact.

The buffer management is nontrivial in this case. It has to be determined how many out-of-order packets can be buffered at the egress overlay node before it gives up waiting for a successful local retransmission. In some extreme case the lost packet is not recovered successfully locally, the sender may invoke end-to-end fast retransmit slower than it would be in classic TCP.

- o If LOOPS network does not buffer the out-of-order packets caused by packet loss, TCP sender which uses a time based loss detection like RACK [I-D.ietf-tcpm-rack] will perform well here. It uses the notion of time to replace the conventional DUPACK threshold approach to detect losses. Hence it prevents the TCP sender from invoking fast retransmit too early. Local retransmission will not interfere the sender's retransmission generally in this case. If time based loss detection is not supported at the sender, end to end retransmission may be invoked as usual. It consumes extra bandwidth Because the lost packets (i.e. recovered packet) is normally a very small percentage of the total packets. Then extra bandwidth cost is not significant.

## 7. Summary

LOOPS will extend the existing overlay protocols in data plane, potential starting from GENEVE [I-D.ietf-nvo3-geneve] which has good extensibility. Path or segment selection can be feature provided by the overlay protocols via SDN techniques [RFC7149] or other approaches and is not a part of LOOPS. LOOPS is a set of functions to be implemented on Overlay Nodes as a tunnel transport with best effort reliability. LOOPS targets the following features.

1. Local recovery: Local recovery: Retransmission, FEC, or combination thereof can be used as local recovery method. Such recovery mechanism is in-network. It is performed by two network nodes with computing and memory resources.

2. Local measurement: Ingress/Egress overlay nodes measure the local segment RTT, loss and/or throughput to immediately get the overlay segment status for loss detection.
3. Interact with end-to-end congestion control: Convert a packet loss signal to an ECN-marking signal to notify the end host sender.

## 8. Security Considerations

LOOPS does not require access to the traffic payload in clear, so encrypted payload does not affect functionality of LOOPS.

The use of LOOPS introduces some issues which impact security. ON with LOOPS function represents a point in the network where the traffic can be potentially manipulated and intercepted by malicious nodes. Means to ensure that only legitimate nodes are involved should be considered.

Denial of service attack can be launched from an ON. A rogue ON might be able to spoof packets as if it come from a legitimate ON. It may also modify the ECN CE marking in packets to influence the sender's rate. In order to protected from such attacks, the overlay protocol itself should have some built-in security protection which is used by LOOPS. The operator should use some authentication mechanism to make sure ONs are valid and non-compromised.

## 9. IANA Considerations

No IANA action is required.

## 10. Acknowledgements

Thanks to etosat mailing list about the discussion about the SatCom and LOOPS use case.

## 11. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <<https://www.rfc-editor.org/info/rfc3135>>.

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <<https://www.rfc-editor.org/info/rfc4588>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.

- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [I-D.dukkipati-tcpm-tcp-loss-probe]  
Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses", draft-dukkipati-tcpm-tcp-loss-probe-01 (work in progress), February 2013.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.
- [I-D.ietf-tcpm-rack]  
Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-08 (work in progress), March 2020.
- [I-D.welzl-loops-gen-info]  
Welzl, M. and C. Bormann, "LOOPS Generic Information Set", draft-welzl-loops-gen-info-03 (work in progress), March 2020.

[DOI\_10.1109\_ICDCS.2016.49]

Cai, C., Le, F., Sun, X., Xie, G., Jamjoom, H., and R. Campbell, "CRONets: Cloud-Routed Overlay Networks", 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), DOI 10.1109/icdcs.2016.49, June 2016.

[DOI\_10.1145\_3038912.3052560]

Haq, O., Raja, M., and F. Dogar, "Measuring and Improving the Reliability of Wide-Area Cloud Paths", Proceedings of the 26th International Conference on World Wide Web, DOI 10.1145/3038912.3052560, April 2017.

[DOI\_10.1109\_INFCOMW.2019.8845208]

Xu, Z., Ju, R., Gu, L., Wang, W., Li, J., Li, F., and L. Han, "Using Overlay Cloud Network to Accelerate Global Communications", IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), DOI 10.1109/infcomw.2019.8845208, April 2019.

#### Authors' Addresses

Yizhou Li  
Huawei Technologies

Email: liyizhou@huawei.com

Xingwang Zhou  
Huawei Technologies

Email: zhouxingwang@huawei.com

Mohamed Boucadair  
Orange

Email: mohamed.boucadair@orange.com

Jianglong Wang  
China Telecom

Email: wangjll.bri@chinatelecom.cn



Fengwei Qin  
China Mobile

Email: qinfengwei@chinamobile.com

TSVWG  
Internet-Draft  
Intended status: Standards Track  
Expires: May 8, 2020

M. Welzl  
University of Oslo  
C. Bormann, Ed.  
Universitaet Bremen TZI  
November 05, 2019

LOOPS Generic Information Set  
draft-welzl-loops-gen-info-02

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery in the presence of losses. [I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol, without already defining a specific data packet format.

The generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. The current version of this document contains sketches of bindings to GUE [I-D.ietf-intarea-gue] and Geneve [I-D.ietf-nvo3-geneve].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 8, 2020.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	5
2. Challenges . . . . .	6
2.1. No Access to End-to-End Transport Information . . . . .	6
2.2. Path Asymmetry . . . . .	6
2.3. Reordering vs. Spurious Retransmission . . . . .	6
2.4. Informing the End-to-End Transport . . . . .	7
2.5. Congestion Detection . . . . .	8
3. Simplifying assumptions . . . . .	8
4. LOOPS Generic Information Set . . . . .	9
4.1. Setup Information . . . . .	9
4.2. Forward Information . . . . .	9
4.3. Reverse Information . . . . .	10
5. LOOPS General Operation . . . . .	11
5.1. Initial Packet Sequence Number . . . . .	11
5.1.1. Minimizing collisions . . . . .	11
5.1.2. Optional Initial PSN procedure . . . . .	12
5.2. Acknowledgement Generation . . . . .	12
5.3. Measurement . . . . .	13
5.3.1. Ingress-relative timestamps . . . . .	13
5.3.2. ACK generation . . . . .	14
5.4. Loss detection and Recovery . . . . .	14
5.4.1. Local Retransmission . . . . .	14
5.4.2. FEC . . . . .	15
5.5. Discussion . . . . .	15
6. Sketches of Bindings to Tunnel Protocols . . . . .	15
6.1. Embedding LOOPS in Geneve . . . . .	16
6.2. Embedding LOOPS in GUE . . . . .	16
7. IANA Considerations . . . . .	16
8. Security Considerations . . . . .	16
8.1. Threat model . . . . .	17

8.2. Discussion . . . . .	18
9. References . . . . .	18
9.1. Normative References . . . . .	18
9.2. Informative References . . . . .	18
Appendix A. Protocol used in Prototype Implementation . . . . .	20
A.1. Block Code FEC . . . . .	21
Appendix B. Transparent mode . . . . .	22
B.1. Packet identification . . . . .	23
B.2. Generic information and protocol operation . . . . .	24
B.3. A hybrid mode . . . . .	24
Acknowledgements . . . . .	26
Authors' Addresses . . . . .	26

## 1. Introduction

Today's networks exhibit a wide variety of data rates and, relative to those, processing power and memory capacities of nodes acting as routers. For instance, networks that employ tunneling to build overlay networks may position powerful virtual router nodes in the network to act as tunnel endpoints. The capabilities available in the more powerful cases provide new opportunities for optimizations.

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery. [I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address. One simplifying assumption (Section 3) in the present document is that LOOPS segments operate independently from each other, each as a pair of a LOOPS Ingress and a LOOPS Egress node.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol between these nodes, without already defining a specific data packet format. The main body of the document defines a mode of the LOOPS protocol that is based on traditional tunneling, the "tunnel mode". Appendix B is an even rougher strawman of a radically different, alternative mode that we call "transparent mode", as well as a slightly more conventional "hybrid mode" (Appendix B.3). These different modes may be applicable to different usage scenarios and will be developed in parallel, with a view of ultimately standardizing one or more of them.

For tunnel mode, the generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. LOOPS is not tied to any specific overlay protocol, but is meant to run embedded into a variety of tunnel protocols. LOOPS information is added as part of a tunnel protocol header at the

LOOPS ingress as shown in Figure 1. The current version of this document contains sketches of bindings to GUE [I-D.ietf-intarea-gue] and Geneve [I-D.ietf-nvo3-geneve].

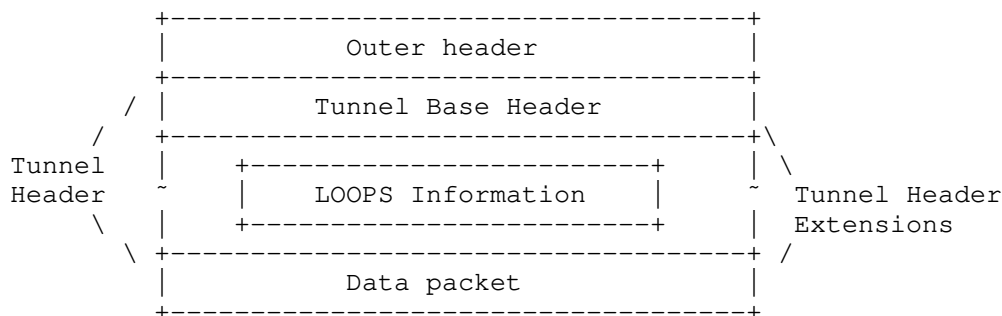


Figure 1: Packet in Tunnel with LOOPS Information

Figure 2 is extracted from the LOOPS problems and opportunities document [I-D.li-tsvwg-loops-problem-opportunities]. It illustrates the basic architecture and terms of the applicable scenario of LOOPS. Not all of the concepts introduced in the problems and opportunities document are actually used in the current strawman specification; Section 3 lays out some simplifying assumptions that the present proposal makes.

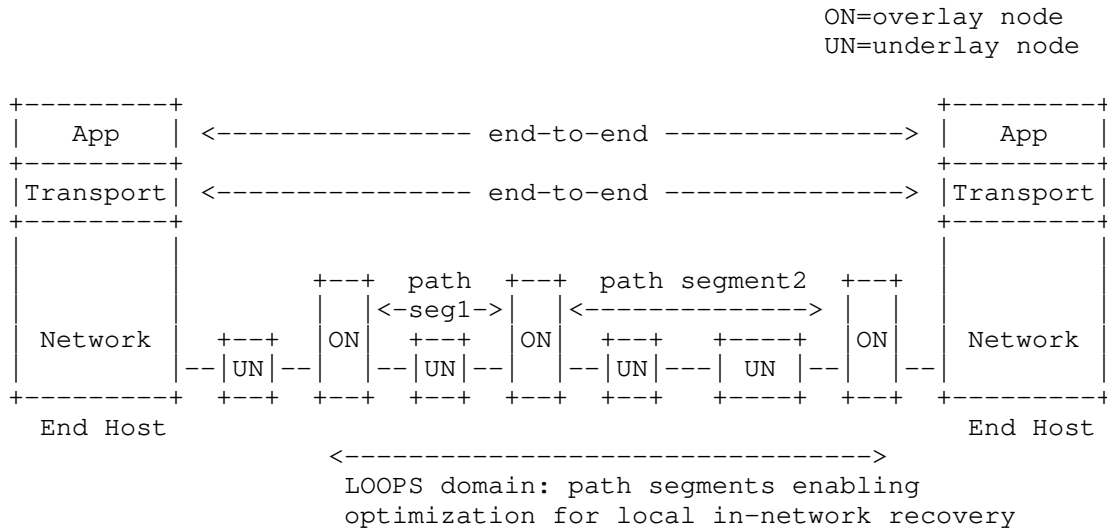


Figure 2: LOOPS Usage Scenario

### 1.1. Terminology

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.

This document makes use of the terminology defined in [I-D.li-tsvwg-loops-problem-opportunities]. This section defines additional terminology used by this document.

**Data packets:** The payload packets that enter and exit a LOOPS segment.

**LOOPS Segment:** A part of an end-to-end path covered by a single instance of the LOOPS protocol, the sub-path between the LOOPS Ingress and the LOOPS Egress.

**LOOPS Ingress:** The node that forwards data packets and forward information into the LOOPS segment, potentially performing retransmission and forward error correction based on acknowledgements and measurements received from the LOOPS Egress.

**LOOPS Egress:** The node that receives the data packets and forward information from the LOOPS ingress, sends acknowledgements and measurements back to the LOOPS ingress (reverse information), potentially recovers data packets from forward error correction information received.

**LOOPS Nodes:** Collective term for LOOPS Ingress and LOOPS Egress in a LOOPS Segment.

**Forward Information:** Information that is added to the stream of data packets in the forward direction by the LOOPS Ingress.

**Reverse Information:** Information that flows in the reverse direction, from the LOOPS Egress back to the LOOPS Ingress.

**Setup Information:** Information that is not transferred as part of the Forward or Reverse Information, but is part of the setup of the LOOPS Nodes.

**PSN:** Packet Sequence Number, a sequence number identifying a data packet.

**Sender:** Original sender of a packet on an end-to-end path that includes one or more LOOPS segment(s).

Receiver: Ultimate receiver of a packet on an end-to-end path that includes one or more LOOPS segment(s).

## 2. Challenges

LOOPS has to perform well in the presence of some challenges, which are discussed in this section.

### 2.1. No Access to End-to-End Transport Information

LOOPS is defined to be independent of the content of the packets being forwarded: there is no dependency on transport-layer or higher information. The intention is to keep LOOPS useful with a traffic mix that may contain encrypted transport protocols such as QUIC as well as encrypted VPN traffic.

### 2.2. Path Asymmetry

A LOOPS segment is defined as a unidirectional forwarding path. The tunnel might be shared with a LOOPS segment in the inverse direction; this then allows to piggyback Reverse Information on encapsulated packets on that segment. But there is no guarantee that the inverse direction of any end-to-end-path crosses that segment, so the LOOPS optimizations have to be useful on their own in each direction.

### 2.3. Reordering vs. Spurious Retransmission

The end-to-end transport layer protocol may have its own retransmission mechanism to recover lost packets. When LOOPS recovers a loss, ideally this local recovery would avoid the triggering of a retransmission at the end-to-end sender.

Whether this is possible depends on the specific end-to-end mechanism used for triggering retransmission. When end-to-end retransmission is triggered by receiving a sequence of duplicate acknowledgements (DUPACKs), and with more than a few packets in flight, the recovered packet is likely to be too late to fill the hole in the sequence number space that triggers the DUPACK detection.

(Given a reasonable setting of parameters, the local retransmission will still arrive earlier than the end-to-end retransmission and will possibly unblock application processing earlier; with spurious retransmission detection, there also will be little long-term effect on the send rate.)

The waste of bandwidth caused by a DUPACK-based end-to-end retransmission can be avoided when the end-to-end loss detection is based on time instead of sequence numbers, e.g., with RACK

[I-D.ietf-tcpm-rack]. This requires a limit on the additional latency that LOOPS will incur in its attempt to recover the loss locally. In the present version of this document, opportunity to set such a limit is provided in the Setup Information. The limit can be used to compute a deadline for retransmission, but also can be used to choose FEC parameters that keep extra latency low.

#### 2.4. Informing the End-to-End Transport

Congestion control at the end-to-end sender is used to adapt its sending rate to the network congestion status. In typical TCP senders, packet loss implies congestion and leads to a reduction in sending rate. With LOOPS operating, packet loss can be masked from the sender as the loss may have been locally recovered. In this case, rate reduction may not be invoked at the sender. This is a desirable performance improvement if the loss was a random loss.

If LOOPS successfully conceals congestion losses from the end-to-end transport protocol, that might increase the rate to a level that congests the LOOPS segment, or that causes excessive queueing at the LOOPS ingress. What LOOPS should be able to achieve is to let the end host sender invoke the rate reduction mechanism when there is a congestion loss no matter if the lost packet was recovered locally.

As with any tunneling protocol, information about congestion events inside the tunnel needs to be exported to the end-to-end path the tunnel is part of. See e.g., [RFC6040] for a discussion of how to do this in the presence of ECN. A more recent draft, [I-D.ietf-tsvwg-tunnel-congestion-feedback], proposes to activate ECN for the tunnel regardless of whether the end-to-end protocol signals the use of an ECN-capable transport (ECT), which requires more complicated action at the tunnel egress.

A sender that interprets reordering as a signal of packet loss (DUPACKs) initiates a retransmission and reduces the sending rate. When spurious retransmission detection (e.g., via F-RTO [RFC5862] or DSACK [RFC3708] is enabled by the TCP sender, it will often be able to undo the unnecessary window reduction. As LOOPS recovers lost packets locally, in most cases the end host sender will eventually find out its reordering-based retransmission (if any) is spurious. This is an appropriate performance improvement if the loss was a random loss. For congestion losses, a congestion event needs to be signaled to the end-to-end transport.

If the end-to-end transport is ECN-capable (which is visible at the IP level), congestion loss events can easily be signaled to them by setting the CE (congestion experienced) mark. If LOOPS detects a congestion loss for a non-ECT packet, it needs to signal a congestion



loss event by introducing a packet loss. This can be done by choosing not to retransmit or repair the packet loss locally in this case. Note that one congestion loss per end-to-end RTT is sufficient to provide the rate reduction, so LOOPS may still be able to recover most packets, in particular for burst losses. (As LOOPS does not interact with the end-to-end transport, it does not know the end-to-end RTT. Some lower bound derived from configuration and measurements could be used instead.)

## 2.5. Congestion Detection

Properly informing the end-to-end transport protocol about congestion loss events requires distinguishing these from random losses. In some special cases, distinguishing information may be available from a link layer (e.g., see Section 3 of [I-D.li-tsvwg-loops-problem-opportunities]). By enabling ECN inside the tunnel, congestion events experienced at ECN-capable routers will usually be identified by the CE mark, which clearly rules out a random loss.

In the general case, the segment may be composed of hops without such special indications. In these cases, some detection mechanism is required to provide this distinguishing information. The specific mechanism used by an implementation is out of scope of LOOPS, but LOOPS will need to provide measurement information for this mechanism. For instance, congestion detection might be based on path segment latency information, the proper measurement of which therefore requires special attention in LOOPS.

## 3. Simplifying assumptions

The above notwithstanding, Implementations may want to make use of indicators such as transport layer port numbers to partition a tunnel flow into separate application flows, e.g., for active queue management (AQM). Any such functionality is orthogonal to the LOOPS protocol itself and thus out of scope for the present document.

One observation that simplifies the design of LOOPS in comparison to that of a reliable transport protocol is that LOOPS does not have to recover every packet loss. Therefore, probabilistic approaches, and simply giving up after some time has elapsed, can simplify the protocol significantly.

For now, we assume that LOOPS segments that may line up on an end-to-end path operate independently of each other. Since the objective of LOOPS ultimately is to assist the end-to-end protocol, it is likely that some cooperation between them would be beneficial, e.g., to obtain some measurements that cover a larger part of the end-to-end

path. For instance, cooperating LOOPS segments could try to divide up permissible increases to end-to-end latency between them. This is out of scope for the present version.

Another simplifying assumption is that LOOPS nodes have reasonably precise absolute time available to them, so there is no need to burden the LOOPS protocol with time synchronization. How this is achieved is out of scope.

LOOPS nodes are created and set up (information about their peers, parameters) by some control plane mechanism that is out of scope for this specification. This means there is no need in the LOOPS protocol itself to manage setup information.

#### 4. LOOPS Generic Information Set

This section sketches a generic information set for the LOOPS protocol. Entries marked with (\*) are items that may not be necessary and probably should be left out of an initial specification.

##### 4.1. Setup Information

Setup Information might include:

- o encapsulation protocol in use, and its vital parameters
- o identity of LOOPS ingress and LOOPS egress; information relevant for running the encapsulation protocol such as port numbers
- o target maximum latency increase caused by the operation of LOOPS on this segment
- o maximum retransmission count (\*)

In the data plane, we have forward information (information added to each data packet) and reverse information. The latter can be sent in separate packets (e.g., Geneve control-only packets [I-D.ietf-nvo3-geneve]) and/or piggybacked like the forward information.

##### 4.2. Forward Information

In the forward information, we have identified:

- o tunnel type (a few bits, meaning agreed between Ingress and Egress)

- o packet sequence number PSN (20+ bits), counting the data packets forwarded transmitted by the LOOPS ingress (i.e., retransmissions re-use the PSN)
- o an "ACK desirable" flag (one bit, usually set for a certain percentage of the data packets only)
- o an optional blob, to be echoed by the egress
- o anything that the FEC scheme needs.

The first four together (say, 3+24+4+1) might even fit into 32 bits, but probably need up to 48 bits total. FEC info of course often needs more space.

(Note that in this proposal there is no timestamp in the forward information; see Section 5.3.)

24 bits of PSN, minus one bit for sequence number arithmetic, gives 8 million packets (or 2.4 GB at typical packetsizes) per worst-case RTT. So if that is, say, 30 seconds, this would be enough to fill 640 Mbit/s.

#### 4.3. Reverse Information

For the reverse information, we have identified:

- o one optional block 1, possibly repeated:
  - \* PSN being acknowledged
  - \* absolute time of reception for the packet acknowledged (PSN)
  - \* the blob, if present, echoed back
- o one optional block 2, possibly repeated:
  - \* an ACK bitmap (based on PSN), always starting at a multiple of 8
  - \* a delta indicating the end PSN of the bitmap (actually the first PSN that is beyond it), using  $(\text{Acked-PSN} \& \sim 7) + 8 * (\text{delta} + 1)$  as the end of the bitmap. Acked-PSN in that formula is the previous block 1 PSN seen in this packet, or 0 if none so far.

Block 1 and Block 2 can be interspersed and repeated. They can be piggybacked on a reverse direction data packet or sent separately if

none occurs within some timeout. They will usually be aggregated in some useful form. Block 1 information sets are only returned for packets that have "ACK desirable" set. Block 2 information is sent by the receiver based on some saturation scheme (e.g., at least three copies for each PSN span over time). Still, it might be possible to go down to 1 or 2 amortized bytes per forward packet spent for all this.

The latency calculation is done by the sender, who occasionally sets "ACK desirable", and notes down the absolute time of transmission for this data packet (the timekeeping can be done quite efficiently as deltas). Upon reception of a block 1 ACK, it can then subtract that from the absolute time of reception indicated. This assumes time synchronization between the nodes is at least as good as the precision of latency measurement needed, which should be no problem with IEEE 1588 PTP synchronization (but could be if using NTP-based synchronization only). A sender can freely garbage collect noted down transmission time information; doing this too early just means that the quality of the RTT sampling will reduce.

## 5. LOOPS General Operation

In the Tunnel Mode described in the main body of this document, LOOPS information is carried by some tunnel encapsulation.

### 5.1. Initial Packet Sequence Number

There is no connection establishment procedure in LOOPS. The initial PSN is assigned unilaterally by the LOOPS Ingress.

Because of the short time that is usually set in the maximum latency increase, there is little damage from a collision of PSNs with packets still in flight from previous instances of LOOPS.

#### 5.1.1. Minimizing collisions

If desired, collisions can be minimized by assigning initial PSNs randomly, or using stable storage. Random assignment is more useful for longer PSNs, where the likelihood of overlap will be low. The specific way a LOOPS ingress uses stable storage is a local matter and thus out of scope. (Implementation note: this can be made to work similar to secure nonce generation with write attenuation: Say, every 10000 packets, the sender notes down the PSN into stable storage. After a reboot, it reloads the PSN and adds 10000 in sequence number arithmetic [RFC1982], plus maybe another 10000 so the sender does not have to wait for the store operation to succeed before sending more packets.)

### 5.1.2. Optional Initial PSN procedure

As a potential option (to be discussed), an initial packet sequence number could be communicated using a simple two-bit protocol, based on an I flag (Initial PSN) carried in the forward information and an R flag (Initial PSN Received) in the reverse information. This procedure essentially clears the egress of any previous state, however, the benefits of this procedure are limited.

The initial PSN is assigned unilaterally by the LOOPS ingress, selected randomly. The ingress will keep setting the I flag to one when it starts to send packets from a new beginning or whenever it believes there is a need to notify the egress about a new initial PSN. The ingress will stop setting the I flag when it receives an acknowledgement with the R flag set from the egress.

When the LOOPS egress receives a packets with the I flag set, it stops performing services that assume a sequential PSN. The egress will no longer provide acknowledgement information for the packets with PSN smaller than this new initial PSN (per sequence number arithmetic [IEN74]). The egress sends acknowledgement information back without any delay by echoing the value of the I flag in the R flag. This also means the egress unsets the R flag in subsequent acknowledgements for packets with the I flag unset.

It may happen that the first few packets are lost in an initial PSN assignment process. In this case, the loss of these packets is not detectable by the LOOPS ingress since the first received PSN will be treated as an initial PSN at the egress. This is an acceptable temporary performance degradation: LOOPS does not intend to provide perfect reliability, and LOOPS usually applies to the aggregated traffic over a tunnel so that the initial PSN assignment happens infrequently.

### 5.2. Acknowledgement Generation

A data packet forwarded by the LOOPS ingress always carries PSN information. The LOOPS egress uses the largest newly received PSN with the "ACK desired" bit as the ACK number in the block 1 part of the acknowledgement. This means that the LOOPS ingress gets to modulate the number of acknowledgement sent by the LOOPS egress. However, whenever an out-of-order packet arrives while there still are "holes" in the PSNs received, the LOOPS receiver should generate a block 2 acknowledgement immediately that the LOOPS sender can use as an ACK list.

Reverse information can be piggybacked in a reverse direction data packet. When the reverse direction has no user data to be sent, a

pure reverse information packet needs to be generated. This may be based on a short delay during which the LOOPS egress waits for a data packet to piggyback on. (To reduce MTU considerations, the egress could wait for less-than-full data packets.)

### 5.3. Measurement

When sending a block 1 acknowledgement, the LOOPS egress indicates the absolute time of reception of the packet. The LOOPS ingress can subtract the absolute time of transmission that it still has available, resulting in one high quality latency sample. (In an alternative design, the forward information could include the absolute time of transmission as well, and block1 information would echo it back. This trades memory management at the ingress for increased bandwidth and MTU reduction.)

When a data packet has been transmitted, it may not be clear which specific copy is acknowledged in a block 1 acknowledgement: the acknowledgement for the initial (or, more generally, an earlier) copy may have been delayed (ACK ambiguity). The LOOPS ingress therefore SHOULD NOT base its measurements on acknowledgements for retransmitted data packets. One way to achieve this is by not setting the "ACK desired" bit on retransmissions in the first place.

The LOOPS ingress can also use the time of reception of the block 1 acknowledgement to obtain a segment RTT sample. Note that this will include any wait time the LOOPS egress incurs while waiting for a piggybacking opportunity -- this is appropriate, as all uses of an RTT will be for keeping a retransmission timeout.

To maintain quality of information during idle times, the LOOPS ingress may send keepalive packets, which are discarded at the LOOPS egress after sending acknowledgements. The indication that a packet is a keepalive packet is dependent on the encapsulation protocol.

#### 5.3.1. Ingress-relative timestamps

As an optional procedure, the ingress node can attach a small blob of data to a forward packet that carries an ACK desired flag; this blob is then echoed by the egress in its block 1 acknowledgement. This is typically used to attach a timestamp on a time scale defined by the ingress; we speak of an ingress-relative timestamp. Alternatively, the ingress can keep a timestamp in its local storage, associated with the PSN of the packet that carries an ACK desired flag; it can then retrieve this timestamp when the block 1 acknowledgement arrives.

In either case, the LOOPS ingress keeps track of the local segment round trip time (LRTT) based on the (saved or received) timestamp and the arrival time of the block 1 acknowledgement, by setting the ACK Desired flag (D flag) occasionally (several times per RTT) and saving/including a sending timestamp for/in the packet.

As the egress will send block 1 acknowledgement information right away when it receives a packet with the D flag set, the measurement of LRTT is more accurate for such packets. A smoothed local segment round trip time S\_LRTT can be computed in a similar way as defined by [RFC0793]. A recent minimum value of LRTT is also kept as min\_LRTT. S\_LRTT is used as a basis for the overall timing of retransmission and state management.

Retransmitted packets MUST NOT be used for local segment round trip time (LRTT) calculation.

#### 5.3.2. ACK generation

A block 1 acknowledgement is generated based on receiving a forward packet with a D flag.

The way block 2 acknowledgement information is sent is more subject to control by the egress. Generally, the egress will aggregate ACK bits for at least K packets before sending a block 2; this can be used to amortize the overhead to close to a couple of bits per ACK. In order to counter loss of reverse information packets, an egress will also want to send an ACK bit more than once -- a saturation value of 3 or more may be chosen based on setup information. Typically, ACK bits already sent will be prepended to ACK bits that are new in this block 2 information set. If K packets do not accumulate for a while, the egress will send one or more packets with block 2 information that covers the unsent ACK bits it has so far.

(Discussion: This works best if the egress has information both about the S\_RTT and min\_RTT that the ingress uses and the reverse packet loss rate.)

#### 5.4. Loss detection and Recovery

There are two ways for LOOPS local recovery, retransmission and FEC.

##### 5.4.1. Local Retransmission

When retransmission is used as recovery mechanism, the LOOPS ingress detects a packet loss by not receiving an ACK for the packet within the time expected based on an RTO value (which might be calculated as

in [RFC6298]). Packet retransmission should then not be performed more than once within an LRTT.

When a retransmission is desired (see Section 2.4 for why it might not be), the LOOPS ingress performs the local in-network recovery by retransmitting the packet. Further retransmissions may be desirable if the lack of ACK is persistent beyond an RTO, as long as the maximum latency increase is not reached.

#### 5.4.2. FEC

FEC is another way to perform local recovery. When FEC is in use, a FEC header is sent with data packets as well as with special repair packets added to the flow. The specific FEC scheme used could be defined in the Setup Information, using a mechanism like [RFC5052]. The FEC rate (amount of redundancy added) and possibly the FEC scheme could be unilaterally adjusted by the LOOPS ingress in an adaptive mechanism based on the measurement information.

#### 5.5. Discussion

Without progress in the way that end-host transport protocols handle reordering, LOOPS will be unable to prevent end-to-end retransmissions that duplicate effort that is spent in local retransmissions. It depends on parameters of the path segment whether this wasted effort is significant or not.

One remedy against this waste could be the introduction of resequencing at the LOOPS Egress node. This increases overall mean packet latency, but does not always increase actual end-to-end data stream latency if a head-of-line blocking transport such as TCP is in use. For applications with a large percentage of legacy TCP end-hosts and sufficient processing capabilities at the LOOPS Egress node, resequencing may be a viable choice. Note that resequencing could be switched off and on depending on some measurement information.

The packet numbering scheme chosen by LOOPS already provides the necessary information for the LOOPS Egress to reconstruct the sequence of data packets at the LOOPS ingress.

#### 6. Sketches of Bindings to Tunnel Protocols

The LOOPS information defined above in a generic way can be mapped to specific tunnel encapsulation protocols. Sketches for two tunnel protocols are given below: Geneve (Section 6.1), and GUE (Section 6.2). The actual encapsulation can be designed in a "native" way by putting each of the various elements into the TLV



format of the encapsulation protocol, or it can be achieved by providing single TLVs for forward and reverse information and using some generic encoding of both kinds of information as shown in Appendix B.3.

#### 6.1. Embedding LOOPS in Geneve

Geneve [I-D.ietf-nvo3-geneve] is an extensible overlay protocol which can embed LOOPS functions. Geneve uses TLVs to carry optional information between NVEs. NVE is logically the same entity as the LOOPS node.

The Geneve header has a mandatory Virtual Network Identifier (VNI) field. The specific VNI value to be used is part of the setup information for the LOOPS tunnel.

More details for a Geneve binding for LOOPS can be found in [I-D.bormann-loops-geneve-binding].

#### 6.2. Embedding LOOPS in GUE

GUE [I-D.ietf-intarea-gue] is an extensible overlay protocol which can embed LOOPS functions. GUE uses flags to indicate the presence of fixed length header extensions. It also allows variable length extensions to be put in "Private data" field. A new LOOPS data block in the "private data" field needs to be defined based on the LOOPS generic information in Section 4.

In the reverse direction, when no data packets are available for piggybacking, LOOPS reverse information is carried in a control message with the C-bit set in the GUE header. The Proto/ctype field contains a control message type when C bit is set. Hence a new control message type should be defined for such LOOPS reverse information.

#### 7. IANA Considerations

No IANA action is required at this stage. When a LOOPS representation is designed for a specific tunneling protocol, new codepoints will be required in the registries that pertain to that protocol.

#### 8. Security Considerations

The security of a specific LOOPS segment will depend both on the properties of the generic information set described here and those of the encapsulation protocol employed. The security considerations of the encapsulation protocol will apply, as will the protection

afforded by any security measures provided by the encapsulation protocol. Any LOOPS encapsulation specification is expected to provide information about preferred configurations of the encapsulation protocol employed, including security mechanisms, and to provide a security considerations section discussing the combination. The following discussion aims at discussing security considerations that will be common between different encapsulations.

### 8.1. Threat model

Attackers might attempt to perturb the operation of a LOOPS segment for a number of purposes:

- o Denial of Service: Damaging the ability of LOOPS to recover packets, or damaging packet forwarding through the LOOPS segment in general.
- o Attacks on Confidentiality or Integrity: Obtaining the content of data packets, modifying them, injecting new or suppressing specific data packets.

For the purposes of these security considerations, we can distinguish three classes of attackers:

1. on-path read-write: The attacker sees packets under way on the segment and can modify, inject, or suppress them.

In this case there is really nothing LOOPS can do, except for acting as a full security protocol on its own, which would be the task of the encapsulation protocol. Without that, attackers already can manipulate the packet stream as they wish. This class of attackers is considered out of scope for these security considerations.

2. on-path read + inject: The attacker sees packets under way on the segment and can inject new packets.

For this case, LOOPS itself similarly cannot add to the confidentiality of the data stream. However, LOOPS could protect against denial of service against its own protocol operation and, in a limited fashion, against attacks on integrity that wouldn't already have been possible by packet injection without LOOPS.

3. off-path inject: The attacker can inject new packets, but cannot see existing packets under way on the segment.

Similar considerations apply as for class 2, except that the "blind" class 3 attacker might need to guess information it could have extracted from the packet stream in class 2.

## 8.2. Discussion

Class 2 attackers can see e.g. sequence numbers and can inject, but not modify traffic. Attacks might include injecting false ACKs, initial PSN flags, ... (TBD)

Class 3 ("blind") attackers might still be able to fake initial PSN bits + false ACKs, but will have a harder time otherwise as it would need to guess the PSN range in which it can wreak havoc. Even random guesses will sometimes hit, though, so the protocol needs to be robust to such injection attacks. ... (TBD)

## 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>>.

### 9.2. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, DOI 10.17487/RFC3708, February 2004, <<https://www.rfc-editor.org/info/rfc3708>>.

- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <<https://www.rfc-editor.org/info/rfc5052>>.
- [RFC5862] Yasukawa, S. and A. Farrel, "Path Computation Clients (PCC) - Path Computation Element (PCE) Requirements for Point-to-Multipoint MPLS-TE", RFC 5862, DOI 10.17487/RFC5862, June 2010, <<https://www.rfc-editor.org/info/rfc5862>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <<https://www.rfc-editor.org/info/rfc6330>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [I-D.ietf-tcpm-rack]  
Cheng, Y., Cardwell, N., Dukkupati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-06 (work in progress), November 2019.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-14 (work in progress), September 2019.
- [I-D.ietf-intarea-gue]  
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-09 (work in progress), October 2019.

- [I-D.li-tsvwg-loops-problem-opportunities]  
Yizhou, L., Zhou, X., Boucadair, M., and J. Wang, "LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement", draft-li-tsvwg-loops-problem-opportunities-03 (work in progress), July 2019.
- [I-D.ietf-tsvwg-tunnel-congestion-feedback]  
Wei, X., Yizhou, L., Boutros, S., and L. Geng, "Tunnel Congestion Feedback", draft-ietf-tsvwg-tunnel-congestion-feedback-07 (work in progress), May 2019.
- [I-D.bormann-loops-geneve-binding]  
"\*\*\* BROKEN REFERENCE \*\*\*".
- [IEN74] Plummer, W., "Sequence Number Arithmetic", Internet Experiment Note 74, September 1978.

#### Appendix A. Protocol used in Prototype Implementation

This appendix describes, in a somewhat abstracted form, the protocol as used in a prototype implementation, as described by Yizhou Li, and Xingwang Zhou.

The prototype protocol can be run in one of two modes (defined by preconfiguration):

- o Retransmission mode
- o Forward Error Correction (FEC) mode

Forward information is piggybacked in data packets.

Reverse information can be carried in a pure acknowledgement packet or piggybacked when carrying packets for the inverse direction.

The forward information includes:

- o Packet Sequence Number (PSN) (32 bits): This identifies a packet over a specific overlay segment from a specific LOOPS Ingress. If a packet is retransmitted by LOOPS, the retransmission uses the original PSN.
- o Timestamp (32 bits): Information, in a format local to the LOOPS ingress, that provides the time when the packet was sent. In the current implementation, a 32-bit unsigned value specifying the time delta in some granularity from the epoch time to the sending time of the packet carrying this timestamp. The granularity can

be from 1 ms to 1 second. The epoch time follows the current TCP practice which is 1 January 1970 00:00:00 UTC. Note that a retransmitted packet uses its own Timestamp.

- o FEC Info for Block Code (56 bits): This header is used in FEC mode. It currently only provides for a block code FEC scheme. It includes the Source Block Number (SBN), Encoding Symbol ID (ESI), number of symbols in a single source block and symbol size. Appendix A.1 gives more details on FEC.

The reverse information includes:

- o ACK Number (32 bits): The largest (in sequence number arithmetic [RFC1982]) PSN received so far.
- o ACK List (variable): This indicates an array of PSN numbers to describe the PSN "holes" preceding the ACK number. It conceptually lists the PSNs of every packet perceived as lost by the LOOPS egress. In actual use, it is truncated.
- o Echoed Timestamp (32 bits): The timestamp received with the packet being acknowledged.

#### A.1. Block Code FEC

The prototype currently uses a block code FEC scheme (RaptorQ [RFC6330]). The fields in the FEC Info forward information are:

- o Source Block Number (SBN): 16 bits. An integer identifier for the source block that the encoding symbols within the packet relate to.
- o Encoding Symbol ID (ESI): 16 bits. An integer identifier for the encoding symbols within the packet.
- o K: 8 bits. Number of symbols in a single source block.
- o T: 16 bits. Symbol size in bytes.

The LOOPS Ingress uses the data packet in Figure 1 to generate the encoding packet. Both source packets and repair packets carry the FEC header information; the LOOPS Egress reconstructs the data packets from both kinds of packets. The LOOPS Egress currently resequences the forwarded and reconstructed packets, so they are passed on in-order when the lost packets are recoverable within the source block.

The LOOPS Nodes need to agree on the use of FEC block mode and on the specific FEC Encoding ID to use; this is currently done by configuration.

## Appendix B. Transparent mode

This appendix defines a very different way to provide the LOOPS services, "transparent mode". (We call the protocol described in the main body of the document "encapsulated mode".)

In transparent mode, the idea is that LOOPS does not meddle with the forward transmission of data packets, but runs on the side exchanging additional information.

An implementation could be based on conventional forwarding switches that just provide a copy of the ingress and egress packet stream to the LOOPS implementations. The LOOPS process would occasionally inject recovered packets back into the LOOPS egress node's forwarding switch, see Figure 3.

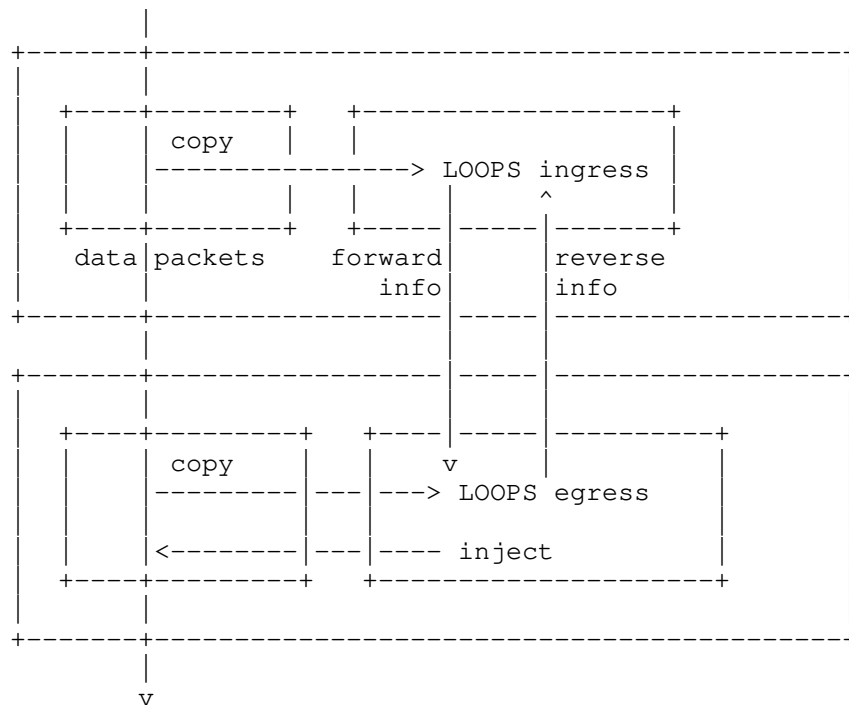


Figure 3: LOOPS Transparent Mode

The obvious advantage of transparent mode is that no encapsulation is needed, reducing processing requirements and keeping the MTU unchanged. The obvious disadvantage is that no forward information can be provided with each data packet, so a replacement needs to be found for the PSN (packet sequence number) employed in encapsulated mode. Any forward information beyond the data packets is sent in separate packets exchanged directly between the LOOPS nodes.

#### B.1. Packet identification

Retransmission mode and FEC mode differ in their needs for packet identification. For retransmission mode, a somewhat probabilistic accuracy of the packet identification is sufficient, for FEC mode, packet identification should not make mistakes (as these would lead to faultily reconstructed packets).

In Retransmission mode, misidentification of a packet could lead to measurement errors as well as missed retransmission opportunities. The latter will be fixed end-to-end. The tolerance for measurement errors would influence the degree of accuracy that is aimed for.

Packet identification can be based on a cryptographic hash of the packet, computed in LOOPS ingress and egress using the same algorithm (excluding fields that can change in transit, such as TTL/hop limit). The hash can directly be used as a packet number, or it can be sent in the forward information together with a packet sequence number, establishing a mapping.

For probabilistic packet identification, it is almost always sufficient to hash the first few (say, 64) bytes of the packet; all known transport protocols keep sufficient identifying information in that part (and, for encrypted protocols, the entropy will be sufficient). Any collisions of the hash could be used to disqualify the packet for measurement purposes, minimizing the measurement errors; this could allow rather short packet identifiers in retransmission mode.

For FEC mode, the packet identification together with the per-packet FEC information needs to be sent in the (separate) forward information, so that a systematic code can be reconstructed. For retransmission mode, there is no need to send any forward information for most packets, or a mapping from packet identifiers to packet sequence numbers could be sent in the forward information (probably in some aggregated form). The latter would allow keeping the acknowledgement form described in the main body (with aggregate acknowledgement); otherwise, packet identifiers need to be acknowledged. With this change, the LOOPS egress will send reverse information as in the encapsulating LOOPS protocol.



## B.2. Generic information and protocol operation

With the changes outlined above, transparent mode operates just as encapsulated mode. If packet sequence numbers are not used, there is no use for block2 reverse information; if they are used, a new block3 needs to be defined that provides the mapping from packet identifiers to packet sequence numbers in the forward information. To avoid MTU reduction, some mechanism will be needed to encapsulate the actual FEC information (additional packets) in the forward information.

## B.3. A hybrid mode

Figure 3 can be modified by including a GRE encapsulator into the top left corner and a GRE decapsulator in the bottom left corner. This provides more defined ingress and egress points, but it also provides an opportunity to add a packet sequence number at the ingress. The copies to the top right and bottom right corners are the encapsulated form, i.e., include the sequence number.

The GRE packet header then has the form:

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|0|0|0|1|          000000000          | 000 |          Protocol Type          |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Sequence Number                             |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

The forward and reverse information can be designed closer to the approach in the main body of the document, to be exchanged using UDP packets between top right ingress and bottom right egress using a port number allocated for this purpose.

Rough ideas for both directions are given below in CDDL [RFC8610]. This information set could be encoded in CBOR or in a bespoke encoding; details such as this can be defined later.

```
forward-information = [  
  [rel-psn, ack-desired, ? fec-info] /  
  fec-repair-data  
]  
  
rel-psn = uint; relative packet sequence number  
; always given as a delta from the previous one in the array  
; starting out with a "previous value" of 0  
  
ack-desired = bool  
  
fec-info = [  
  sbn: uint, ; Source Block Number  
  esi: uint, ; Encoding Symbol ID  
  ? (  
    nsssb: uint; number of symbols in a single source block  
    ss: uint; symbol size  
  )  
]  
  
fec-repair-data = [  
  repair-data: bytes  
  ? (  
    sbn: uint, ; Source Block Number  
    esi: uint, ; Encoding Symbol ID  
  )  
]
```

If left out for a sequence number, the fec-info block is constructed by adding one to the previous one. fec-repair-data contain repair symbols for the sbn/esi given (which, again, are reconstructed from context if not given).

```
reverse-information = [  
  block1 / block2  
]  
  
block1 = [rel-psn, timestamp]  
block2 = [end-psn-delta: uint, acked-bits: bytes]
```

The acked-bits in a block2 is a bitmap that gives acknowledgments for received data packets. The bitmap always comes as a multiple of 8 bits (all bytes are filled in with 8 bits, each identifying a PSN). The end PSN of the bitmap (actually the first PSN that would be beyond it) is computed from the current PSN as set by rel-psn, rounded down to a multiple of 8, and adding  $8 \cdot (\text{end-psn-delta} + 1)$  to that value.

#### Acknowledgements

Sami Boutros helped with sketching the use of Geneve (Section 6.1), and Tom Herbert helped with sketching the use of GUE (Section 6.2).

Michael Welzl has been supported by the Research Council of Norway under its "Toppforsk" programme through the "OCARINA" project.

#### Authors' Addresses

Michael Welzl  
University of Oslo  
PO Box 1080 Blindern  
Oslo N-0316  
Norway  
  
Phone: +47 22 85 24 20  
Email: [michawe@ifi.uio.no](mailto:michawe@ifi.uio.no)

Carsten Bormann (editor)  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany  
  
Phone: +49-421-218-63921  
Email: [cabo@tzi.org](mailto:cabo@tzi.org)

TSVWG  
Internet-Draft  
Intended status: Standards Track  
Expires: 14 January 2021

M. Welzl  
University of Oslo  
C. Bormann, Ed.  
Universität Bremen TZI  
13 July 2020

LOOPS Generic Information Set  
draft-welzl-loops-gen-info-04

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery in the presence of losses.

[I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol, without already defining a specific data packet format.

The generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. A companion document contains a sketch of a binding to the tunnel encapsulation protocol Geneve [I-D.ietf-nvo3-geneve].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 January 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	5
2. Challenges . . . . .	6
2.1. No Access to End-to-End Transport Information . . . . .	6
2.2. Path Asymmetry . . . . .	6
2.3. Reordering vs. Spurious Retransmission . . . . .	6
2.4. Informing the End-to-End Transport . . . . .	7
3. Simplifying assumptions . . . . .	8
4. LOOPS Architecture . . . . .	9
5. LOOPS Generic Information Set . . . . .	10
5.1. Setup Information . . . . .	10
5.2. Forward Information . . . . .	10
5.3. Reverse Information . . . . .	11
6. LOOPS General Operation . . . . .	12
6.1. Initial Packet Sequence Number . . . . .	12
6.1.1. Minimizing collisions . . . . .	12
6.1.2. Optional Initial PSN procedure . . . . .	12
6.2. Acknowledgement Generation . . . . .	13
6.3. Measurement . . . . .	14
6.3.1. Ingress-relative timestamps . . . . .	14
6.3.2. ACK generation . . . . .	15
6.4. Loss detection and Recovery . . . . .	15
6.4.1. Local Retransmission . . . . .	15
6.4.2. FEC . . . . .	16
6.5. Discussion . . . . .	16
7. Sketches of Bindings to Tunnel Protocols . . . . .	16
7.1. Embedding LOOPS in Geneve . . . . .	17
8. IANA Considerations . . . . .	17
9. Security Considerations . . . . .	17
9.1. Threat model . . . . .	17
9.2. Discussion . . . . .	18
10. References . . . . .	18
10.1. Normative References . . . . .	18
10.2. Informative References . . . . .	19
Appendix A. Protocol used in Prototype Implementation . . . . .	21
A.1. Block Code FEC . . . . .	22
Appendix B. Transparent mode . . . . .	22

B.1. Packet identification . . . . .	24
B.2. Generic information and protocol operation . . . . .	25
B.3. A hybrid mode . . . . .	25
Acknowledgements . . . . .	27
Authors' Addresses . . . . .	27

## 1. Introduction

Today's networks exhibit a wide variety of data rates and, relative to those, processing power and memory capacities of nodes acting as routers. For instance, networks that employ tunneling to build overlay networks may position powerful virtual router nodes in the network to act as tunnel endpoints. The capabilities available in the more powerful cases provide new opportunities for optimizations.

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery. [I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address. One simplifying assumption (Section 3) in the present document is that LOOPS segments operate independently from each other, each as a pair of a LOOPS Ingress and a LOOPS Egress node.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol between these nodes, without already defining a specific data packet format. The main body of the document defines a mode of the LOOPS protocol that is based on traditional tunneling, the "tunnel mode". Appendix B is an even rougher strawman of a radically different, alternative mode that we call "transparent mode", as well as a slightly more conventional "hybrid mode" (Appendix B.3). These different modes may be applicable to different usage scenarios and will be developed in parallel, with a view of ultimately standardizing one or more of them.

For tunnel mode, the generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. LOOPS is not tied to any specific overlay protocol, but is meant to run embedded into a variety of tunnel protocols. LOOPS information is added as part of a tunnel protocol header at the LOOPS ingress as shown in Figure 1. A companion document [I-D.bormann-loops-geneve-binding] contains a sketch of a binding to the tunnel encapsulation protocol Geneve [I-D.ietf-nvo3-geneve].

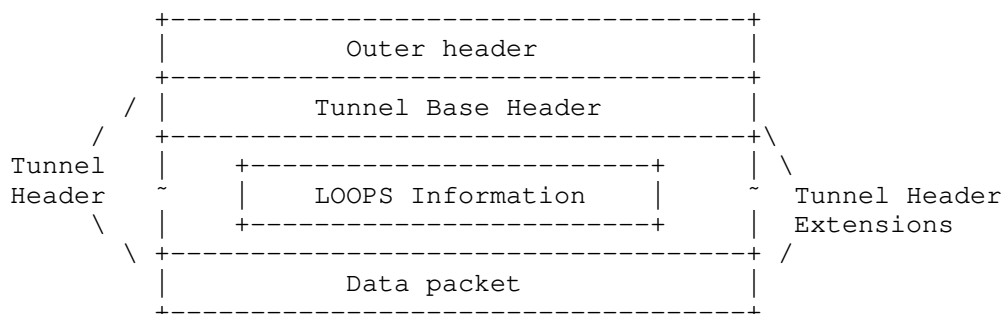


Figure 1: Packet in Tunnel with LOOPS Information

Figure 2 is extracted from the LOOPS problems and opportunities document [I-D.li-tsvwg-loops-problem-opportunities]. It illustrates the basic architecture and terms of the applicable scenario of LOOPS. Not all of the concepts introduced in the problems and opportunities document are actually used in the current strawman specification; Section 3 lays out some simplifying assumptions that the present proposal makes.

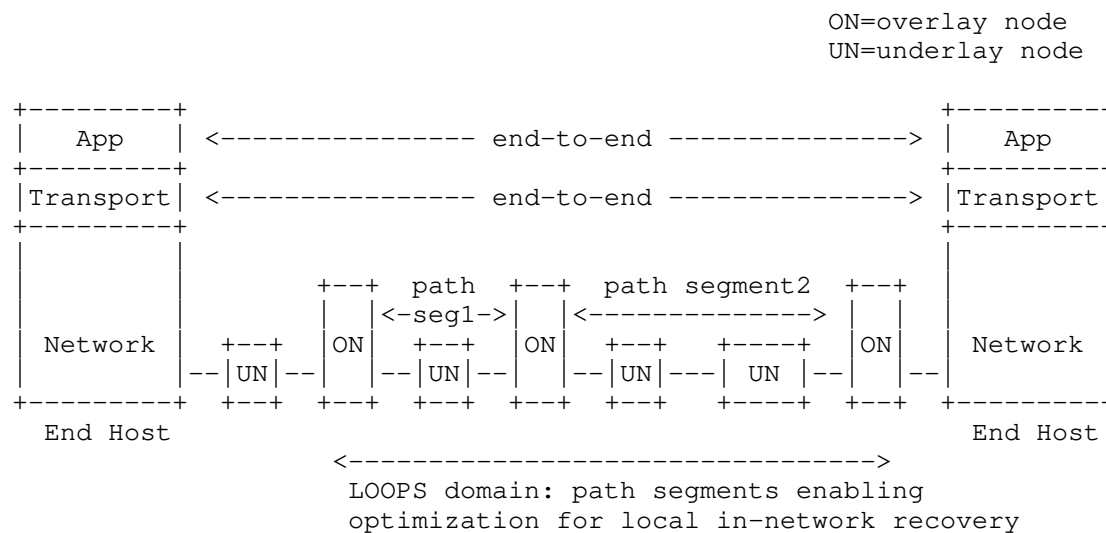


Figure 2: LOOPS Usage Scenario

## 1.1. Terminology

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.

This document makes use of the terminology defined in [I-D.li-tsvwg-loops-problem-opportunities]. This section defines additional terminology used by this document.

**Data packets:** The payload packets that enter and exit a LOOPS segment.

**LOOPS Segment:** A part of an end-to-end path covered by a single instance of the LOOPS protocol, the sub-path between the LOOPS Ingress and the LOOPS Egress. Several LOOPS segments may be encountered on an end-to-end path, with or without intervening routers.

**LOOPS Ingress:** The node that forwards data packets and forward information into the LOOPS segment, potentially performing retransmission and forward error correction based on acknowledgements and measurements received from the LOOPS Egress.

**LOOPS Egress:** The node that receives the data packets and forward information from the LOOPS ingress, sends acknowledgements and measurements back to the LOOPS ingress (reverse information), potentially recovers data packets from forward error correction information received.

**LOOPS Nodes:** Collective term for LOOPS Ingress and LOOPS Egress in a LOOPS Segment.

**Forward Information:** Information that is added to the stream of data packets in the forward direction by the LOOPS Ingress.

**Reverse Information:** Information that flows in the reverse direction, from the LOOPS Egress back to the LOOPS Ingress.

**Setup Information:** Information that is not transferred as part of the Forward or Reverse Information, but is part of the setup of the LOOPS Nodes.

**PSN:** Packet Sequence Number, a sequence number identifying a data packet between the LOOPS Ingress and Egress.



Sender: Original sender of a packet on an end-to-end path that includes one or more LOOPS segment(s).

Receiver: Ultimate receiver of a packet on an end-to-end path that includes one or more LOOPS segment(s).

## 2. Challenges

LOOPS has to perform well in the presence of some challenges, which are discussed in this section.

### 2.1. No Access to End-to-End Transport Information

LOOPS is defined to be independent of the content of the packets being forwarded: there is no dependency on transport-layer or higher information. The intention is to keep LOOPS useful with a traffic mix that may contain encrypted transport protocols such as QUIC as well as encrypted VPN traffic.

### 2.2. Path Asymmetry

A LOOPS segment is defined as a unidirectional forwarding path. The tunnel might be shared with a LOOPS segment in the inverse direction; this then allows to piggyback Reverse Information on encapsulated packets on that segment. But there is no guarantee that the inverse direction of any end-to-end-path crosses that segment, so the LOOPS optimizations have to be useful on their own in each direction.

### 2.3. Reordering vs. Spurious Retransmission

The end-to-end transport layer protocol may have its own retransmission mechanism to recover lost packets. When LOOPS recovers a loss, ideally this local recovery would replace the triggering of a retransmission at the end-to-end sender.

Whether this is possible depends on the specific end-to-end mechanism used for triggering retransmission. When end-to-end retransmission is triggered by receiving a sequence of duplicate acknowledgements (DUPACKs), and with more than a few packets in flight, the recovered packet is likely to be too late to fill the hole in the sequence number space that triggers the DUPACK detection.

(Given a reasonable setting of parameters, the local retransmission will still arrive earlier than the end-to-end retransmission and will possibly unblock application processing earlier; with spurious retransmission detection, there also will be little long-term effect on the send rate.)

While LOOPS makes no requirements on end-to-end protocols, it is worth noting that the waste of bandwidth caused by a DUPACK-based end-to-end retransmission can be avoided when the end-to-end loss detection is based on time instead of sequence numbers, e.g., with RACK [I-D.ietf-tcpm-rack]. This requires a limit on the additional latency that LOOPS will incur in its attempt to recover the loss locally. In the present version of this document, opportunity to set such a limit is provided in the Setup Information. The limit can be used to compute a deadline for retransmission, but also can be used to choose FEC parameters that keep extra latency low.

#### 2.4. Informing the End-to-End Transport

Congestion control at the end-to-end sender is used to adapt its sending rate to the network congestion status. In typical TCP senders, packet loss implies congestion and leads to a reduction in sending rate. With LOOPS operating, packet loss can be masked from the sender as the loss may have been locally recovered. In this case, rate reduction may not be invoked at the sender. This is a desirable performance improvement if the loss was a random loss, but it is hard to ascertain that.

If LOOPS successfully conceals congestion losses from the end-to-end transport protocol, that might increase the rate to a level that congests the LOOPS segment, or that causes excessive queueing at the LOOPS ingress. What LOOPS should be able to achieve is to let the end host sender invoke the rate reduction mechanism when there is a congestion loss no matter if the lost packet was recovered locally.

As with any tunneling protocol, information about congestion events inside the tunnel needs to be exported to the end-to-end path the tunnel is part of. See e.g., [RFC6040] for a discussion of how to do this in the presence of ECN. A more recent draft, [I-D.ietf-tsvwg-tunnel-congestion-feedback], proposes to activate ECN for the tunnel regardless of whether the end-to-end protocol signals the use of an ECN-capable transport (ECT), which requires more complicated action at the tunnel egress.

A sender that interprets reordering as a signal of packet loss (DUPACKs) initiates a retransmission and reduces the sending rate. When spurious retransmission detection (e.g., via F-RTO [RFC5862] or DSACK [RFC3708]) is enabled by the TCP sender, it will often be able undo the unnecessary window reduction shortly afterwards. As LOOPS recovers lost packets locally, in most cases the end host sender will eventually find out its reordering-based retransmission (if any) is spurious. This is an appropriate performance improvement if the loss was a random loss. For congestion losses, a congestion event needs to be signaled to the end-to-end transport.

The present version of LOOPS requires the end-to-end transport to be ECN-capable (which is visible at the IP level). Congestion loss events can easily be signaled to them by setting the CE (congestion experienced) mark. Effectively, LOOPS converts a packet loss (which would be a congestion indication) to a CE mark (which also is a congestion indication).

In effect, LOOPS can be used to convert a path segment that does not yet use CE marks for congestion indication, and drops packets instead, into a segment that marks for congestion and does not drop packets except in extreme cases, incurring the benefits of Using Explicit Congestion Notification (ECN) [RFC8087]. We speak about the "drop-to-mark" function of LOOPS.

### 3. Simplifying assumptions

The above notwithstanding, Implementations may want to make use of indicators such as transport layer port numbers to partition a tunnel flow into separate application flows, e.g., for active queue management (AQM). Any such functionality is orthogonal to the LOOPS protocol itself and thus out of scope for the present document.

One observation that simplifies the design of LOOPS in comparison to that of a reliable transport protocol is that LOOPS does not have to recover every packet loss. Therefore, probabilistic approaches, and simply giving up after some time has elapsed, can simplify the protocol significantly.

For now, we assume that LOOPS segments that may line up on an end-to-end path operate independently of each other. Since the objective of LOOPS ultimately is to assist the end-to-end protocol, it is likely that some cooperation between them would be beneficial, e.g., to obtain some measurements that cover a larger part of the end-to-end path. For instance, cooperating LOOPS segments could try to divide up permissible increases to end-to-end latency between them. This is out of scope for the present version.

Another simplifying assumption is that LOOPS nodes have reasonably precise absolute time available to them, so there is no need to burden the LOOPS protocol with time synchronization. How this is achieved is out of scope.

LOOPS nodes are created and set up (information about their peers, parameters) by some control plane mechanism that is out of scope for this specification. This means there is no need in the LOOPS protocol itself to manage setup information.

#### 4. LOOPS Architecture

From the above, the following architecture is derived for LOOPS.

LOOPS governs the segment from an ingress node to an egress node, which is part of one or more end-to-end paths. Often, a LOOPS segment will operate on aggregate traffic from many such end-to-end paths.

The LOOPS protocol itself does not define how a LOOPS segment and the protocol entities in the ingress and egress node are set up. We expect that a `_setup protocol_` on the control plane will provide some `_setup information_` to the two nodes, including when to start and to tear down processing.

Each LOOPS segment governs traffic on one direction in the segment. The LOOPS ingress adds `_forward information_` to that traffic; the LOOPS egress removes the forward information and sends some `_reverse information_` to inform the behavior of the ingress.

Hence, in the data plane, forward information is added to each data packet. Reverse information can be sent in separate packets (e.g., Geneve control-only packets [I-D.ietf-nvo3-geneve]) and/or piggybacked on a related, reverse-direction LOOPS flow, similar to the way the forward information for that flow is carried. The setup protocol is used to provide the relationship between the LOOPS segments in the two directions that is used for piggybacking reverse information.

The above describes the "tunnel mode". A transparent mode is described in Appendix B, which does not modify the data packets and therefore needs to send any forward information (if needed, e.g., for FEC) in separate packets, usually aggregated.

The LOOPS `_generic information set_` defines what information is provided as setup information, forward information, and reverse information. `_Bindings_` map this information set to specific control plane and data plane protocols, including defining the specific encoding being used. Where separate packets (outside the data plane protocols being used) need to be sent, a special UDP-based protocol needs to be defined as well. The various bindings aim for some commonality, so that an implementation for multiple bindings does not need to support gratuitous variety between them.

## 5. LOOPS Generic Information Set

This section sketches a generic information set for the LOOPS protocol. Entries marked with (\*) are items that may not be necessary and probably should be left out of an initial specification.

### 5.1. Setup Information

Setup Information might include:

- \* encapsulation protocol in use, and its vital parameters
- \* identity of LOOPS ingress and LOOPS egress; information relevant for running the encapsulation protocol such as port numbers
- \* target maximum latency increase caused by the operation of LOOPS on this segment
- \* maximum retransmission count (\*)

### 5.2. Forward Information

In the forward information, we have identified:

- \* tunnel type (a few bits, meaning agreed between Ingress and Egress)
- \* packet sequence number PSN (20+ bits), counting the data packets forwarded transmitted by the LOOPS ingress (i.e., retransmissions re-use the PSN)
- \* an "ACK desirable" flag (one bit, usually set for a certain percentage of the data packets only)
- \* an optional blob, to be echoed by the egress
- \* anything that the FEC scheme needs.

The first four together (say, 3+24+4+1) might even fit into 32 bits, but probably need up to 48 bits total. FEC info of course often needs more space.

(Note that in this proposal there is no timestamp in the forward information; see Section 6.3.)

24 bits of PSN, minus one bit for sequence number arithmetic, gives 8 million packets (or 2.4 GB at typical packet sizes) per worst-case RTT. So if that is, say, 30 seconds, this would be enough to fill 640 Mbit/s.

### 5.3. Reverse Information

For the reverse information, we have identified:

- \* one optional block 1, possibly repeated:
  - PSN being acknowledged
  - absolute time of reception for the packet acknowledged (PSN)
  - the blob, if present, echoed back
- \* one optional block 2, possibly repeated:
  - an ACK bitmap (based on PSN), always starting at a multiple of 8
  - a delta indicating the end PSN of the bitmap (actually the first PSN that is beyond it), using  $(\text{Acked-PSN} \& \sim 7) + 8 * (\text{delta} + 1)$  as the end of the bitmap. Acked-PSN in that formula is the previous block 1 PSN seen in this packet, or 0 if none so far.

Block 1 and Block 2 can be interspersed and repeated. They can be piggybacked on a reverse direction data packet or sent separately if none occurs within some timeout. They will usually be aggregated in some useful form. Block 1 information sets are only returned for packets that have "ACK desirable" set. Block 2 information is sent by the receiver based on some saturation scheme (e.g., at least three copies for each PSN span over time). Still, it might be possible to go down to 1 or 2 amortized bytes per forward packet spent for all this.

The latency calculation is done by the sender, who occasionally sets "ACK desirable", and notes down the absolute time of transmission for this data packet (the timekeeping can be done quite efficiently as deltas). Upon reception of a block 1 ACK, it can then subtract that from the absolute time of reception indicated. This assumes time synchronization between the nodes is at least as good as the precision of latency measurement needed, which should be no problem with IEEE 1588 PTP synchronization (but could be if using NTP-based synchronization only). A sender can freely garbage collect noted down transmission time information; doing this too early just means that the quality of the RTT sampling will reduce.

## 6. LOOPS General Operation

In the Tunnel Mode described in the main body of this document, LOOPS information is carried by some tunnel encapsulation.

### 6.1. Initial Packet Sequence Number

There is no connection establishment procedure in LOOPS. The initial PSN is assigned unilaterally by the LOOPS Ingress.

Because of the short time that is usually set in the maximum latency increase, there is little damage from a collision of PSNs with packets still in flight from previous instances of LOOPS.

#### 6.1.1. Minimizing collisions

If desired, collisions can be minimized by assigning initial PSNs randomly, or using stable storage. Random assignment is more useful for longer PSNs, where the likelihood of overlap will be low. The specific way a LOOPS ingress uses stable storage is a local matter and thus out of scope. (Implementation note: this can be made to work similar to secure nonce generation with write attenuation: Say, every 10000 packets, the sender notes down the PSN into stable storage. After a reboot, it reloads the PSN and adds 10000 in sequence number arithmetic [RFC1982], plus maybe another 10000 so the sender does not have to wait for the store operation to succeed before sending more packets.)

#### 6.1.2. Optional Initial PSN procedure

As a potential option (to be discussed), an initial packet sequence number could be communicated using a simple two-bit protocol, based on an I flag (Initial PSN) carried in the forward information and an R flag (Initial PSN Received) in the reverse information. This procedure essentially clears the egress of any previous state, however, the benefits of this procedure are limited.

The initial PSN is assigned unilaterally by the LOOPS ingress, selected randomly. The ingress will keep setting the I flag to one when it starts to send packets from a new beginning or whenever it believes there is a need to notify the egress about a new initial PSN. The ingress will stop setting the I flag when it receives an acknowledgement with the R flag set from the egress.

When the LOOPS egress receives a packets with the I flag set, it stops performing services that assume a sequential PSN. The egress will no longer provide acknowledgement information for the packets with PSN smaller than this new initial PSN (per sequence number arithmetic [IEN74]). The egress sends acknowledgement information back without any delay by echoing the value of the I flag in the R flag. This also means the egress unsets the R flag in subsequent acknowledgements for packets with the I flag unset.

It may happen that the first few packets are lost in an initial PSN assignment process. In this case, the loss of these packets is not detectable by the LOOPS ingress since the first received PSN will be treated as an initial PSN at the egress. This is an acceptable temporary performance degradation: LOOPS does not intend to provide perfect reliability, and LOOPS usually applies to the aggregated traffic over a tunnel so that the initial PSN assignment happens infrequently.

## 6.2. Acknowledgement Generation

A data packet forwarded by the LOOPS ingress always carries PSN information. The LOOPS egress uses the largest newly received PSN with the "ACK desired" bit as the ACK number in the block 1 part of the acknowledgement. This means that the LOOPS ingress gets to modulate the number of acknowledgement sent by the LOOPS egress. However, whenever an out-of-order packet arrives while there still are "holes" in the PSNs received, the LOOPS receiver should generate a block 2 acknowledgement immediately that the LOOPS sender can use as an ACK list.

Reverse information can be piggybacked in a reverse direction data packet. When the reverse direction has no user data to be sent, a pure reverse information packet needs to be generated. This may be based on a short delay during which the LOOPS egress waits for a data packet to piggyback on. (To reduce MTU considerations, the egress could wait for less-than-full data packets.)



### 6.3. Measurement

When sending a block 1 acknowledgement, the LOOPS egress indicates the absolute time of reception of the packet. The LOOPS ingress can subtract the absolute time of transmission that it still has available, resulting in one high quality latency sample. (In an alternative design, the forward information could include the absolute time of transmission as well, and block1 information would echo it back. This trades memory management at the ingress for increased bandwidth and MTU reduction.)

When a data packet has been transmitted, it may not be clear which specific copy is acknowledged in a block 1 acknowledgement: the acknowledgement for the initial (or, more generally, an earlier) copy may have been delayed (ACK ambiguity)). The LOOPS ingress therefore SHOULD NOT base its measurements on acknowledgements for retransmitted data packets. One way to achieve this is by not setting the "ACK desired" bit on retransmissions in the first place.

The LOOPS ingress can also use the time of reception of the block 1 acknowledgement to obtain a segment RTT sample. Note that this will include any wait time the LOOPS egress incurs while waiting for a piggybacking opportunity -- this is appropriate, as all uses of an RTT will be for keeping a retransmission timeout.

To maintain quality of information during idle times, the LOOPS ingress may send keepalive packets, which are discarded at the LOOPS egress after sending acknowledgements. The indication that a packet is a keepalive packet is dependent on the encapsulation protocol.

#### 6.3.1. Ingress-relative timestamps

As an optional procedure, the ingress node can attach a small blob of data to a forward packet that carries an ACK desired flag; this blob is then echoed by the egress in its block 1 acknowledgement. This is typically used to attach a timestamp on a time scale defined by the ingress; we speak of an ingress-relative timestamp. Alternatively, the ingress can keep a timestamp in its local storage, associated with the PSN of the packet that carries an ACK desired flag; it can then retrieve this timestamp when the block 1 acknowledgement arrives.

In either case, the LOOPS ingress keeps track of the local segment round trip time (LRTT) based on the (saved or received) timestamp and the arrival time of the block 1 acknowledgement, by setting the ACK Desired flag (D flag) occasionally (several times per RTT) and saving/including a sending timestamp for/in the packet.

As the egress will send block 1 acknowledgement information right away when it receives a packet with the D flag set, the measurement of LRTT is more accurate for such packets. A smoothed local segment round trip time S\_LRTT can be computed in a similar way as defined by [RFC0793]. A recent minimum value of LRTT is also kept as min\_LRTT. S\_LRTT is used as a basis for the overall timing of retransmission and state management.

Retransmitted packets MUST NOT be used for local segment round trip time (LRTT) calculation.

#### 6.3.2. ACK generation

A block 1 acknowledgement is generated based on receiving a forward packet with a D flag.

The way block 2 acknowledgement information is sent is more subject to control by the egress. Generally, the egress will aggregate ACK bits for at least K packets before sending a block 2; this can be used to amortize the overhead to close to a couple of bits per ACK. In order to counter loss of reverse information packets, an egress will also want to send an ACK bit more than once -- a saturation value of 3 or more may be chosen based on setup information. Typically, ACK bits already sent will be prepended to ACK bits that are new in this block 2 information set. If K packets do not accumulate for a while, the egress will send one or more packets with block 2 information that covers the unsent ACK bits it has so far.

(Discussion: This works best if the egress has information both about the S\_RTT and min\_RTT that the ingress uses and the reverse packet loss rate.)

#### 6.4. Loss detection and Recovery

There are two ways for LOOPS local recovery, retransmission and FEC.

##### 6.4.1. Local Retransmission

When retransmission is used as recovery mechanism, the LOOPS ingress detects a packet loss by not receiving an ACK for the packet within the time expected based on an RTO value (which might be calculated as in [RFC6298]). Packet retransmission should then not be performed more than once within an LRTT.

When a retransmission is desired, the LOOPS ingress performs the local in-network recovery by retransmitting the packet. Further retransmissions may be desirable if the lack of ACK is persistent beyond an RTO, as long as the maximum latency increase is not reached.

#### 6.4.2. FEC

FEC is another way to perform local recovery. When FEC is in use, a FEC header is sent with data packets as well as with special repair packets added to the flow. The specific FEC scheme used could be defined in the Setup Information, using a mechanism like [RFC5052]. The FEC rate (amount of redundancy added) and possibly the FEC scheme could be unilaterally adjusted by the LOOPS ingress in an adaptive mechanism based on the measurement information.

#### 6.5. Discussion

Without progress in the way that end-host transport protocols handle reordering, LOOPS will be unable to prevent end-to-end retransmissions that duplicate effort that is spent in local retransmissions. It depends on parameters of the path segment whether this wasted effort is significant or not.

One remedy against this waste could be the introduction of resequencing at the LOOPS Egress node. This increases overall mean packet latency, but does not always increase actual end-to-end data stream latency if a head-of-line blocking transport such as TCP is in use. For applications with a large percentage of legacy TCP end-hosts and sufficient processing capabilities at the LOOPS Egress node, resequencing may be a viable choice. Note that resequencing could be switched off and on depending on some measurement information.

The packet numbering scheme chosen by LOOPS already provides the necessary information for the LOOPS Egress to reconstruct the sequence of data packets at the LOOPS ingress.

#### 7. Sketches of Bindings to Tunnel Protocols

The LOOPS information defined above in a generic way can be mapped to specific tunnel encapsulation protocols. A sketch for the tunnel protocol Geneve is given below (Section 7.1). The actual encapsulation can be designed in a "native" way by putting each of the various elements into the TLV format of the encapsulation protocol, or it can be achieved by providing single TLVs for forward and reverse information and using some generic encoding of both kinds of information as shown in Appendix B.3.

### 7.1. Embedding LOOPS in Geneve

Geneve [I-D.ietf-nvo3-geneve] is an extensible overlay protocol which can embed LOOPS functions. Geneve uses TLVs to carry optional information between NVEs. NVE is logically the same entity as the LOOPS node.

The Geneve header has a mandatory Virtual Network Identifier (VNI) field. The specific VNI value to be used is part of the setup information for the LOOPS tunnel.

More details for a Geneve binding for LOOPS can be found in [I-D.bormann-loops-geneve-binding].

### 8. IANA Considerations

No IANA action is required at this stage. When a LOOPS representation is designed for a specific tunneling protocol, new codepoints will be required in the registries that pertain to that protocol.

### 9. Security Considerations

The security of a specific LOOPS segment will depend both on the properties of the generic information set described here and those of the encapsulation protocol employed. The security considerations of the encapsulation protocol will apply, as will the protection afforded by any security measures provided by the encapsulation protocol. Any LOOPS encapsulation specification is expected to provide information about preferred configurations of the encapsulation protocol employed, including security mechanisms, and to provide a security considerations section discussing the combination. The following discussion aims at discussing security considerations that will be common between different encapsulations.

#### 9.1. Threat model

Attackers might attempt to perturb the operation of a LOOPS segment for a number of purposes:

- \* Denial of Service: Damaging the ability of LOOPS to recover packets, or damaging packet forwarding through the LOOPS segment in general.
- \* Attacks on Confidentiality or Integrity: Obtaining the content of data packets, modifying them, injecting new or suppressing specific data packets.

For the purposes of these security considerations, we can distinguish three classes of attackers:

1. on-path read-write: The attacker sees packets under way on the segment and can modify, inject, or suppress them.

In this case there is really nothing LOOPS can do, except for acting as a full security protocol on its own, which would be the task of the encapsulation protocol. Without that, attackers already can manipulate the packet stream as they wish. This class of attackers is considered out of scope for these security considerations.

2. on-path read + inject: The attacker sees packets under way on the segment and can inject new packets.

For this case, LOOPS itself similarly cannot add to the confidentiality of the data stream. However, LOOPS could protect against denial of service against its own protocol operation and, in a limited fashion, against attacks on integrity that wouldn't already have been possible by packet injection without LOOPS.

3. off-path inject: The attacker can inject new packets, but cannot see existing packets under way on the segment.

Similar considerations apply as for class 2, except that the "blind" class 3 attacker might need to guess information it could have extracted from the packet stream in class 2.

## 9.2. Discussion

Class 2 attackers can see e.g. sequence numbers and can inject, but not modify traffic. Attacks might include injecting false ACKs, initial PSN flags, ... (TBD)

Class 3 ("blind") attackers might still be able to fake initial PSN bits + false ACKs, but will have a harder time otherwise as it would need to guess the PSN range in which it can wreak havoc. Even random guesses will sometimes hit, though, so the protocol needs to be robust to such injection attacks. ... (TBD)

## 10. References

### 10.1. 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>>.

## 10.2. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, DOI 10.17487/RFC3708, February 2004, <<https://www.rfc-editor.org/info/rfc3708>>.
- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <<https://www.rfc-editor.org/info/rfc5052>>.
- [RFC5862] Yasukawa, S. and A. Farrel, "Path Computation Clients (PCC) - Path Computation Element (PCE) Requirements for Point-to-Multipoint MPLS-TE", RFC 5862, DOI 10.17487/RFC5862, June 2010, <<https://www.rfc-editor.org/info/rfc5862>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.

- [RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <<https://www.rfc-editor.org/info/rfc6330>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [I-D.ietf-tcpm-rack]  
Cheng, Y., Cardwell, N., Dukkupati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-rack-08, 9 March 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-tcpm-rack-08.txt>>.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", Work in Progress, Internet-Draft, draft-ietf-nvo3-geneve-16, 7 March 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-nvo3-geneve-16.txt>>.
- [I-D.li-tsvwg-loops-problem-opportunities]  
Yizhou, L., Zhou, X., Boucadair, M., Wang, J., and F. Qin, "LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement", Work in Progress, Internet-Draft, draft-li-tsvwg-loops-problem-opportunities-05, 6 July 2020, <<http://www.ietf.org/internet-drafts/draft-li-tsvwg-loops-problem-opportunities-05.txt>>.
- [I-D.ietf-tsvwg-tunnel-congestion-feedback]  
Wei, X., Yizhou, L., Boutros, S., and L. Geng, "Tunnel Congestion Feedback", Work in Progress, Internet-Draft, draft-ietf-tsvwg-tunnel-congestion-feedback-07, 5 May 2019, <<http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-tunnel-congestion-feedback-07.txt>>.

- [I-D.bormann-loops-geneve-binding]  
Bormann, C., "Embedding LOOPS in Geneve", Work in Progress, Internet-Draft, draft-bormann-loops-geneve-binding-01, 12 June 2020, <<http://www.ietf.org/internet-drafts/draft-bormann-loops-geneve-binding-01.txt>>.
- [IEN74] Plummer, W.W., "Sequence Number Arithmetic", Internet Experiment Note 74, September 1978.

#### Appendix A. Protocol used in Prototype Implementation

This appendix describes, in a somewhat abstracted form, the protocol as used in a prototype implementation, as described by Yizhou Li, and Xingwang Zhou.

The prototype protocol can be run in one of two modes (defined by preconfiguration):

- \* Retransmission mode
- \* Forward Error Correction (FEC) mode

Forward information is piggybacked in data packets.

Reverse information can be carried in a pure acknowledgement packet or piggybacked when carrying packets for the inverse direction.

The forward information includes:

- \* Packet Sequence Number (PSN) (32 bits): This identifies a packet over a specific overlay segment from a specific LOOPS Ingress. If a packet is retransmitted by LOOPS, the retransmission uses the original PSN.
- \* Timestamp (32 bits): Information, in a format local to the LOOPS ingress, that provides the time when the packet was sent. In the current implementation, a 32-bit unsigned value specifying the time delta in some granularity from the epoch time to the sending time of the packet carrying this timestamp. The granularity can be from 1 ms to 1 second. The epoch time follows the current TCP practice which is 1 January 1970 00:00:00 UTC. Note that a retransmitted packet uses its own Timestamp.
- \* FEC Info for Block Code (56 bits): This header is used in FEC mode. It currently only provides for a block code FEC scheme. It includes the Source Block Number (SBN), Encoding Symbol ID (ESI), number of symbols in a single source block and symbol size. Appendix A.1 gives more details on FEC.



The reverse information includes:

- \* ACK Number (32 bits): The largest (in sequence number arithmetic [RFC1982]) PSN received so far.
- \* ACK List (variable): This indicates an array of PSN numbers to describe the PSN "holes" preceding the ACK number. It conceptually lists the PSNs of every packet perceived as lost by the LOOPS egress. In actual use, it is truncated.
- \* Echoed Timestamp (32 bits): The timestamp received with the packet being acknowledged.

#### A.1. Block Code FEC

The prototype currently uses a block code FEC scheme (RaptorQ [RFC6330]). The fields in the FEC Info forward information are:

- \* Source Block Number (SBN): 16 bits. An integer identifier for the source block that the encoding symbols within the packet relate to.
- \* Encoding Symbol ID (ESI): 16 bits. An integer identifier for the encoding symbols within the packet.
- \* K: 8 bits. Number of symbols in a single source block.
- \* T: 16 bits. Symbol size in bytes.

The LOOPS Ingress uses the data packet in Figure 1 to generate the encoding packet. Both source packets and repair packets carry the FEC header information; the LOOPS Egress reconstructs the data packets from both kinds of packets. The LOOPS Egress currently resequences the forwarded and reconstructed packets, so they are passed on in-order when the lost packets are recoverable within the source block.

The LOOPS Nodes need to agree on the use of FEC block mode and on the specific FEC Encoding ID to use; this is currently done by configuration.

#### Appendix B. Transparent mode

This appendix defines a very different way to provide the LOOPS services, "transparent mode". (We call the protocol described in the main body of the document "encapsulated mode".)

In transparent mode, the idea is that LOOPS does not meddle with the forward transmission of data packets, but runs on the side exchanging additional information.

An implementation could be based on conventional forwarding switches that just provide a copy of the ingress and egress packet stream to the LOOPS implementations. The LOOPS process would occasionally inject recovered packets back into the LOOPS egress node's forwarding switch, see Figure 3.

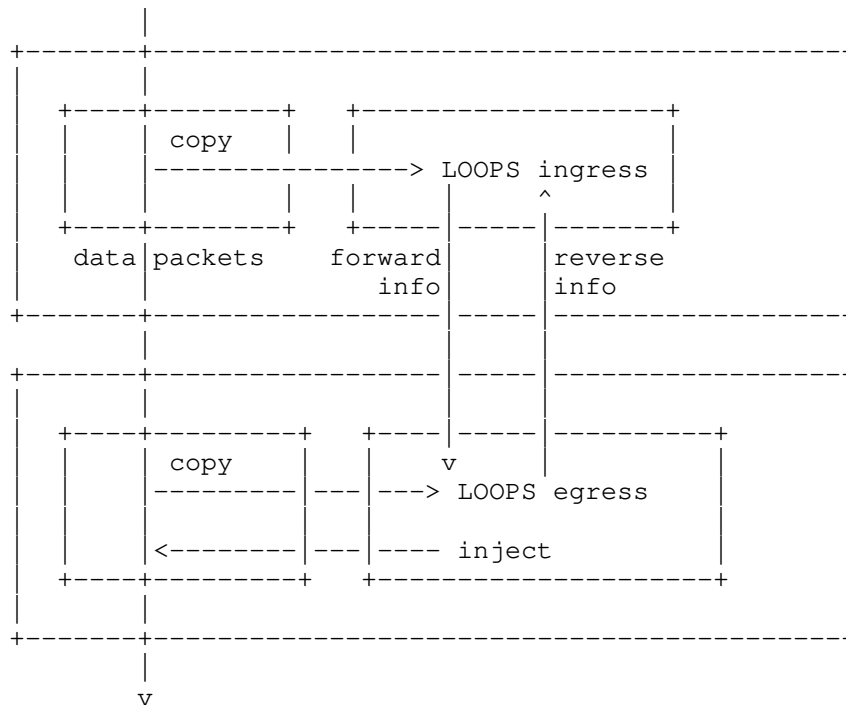


Figure 3: LOOPS Transparent Mode

The obvious advantage of transparent mode is that no encapsulation is needed, reducing processing requirements and keeping the MTU unchanged. The obvious disadvantage is that no forward information can be provided with each data packet, so a replacement needs to be found for the PSN (packet sequence number) employed in encapsulated mode. Any forward information beyond the data packets is sent in separate packets exchanged directly between the LOOPS nodes.

### B.1. Packet identification

Retransmission mode and FEC mode differ in their needs for packet identification. For retransmission mode, a somewhat probabilistic accuracy of the packet identification is sufficient, for FEC mode, packet identification should not make mistakes (as these would lead to faultily reconstructed packets).

In Retransmission mode, misidentification of a packet could lead to measurement errors as well as missed retransmission opportunities. The latter will be fixed end-to-end. The tolerance for measurement errors would influence the degree of accuracy that is aimed for.

Packet identification can be based on a cryptographic hash of the packet, computed in LOOPS ingress and egress using the same algorithm (excluding fields that can change in transit, such as TTL/hop limit). The hash can directly be used as a packet number, or it can be sent in the forward information together with a packet sequence number, establishing a mapping.

For probabilistic packet identification, it is almost always sufficient to hash the first few (say, 64) bytes of the packet; all known transport protocols keep sufficient identifying information in that part (and, for encrypted protocols, the entropy will be sufficient). Any collisions of the hash could be used to disqualify the packet for measurement purposes, minimizing the measurement errors; this could allow rather short packet identifiers in retransmission mode.

For FEC mode, the packet identification together with the per-packet FEC information needs to be sent in the (separate) forward information, so that a systematic code can be reconstructed. For retransmission mode, there is no need to send any forward information for most packets, or a mapping from packet identifiers to packet sequence numbers could be sent in the forward information (probably in some aggregated form). The latter would allow keeping the acknowledgement form described in the main body (with aggregate acknowledgement); otherwise, packet identifiers need to be acknowledged. With this change, the LOOPS egress will send reverse information as in the encapsulating LOOPS protocol.

## B.2. Generic information and protocol operation

With the changes outlined above, transparent mode operates just as encapsulated mode. If packet sequence numbers are not used, there is no use for block2 reverse information; if they are used, a new block3 needs to be defined that provides the mapping from packet identifiers to packet sequence numbers in the forward information. To avoid MTU reduction, some mechanism will be needed to encapsulate the actual FEC information (additional packets) in the forward information.

## B.3. A hybrid mode

Figure 3 can be modified by including a GRE encapsulator into the top left corner and a GRE decapsulator in the bottom left corner. This provides more defined ingress and egress points, but it also provides an opportunity to add a packet sequence number at the ingress. The copies to the top right and bottom right corners are the encapsulated form, i.e., include the sequence number.

The GRE packet header then has the form:

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|0|0|1|          000000000          | 000 |          Protocol Type          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Sequence Number                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The forward and reverse information can be designed closer to the approach in the main body of the document, to be exchanged using UDP packets between top right ingress and bottom right egress using a port number allocated for this purpose.

Rough ideas for both directions are given below in CDDL [RFC8610]. This information set could be encoded in CBOR or in a bespoke encoding; details such as this can be defined later.

```
forward-information = [  
  [rel-psn, ack-desired, ? fec-info] /  
  fec-repair-data  
]  
  
rel-psn = uint; relative packet sequence number  
; always given as a delta from the previous one in the array  
; starting out with a "previous value" of 0  
  
ack-desired = bool  
  
fec-info = [  
  sbn: uint, ; Source Block Number  
  esi: uint, ; Encoding Symbol ID  
  ? (  
    nsssb: uint; number of symbols in a single source block  
    ss: uint; symbol size  
  )  
]  
  
fec-repair-data = [  
  repair-data: bytes  
  ? (  
    sbn: uint, ; Source Block Number  
    esi: uint, ; Encoding Symbol ID  
  )  
]
```

If left out for a sequence number, the fec-info block is constructed by adding one to the previous one. fec-repair-data contain repair symbols for the sbn/esi given (which, again, are reconstructed from context if not given).

```
reverse-information = [  
  block1 / block2  
]
```

```
block1 = [rel-psn, timestamp]  
block2 = [end-psn-delta: uint, acked-bits: bytes]
```

The acked-bits in a block2 is a bitmap that gives acknowledgments for received data packets. The bitmap always comes as a multiple of 8 bits (all bytes are filled in with 8 bits, each identifying a PSN). The end PSN of the bitmap (actually the first PSN that would be beyond it) is computed from the current PSN as set by rel-psn, rounded down to a multiple of 8, and adding  $8 * (\text{end-psn-delta} + 1)$  to that value.

#### Acknowledgements

Sami Boutros helped with sketching the use of Geneve (Section 7.1).

Michael Welzl has been supported by the Research Council of Norway under its "Toppforsk" programme through the "OCARINA" project.

#### Authors' Addresses

Michael Welzl  
University of Oslo  
PO Box 1080 Blindern  
N-0316 Oslo  
Norway

Phone: +47 22 85 24 20  
Email: michawe@ifi.uio.no

Carsten Bormann (editor)  
Universität Bremen TZI  
Postfach 330440  
D-28359 Bremen  
Germany

Phone: +49-421-218-63921  
Email: cabo@tzi.org

NVO3 Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 24, 2020

X. Min  
G. Mirsky  
ZTE Corp.  
S. Pallagatti  
VMware  
October 22, 2019

BFD for Geneve  
draft-xiao-nvo3-bfd-geneve-01

Abstract

This document describes the use of the Bidirectional Forwarding Detection (BFD) protocol in point-to-point Generic Network Virtualization Encapsulation (Geneve) tunnels forming up an overlay network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions Used in This Document . . . . .	3
2.1. Terminology . . . . .	3
2.2. Requirements Language . . . . .	3
3. BFD Packet Transmission over Geneve Tunnel . . . . .	3
3.1. BFD Encapsulation With Inner Ethernet/IP/UDP Header . . . . .	3
3.2. BFD Encapsulation With Inner IP/UDP Header . . . . .	6
3.3. BFD Encapsulation With Inner MPLS Header . . . . .	8
4. Reception of BFD packet from Geneve Tunnel . . . . .	10
4.1. Demultiplexing of the BFD packet . . . . .	11
5. Security Considerations . . . . .	11
6. IANA Considerations . . . . .	11
7. Acknowledgements . . . . .	12
8. Normative References . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

"Generic Network Virtualization Encapsulation" (Geneve)

[I-D.ietf-nvo3-geneve] provides an encapsulation scheme that allows building an overlay network by decoupling the address space of the attached virtual hosts from that of the network.

This document describes the use of Bidirectional Forwarding Detection (BFD) protocol [RFC5880] to enable monitoring continuity of the path between two Geneve tunnel endpoints, which may be NVE (Network Virtualization Edge) or other device acting as a Geneve tunnel endpoint. For simplicity, in this document, NVE is used to represent Geneve tunnel endpoint, TS (Tenant System) is used to represent the physical or virtual device attached to a Geneve tunnel endpoint from the outside. VAP (Virtual Access Point) is the NVE side of the interface between the NVE and the TS, and a VAP is a logical network port (virtual or physical) into a specific virtual network. For detailed definitions and descriptions of NVE, TS and VAP, please refer to [RFC7365] and [RFC8014].

The use cases and the deployment of BFD for Geneve are consistent with what's described in Section 1 and Section 3 of [I-D.ietf-bfd-vxlan]. The major difference between Geneve and "Virtual eXtensible Local Area Network" (VXLAN) [RFC7348] encapsulation is that Geneve supports multi-protocol payload and variable length options.



## 2. Conventions Used in This Document

### 2.1. Terminology

BFD: Bidirectional Forwarding Detection

CC: Continuity Check

GAL: Generic Associated Channel Label

G-ACh: Generic Associated Channel

Geneve: Generic Network Virtualization Encapsulation

MPLS: Multiprotocol Label Switching

NVE: Network Virtualization Edge

TS: Tenant System

VAP: Virtual Access Point

VNI: Virtual Network Identifier

VXLAN: Virtual eXtensible Local Area Network

### 2.2. Requirements Language

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. BFD Packet Transmission over Geneve Tunnel

Concerning whether or not the Geneve data packets include an IP protocol data unit, and whether or not the Geneve data packets include an MPLS protocol data unit, this document considers three options of BFD packet encapsulation in Geneve.

### 3.1. BFD Encapsulation With Inner Ethernet/IP/UDP Header

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that there exists an inner Ethernet header, i.e., the Protocol Type equals to 0x6558 (Ethernet frame), then BFD packets are encapsulated in Geneve as described below. The Geneve packet format over IPv4 is defined in

Section 3.1 of [I-D.ietf-nvo3-geneve]. The Geneve packet format over IPv6 is defined in Section 3.2 of [I-D.ietf-nvo3-geneve]. The Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve]. Note that the outer IP header and the inner IP header may not be of the same address family, in other words, outer IPv6 header accompanied with inner IPv4 header and outer IPv4 header accompanied with inner IPv6 header are both possible.

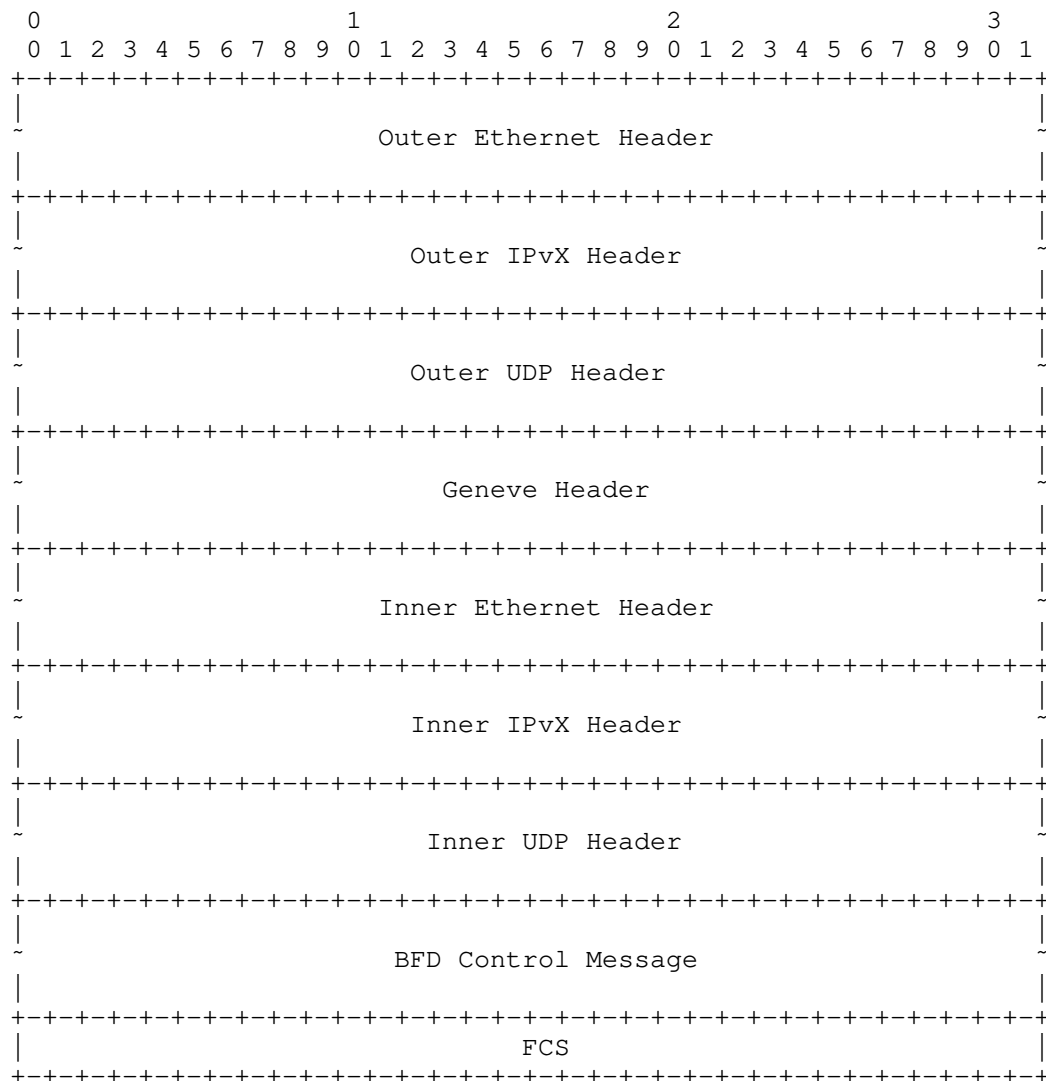


Figure 1: Geneve Encapsulation of BFD Control Message With the Inner Ethernet/IP/UDP Header

The BFD packet **MUST** be carried inside the inner Ethernet frame of the Geneve packet, as specified in Section 4 of [I-D.ietf-bfd-vxlan].

When the BFD packets are encapsulated in Geneve in this way, the Geneve header follows the value set below.

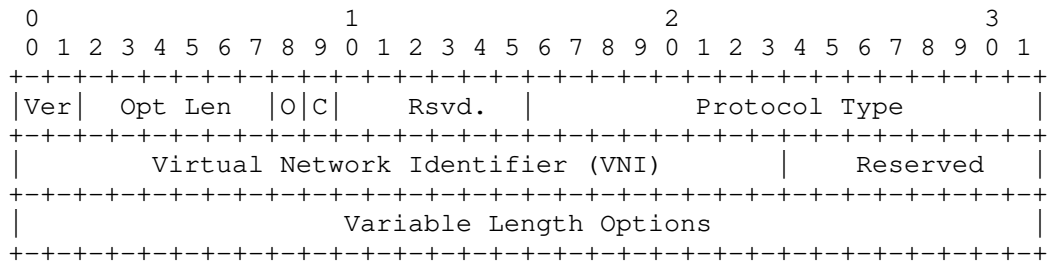


Figure 2: Geneve Header

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0.

Protocol Type field MUST be set to 0x6558 (Ethernet frame).

### 3.2. BFD Encapsulation With Inner IP/UDP Header

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that there exists an inner IP header, i.e., the Protocol Type equals to 0x0800 (IPv4) or 0x86DD (IPv6), then BFD packets are encapsulated in Geneve as described below. The Geneve packet format over IPv4 is defined in Section 3.1 of [I-D.ietf-nvo3-geneve]. The Geneve packet format over IPv6 is defined in Section 3.2 of [I-D.ietf-nvo3-geneve]. The Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve]. Note that the outer IP header and the inner IP header may not be of the same address family, in other words, outer IPv6 header accompanied with inner IPv4 header and outer IPv4 header accompanied with inner IPv6 header are both possible.

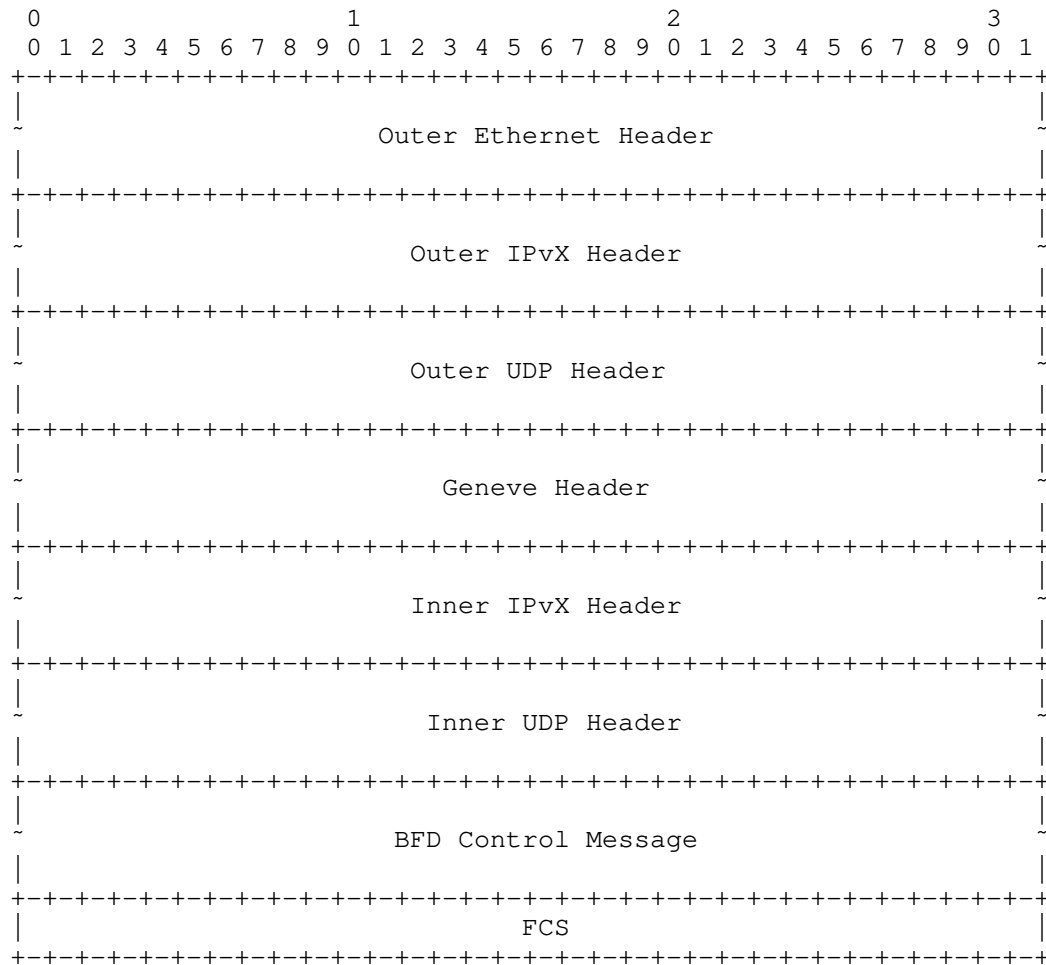


Figure 3: Geneve Encapsulation of BFD Control Message With the Inner IP/UDP Header

The BFD packet MUST be carried inside the inner IP packet of the Geneve packet. The inner IP packet carrying the BFD payload has the following format:

IP header:

Source IP: IP address of a VAP of the originating NVE.

Destination IP: IP address of a VAP of the terminating NVE.

TTL: MUST be set to 1 to ensure that the BFD packet is not routed within the L3 underlay network.

The fields of the UDP header and the BFD control packet are encoded as specified in [RFC5881].

When the BFD packets are encapsulated in Geneve in this way, the Geneve header follows the value set below.

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0.

Protocol Type field MUST be set to 0x0800 (IPv4) or 0x86DD (IPv6), depending on the address family of the inner IP packet.

### 3.3. BFD Encapsulation With Inner MPLS Header

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that there exists an inner MPLS header, i.e., the Protocol Type equals to 0x8847 (MPLS) or 0x8848 (MPLS with the upstream-assigned label), then BFD packets are encapsulated in Geneve as described below. The Geneve packet format over IPv4 is defined in Section 3.1 of [I-D.ietf-nvo3-geneve]. The Geneve packet format over IPv6 is defined in Section 3.2 of [I-D.ietf-nvo3-geneve]. The Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve].

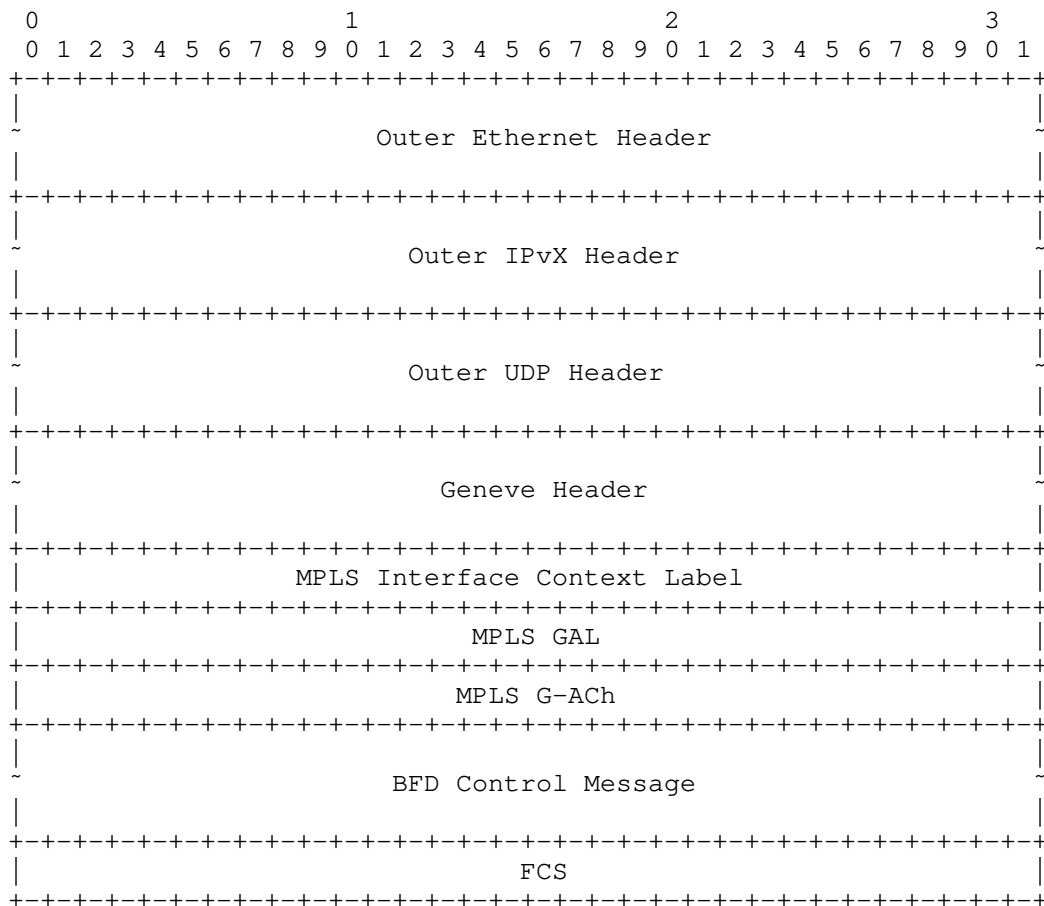


Figure 4: Geneve Encapsulation of BFD Control Message With the Inner MPLS GAL/G-ACh

The BFD packet MUST be carried inside the inner MPLS packet of the Geneve packet. The inner MPLS packet carrying the BFD payload has the following format:

MPLS Interface Context Label: This Label would be used to identify a VAP of the originating NVE and a VAP of the terminating NVE.

MPLS GAL (Generic Associated Channel Label):

Label value: MUST be set to 13, as specified in [RFC5586].

S bit: MUST be set to 1.

TTL: MUST be set to 1.

The fields of the MPLS G-ACh (Generic Associated Channel) and the BFD control packet are encoded as specified for MPLS-TP CC (Continuity Check) message in [RFC6428].

When the BFD packets are encapsulated in Geneve in this way, the Geneve header follows the value set below.

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0.

Protocol Type field MUST be set to 0x8847 (MPLS).

#### 4. Reception of BFD packet from Geneve Tunnel

Once a packet is received, NVE MUST validate the packet as described in [I-D.ietf-nvo3-geneve].

If the Protocol Type field equals 0x6558 (Ethernet frame), and the Destination MAC of the inner Ethernet frame matches one MAC address owned by the NVE, the Destination IP, the UDP destination port and the TTL of the inner IP packet MUST be validated to determine whether the received packet can be processed by BFD. BFD packet with inner MAC set to NVE MUST NOT be forwarded to TSs.

If the Protocol Type field equals 0x0800 (IPv4) or 0x86DD (IPv6), and the Destination IP of the inner IP packet matches a VAP IP address of the NVE, the UDP destination port and the TTL of the inner IP packet MUST be validated to determine whether the received packet can be processed by BFD. BFD packet with inner IP set to NVE MUST NOT be forwarded to TSs.

If the Protocol Type field equals 0x8847 (MPLS), the MPLS Interface Context Label, the MPLS GAL and the MPLS G-ACh of the inner MPLS packet MUST be validated to determine whether the received packet can be processed by BFD. BFD packet with MPLS GAL MUST NOT be forwarded to TSs.



#### 4.1. Demultiplexing of the BFD packet

In BFD over Geneve, a BFD session is originated and terminated at VAP, usually one NVE owns multiple VAPs, so multiple BFD sessions may be running between two NVEs, there needs to be a mechanism for demultiplexing received BFD packets to the proper session.

If the BFD packet is received with Your Discriminator equals to 0, for different BFD encapsulation, the procedure for demultiplexing the received BFD packets is different.

When the BFD Encapsulation With Inner Ethernet/IP/UDP Header is used, the BFD session MUST be identified using the procedure specified in Section 5.1 of [I-D.ietf-bfd-vxlan].

When the BFD Encapsulation With Inner IP/UDP Header is used, the BFD session MUST be identified using the inner IP/UDP header, i.e., the source IP and the destination IP present in the inner IP/UDP header.

When the BFD Encapsulation With Inner MPLS Header is used, the BFD session MUST be identified using the inner MPLS header, i.e., the MPLS Interface Context Label present in the inner MPLS header.

If the BFD packet is received with non-zero Your Discriminator, then BFD session MUST be demultiplexed only with Your Discriminator as the key.

With respect to BFD for Geneve, the use of the specific VNI would follow the principle as specified in Section 6 of [I-D.ietf-bfd-vxlan].

[Ed.Note]: Currently it's still undetermined whether "BFD for VxLAN" should allow multiple BFD sessions for the same VNI. The Editor leans to believe "BFD for Geneve" should allow multiple BFD sessions for the same VNI, and it needs further discussion.

#### 5. Security Considerations

This document does not raise any additional security issues beyond those of the specifications referred to in the list of normative references.

#### 6. IANA Considerations

This document has no IANA action requested.

## 7. Acknowledgements

The authors would like to acknowledge Reshad Rahman, Jeffrey Haas and Matthew Bocci for their guidance on this work.

## 8. Normative References

- [I-D.ietf-bfd-vxlan]  
Networks, J., Paragiri, S., Govindan, V., Mudigonda, M.,  
and G. Mirsky, "BFD for VXLAN", draft-ietf-bfd-vxlan-07  
(work in progress), May 2019.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic  
Network Virtualization Encapsulation", draft-ietf-  
nvo3-geneve-14 (work in progress), September 2019.
- [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>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed.,  
"MPLS Generic Associated Channel", RFC 5586,  
DOI 10.17487/RFC5586, June 2009,  
<<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection  
(BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,  
<<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection  
(BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881,  
DOI 10.17487/RFC5881, June 2010,  
<<https://www.rfc-editor.org/info/rfc5881>>.
- [RFC6428] Allan, D., Ed., Swallow, G., Ed., and J. Drake, Ed.,  
"Proactive Connectivity Verification, Continuity Check,  
and Remote Defect Indication for the MPLS Transport  
Profile", RFC 6428, DOI 10.17487/RFC6428, November 2011,  
<<https://www.rfc-editor.org/info/rfc6428>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger,  
L., Sridhar, T., Bursell, M., and C. Wright, "Virtual  
eXtensible Local Area Network (VXLAN): A Framework for  
Overlaying Virtualized Layer 2 Networks over Layer 3  
Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014,  
<<https://www.rfc-editor.org/info/rfc7348>>.

- [RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <<https://www.rfc-editor.org/info/rfc7365>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [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>>.

## Authors' Addresses

Xiao Min  
ZTE Corp.  
Nanjing  
China

Phone: +86 25 88013062  
Email: xiao.min2@zte.com.cn

Greg Mirsky  
ZTE Corp.  
USA

Email: gregimirsky@gmail.com

Santosh Pallagatti  
VMware

Email: santosh.pallagatti@gmail.com

NVO3 Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 10, 2021

X. Min  
G. Mirsky  
ZTE Corp.  
S. Pallagatti  
VMware  
J. Tantsura  
Apstra  
July 9, 2020

BFD for Geneve  
draft-xiao-nvo3-bfd-geneve-03

## Abstract

This document describes the use of the Bidirectional Forwarding Detection (BFD) protocol in point-to-point Generic Network Virtualization Encapsulation (Geneve) tunnels used to make up an overlay network.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 10, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions Used in This Document . . . . .	3
2.1. Abbreviations . . . . .	3
2.2. Requirements Language . . . . .	3
3. BFD Packet Transmission over Geneve Tunnel . . . . .	3
3.1. BFD Encapsulation With Inner Ethernet/IP/UDP Header . . . . .	3
3.2. BFD Encapsulation With Inner IP/UDP Header . . . . .	6
4. Reception of BFD packet from Geneve Tunnel . . . . .	8
4.1. Demultiplexing of the BFD packet . . . . .	9
5. Security Considerations . . . . .	9
6. IANA Considerations . . . . .	9
7. Acknowledgements . . . . .	9
8. References . . . . .	10
8.1. Normative References . . . . .	10
8.2. Informative References . . . . .	10
Authors' Addresses . . . . .	11

## 1. Introduction

"Generic Network Virtualization Encapsulation" (Geneve) [I-D.ietf-nvo3-geneve] provides an encapsulation scheme that allows building an overlay network by decoupling the address space of the attached virtual hosts from that of the network.

This document describes the use of Bidirectional Forwarding Detection (BFD) protocol [RFC5880] to enable monitoring continuity of the path between two Geneve tunnel endpoints, which may be NVE (Network Virtualization Edge) or other device acting as a Geneve tunnel endpoint. Specifically, the asynchronous mode of BFD, as defined in [RFC5880], is used to monitor a p2p Geneve tunnel, and support for BFD Echo function is outside the scope of this document. For simplicity, in this document, NVE is used to represent Geneve tunnel endpoint, TS (Tenant System) is used to represent the physical or virtual device attached to a Geneve tunnel endpoint from the outside. VAP (Virtual Access Point) is the NVE side of the interface between the NVE and the TS, and a VAP is a logical network port (virtual or physical) into a specific virtual network. For detailed definitions and descriptions of NVE, TS and VAP, please refer to [RFC7365] and [RFC8014].

The use cases and the deployment of BFD for Geneve are consistent with what's described in Section 1 and 3 of [I-D.ietf-bfd-vxlan].

The major difference between Geneve and "Virtual eXtensible Local Area Network" (VXLAN) [RFC7348] encapsulation is that Geneve supports multi-protocol payload and variable length options.

## 2. Conventions Used in This Document

### 2.1. Abbreviations

BFD: Bidirectional Forwarding Detection

Geneve: Generic Network Virtualization Encapsulation

NVE: Network Virtualization Edge

TS: Tenant System

VAP: Virtual Access Point

VNI: Virtual Network Identifier

VXLAN: Virtual eXtensible Local Area Network

### 2.2. Requirements Language

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. BFD Packet Transmission over Geneve Tunnel

Concerning whether the Geneve data packets include an Ethernet frame or an IP packet, this document defines two formats of BFD packet encapsulation in Geneve. BFD session is originated and terminated at VAP of an NVE, selection of the BFD packet encapsulation is based on how the VAP encapsulates the data packets.

### 3.1. BFD Encapsulation With Inner Ethernet/IP/UDP Header

If the VAP that originates the BFD packets is used to encapsulate Ethernet data frames, then BFD packets are encapsulated in Geneve as described below. The Geneve packet format over IPv4 is defined in Section 3.1 of [I-D.ietf-nvo3-geneve]. The Geneve packet format over IPv6 is defined in Section 3.2 of [I-D.ietf-nvo3-geneve]. The Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve]. Note that the outer IP header and the inner IP header may not be of the same address family, in other words,

outer IPv6 header accompanied with inner IPv4 header and outer IPv4 header accompanied with inner IPv6 header are both possible.

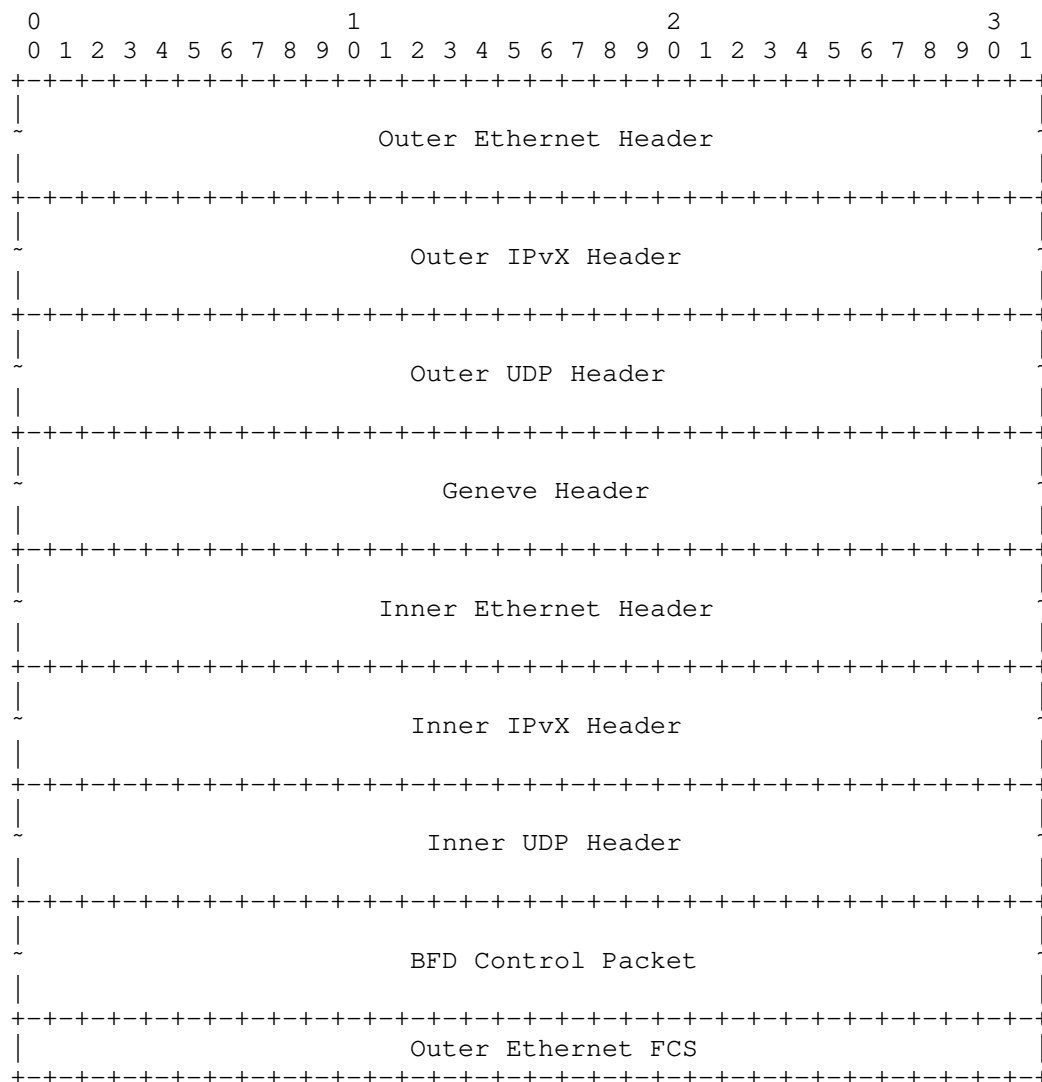


Figure 1: Geneve Encapsulation of BFD Control Packet With the Inner Ethernet/IP/UDP Header

The BFD packet MUST be carried inside the inner Ethernet frame of the Geneve packet. The inner Ethernet frame carrying the BFD Control packet has the following format:

Ethernet Header:

Source MAC: MAC address of a VAP of the originating NVE.

Destination MAC: MAC address of a VAP of the terminating NVE.

IP Header:

Source IP: IP address of a VAP of the originating NVE. If the VAP of the originating NVE has no IP address, then the IP address of the originating NVE is used.

Destination IP: IP address of a VAP of the terminating NVE. If the VAP of the terminating NVE has no IP address, then the IP address MUST be chosen from the 127/8 range for IPv4, and from the ::ffff:127.0.0.0/104 range for IPv6.

TTL or Hop Limit: MUST be set to 255 in accordance with [RFC5881].

The fields of the UDP header and the BFD Control packet are encoded as specified in [RFC5881].

When the BFD packets are encapsulated in Geneve in this way, the Geneve header defined in [I-D.ietf-nvo3-geneve] follows the value set below.

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0, which indicates there isn't any critical option.

Protocol Type field MUST be set to 0x6558 (Ethernet frame).

Virtual Network Identifier (VNI) field SHOULD be set to the VNI number that the originating VAP is mapped to.



### 3.2. BFD Encapsulation With Inner IP/UDP Header

If the VAP that originates the BFD packets is used to encapsulate IP data packets, then BFD packets are encapsulated in Geneve as described below. The Geneve packet format over IPv4 is defined in Section 3.1 of [I-D.ietf-nvo3-geneve]. The Geneve packet format over IPv6 is defined in Section 3.2 of [I-D.ietf-nvo3-geneve]. The Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve]. Note that the outer IP header and the inner IP header may not be of the same address family, in other words, outer IPv6 header accompanied with inner IPv4 header and outer IPv4 header accompanied with inner IPv6 header are both possible.

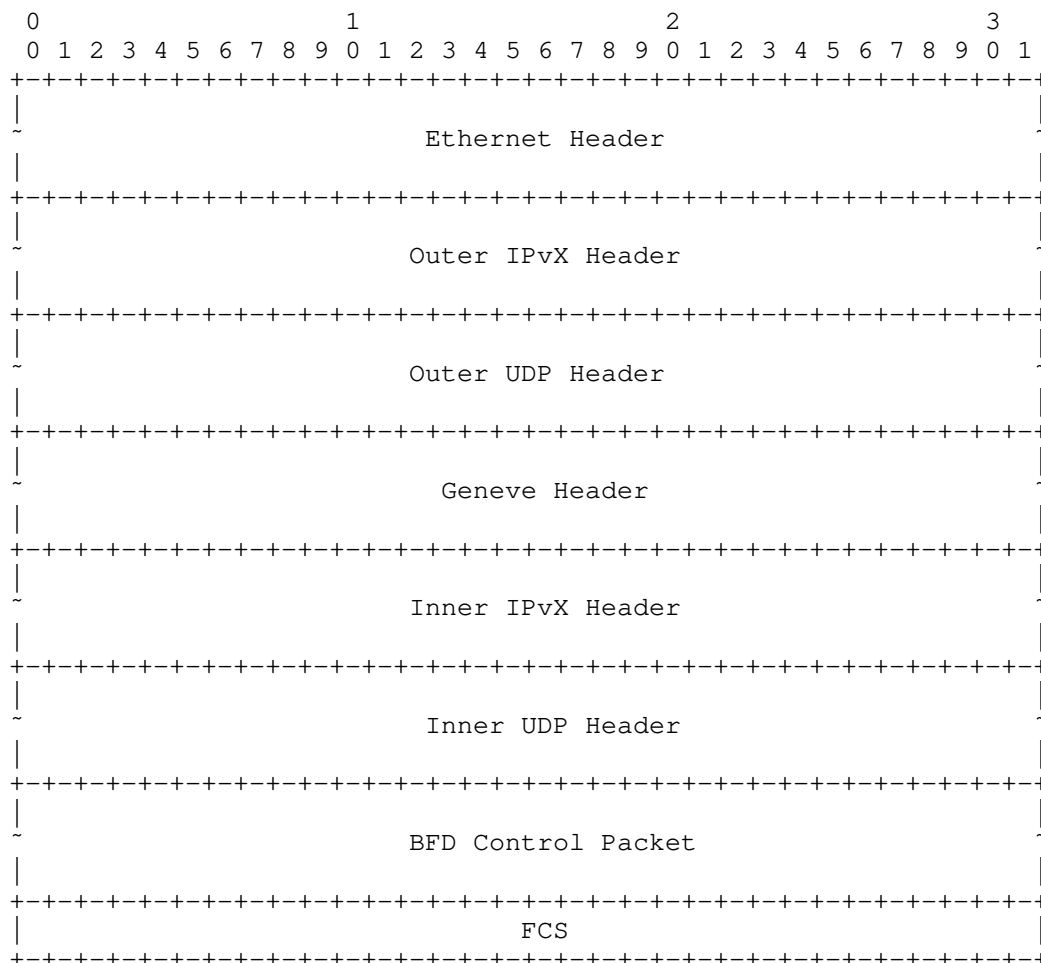


Figure 2: Geneve Encapsulation of BFD Control Packet With the Inner IP/UDP Header

The BFD packet MUST be carried inside the inner IP packet of the Geneve packet. The inner IP packet carrying the BFD Control packet has the following format:

IP header:

Source IP: IP address of a VAP of the originating NVE.

Destination IP: IP address of a VAP of the terminating NVE.

TTL or Hop Limit: MUST be set to 255 in accordance with [RFC5881].

The fields of the UDP header and the BFD Control packet are encoded as specified in [RFC5881].

When the BFD packets are encapsulated in Geneve in this way, the Geneve header defined in [I-D.ietf-nvo3-geneve] follows the value set below.

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0, which indicates there isn't any critical option.

Protocol Type field MUST be set to 0x0800 (IPv4) or 0x86DD (IPv6), depending on the address family of the inner IP packet.

Virtual Network Identifier (VNI) field SHOULD be set to the VNI number that the originating VAP is mapped to.

#### 4. Reception of BFD packet from Geneve Tunnel

Once a packet is received, the NVE MUST validate the packet as described in [I-D.ietf-nvo3-geneve].

If the Protocol Type field equals 0x6558 (Ethernet frame), and the Destination MAC of the inner Ethernet frame matches the MAC address of a VAP which is mapped to the same as received VNI, then the Destination IP, the UDP destination port and the TTL or Hop Limit of the inner IP packet MUST be validated to determine whether the received packet can be processed by BFD.

If the Protocol Type field equals 0x0800 (IPv4) or 0x86DD (IPv6), and the Destination IP of the inner IP packet matches the IP address of a VAP which is mapped to the same as received VNI, then the UDP destination port and the TTL or Hop Limit of the inner IP packet MUST be validated to determine whether the received packet can be processed by BFD.

#### 4.1. Demultiplexing of the BFD packet

In BFD over Geneve, a BFD session is originated and terminated at VAP, usually one NVE owns multiple VAPs, so multiple BFD sessions may be running between two NVEs, there needs to be a mechanism for demultiplexing received BFD packets to the proper session. Furthermore, due to the fact that [RFC8014] allows for N-to-1 mapping between VAP and VNI at one NVE, multiple BFD sessions between two NVEs for the same VNI are allowed. Also note that a BFD session can only be established between two VAPs that are mapped to the same VNI and use the same way to encapsulate data packets.

If the BFD packet is received with Your Discriminator equals to 0, for different BFD encapsulation, the procedure for demultiplexing the received BFD packets is different.

When the BFD Encapsulation With Inner Ethernet/IP/UDP Header is used, the BFD session MUST be identified using the VNI number, and the inner Ethernet/IP/UDP Header, i.e., the source MAC, the source IP, the destination MAC, the destination IP, and the source UDP port number present in the inner Ethernet/IP/UDP header.

When the BFD Encapsulation With Inner IP/UDP Header is used, the BFD session MUST be identified using the VNI number, and the inner IP/UDP header, i.e., the source IP, the destination IP, and the source UDP port number present in the inner IP/UDP header.

If the BFD packet is received with non-zero Your Discriminator, then the BFD session MUST be demultiplexed only with Your Discriminator as the key.

#### 5. Security Considerations

This document does not raise any additional security issues beyond those of the specifications referred to in the list of references.

#### 6. IANA Considerations

This document has no IANA action requested.

#### 7. Acknowledgements

The authors would like to acknowledge Reshad Rahman, Jeffrey Haas and Matthew Bocci for their guidance on this work.

The authors would like to acknowledge David Black for his explanation on the mapping relation between VAP and VNI.

## 8. References

### 8.1. Normative References

- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.
- [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>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <<https://www.rfc-editor.org/info/rfc5881>>.
- [RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <<https://www.rfc-editor.org/info/rfc7365>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [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>>.

### 8.2. Informative References

- [I-D.ietf-bfd-vxlan]  
Networks, J., Paragiri, S., Govindan, V., Mudigonda, M., and G. Mirsky, "BFD for VXLAN", draft-ietf-bfd-vxlan-13 (work in progress), July 2020.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.

#### Authors' Addresses

Xiao Min  
ZTE Corp.  
Nanjing  
China

Phone: +86 25 88013062  
Email: [xiao.min2@zte.com.cn](mailto:xiao.min2@zte.com.cn)

Greg Mirsky  
ZTE Corp.  
USA

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

Santosh Pallagatti  
VMware

Email: [santosh.pallagatti@gmail.com](mailto:santosh.pallagatti@gmail.com)

Jeff Tantsura  
Apstra

Email: [jefftant.ietf@gmail.com](mailto:jefftant.ietf@gmail.com)

NVO3 Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 5, 2020

X. Min  
G. Mirsky  
ZTE Corp.  
S. Pallagatti  
VMware  
November 2, 2019

Performance Measurement for Geneve  
draft-xiao-nvo3-pm-geneve-00

Abstract

This document describes the method to achieve Performance Measurement (PM) in point-to-point Generic Network Virtualization Encapsulation (Geneve) tunnels that form an overlay network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 5, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions Used in This Document . . . . .	2
2.1. Terminology . . . . .	2
2.2. Requirements Language . . . . .	3
3. PM Packet Transmission over Geneve Tunnel . . . . .	3
3.1. PM Encapsulation With Inner Ethernet/IP/UDP Headers . . . . .	3
3.2. PM Encapsulation With Inner IP/UDP Headers . . . . .	5
3.3. PM Encapsulation With Inner MPLS Header . . . . .	7
4. Reception of PM packet from Geneve Tunnel . . . . .	9
4.1. Demultiplexing of the PM packet . . . . .	9
5. Security Considerations . . . . .	10
6. IANA Considerations . . . . .	10
7. Acknowledgements . . . . .	10
8. Normative References . . . . .	10
Authors' Addresses . . . . .	11

## 1. Introduction

"Generic Network Virtualization Encapsulation" (Geneve) [I-D.ietf-nvo3-geneve] provides an encapsulation scheme that allows building an overlay network by decoupling the address space of the attached virtual hosts from that of the network.

This document describes the use of Packet Loss and Delay Measurement for MPLS Networks [RFC6374], as well as Simple Two-way Active Measurement Protocol [I-D.ietf-ippm-stamp], to enable measuring the performance of the path between two Geneve tunnel endpoints.

In this document, NVE (Network Virtualization Edge) represents a Geneve tunnel endpoint, TS (Tenant System) represents a physical or virtual device attached to a Geneve tunnel endpoint, and VAP (Virtual Access Point) represents the NVE side of the interface between the NVE and the TS.

## 2. Conventions Used in This Document

## 2.1. Terminology

GAL: Generic Associated Channel Label

G-ACh: Generic Associated Channel

Geneve: Generic Network Virtualization Encapsulation

MPLS: Multiprotocol Label Switching



NVE: Network Virtualization Edge

PM: Performance Measurement

STAMP: Simple Two-way Active Measurement Protocol

TS: Tenant System

VAP: Virtual Access Point

VNI: Virtual Network Identifier

## 2.2. Requirements Language

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. PM Packet Transmission over Geneve Tunnel

Analogous to what's specified in Section 3 of [I-D.xiao-nvo3-bfd-geneve], this document considers three options of PM packet encapsulation in Geneve:

- o with Ethernet and IP/UDP encapsulation;
- o with IP/UDP encapsulation;
- o with MPLS encapsulation.

### 3.1. PM Encapsulation With Inner Ethernet/IP/UDP Headers

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that an inner Ethernet header immediately follows the Geneve header, i.e., the Protocol Type equals to 0x6558 (Ethernet frame), then PM packets are encapsulated in Geneve as described below.

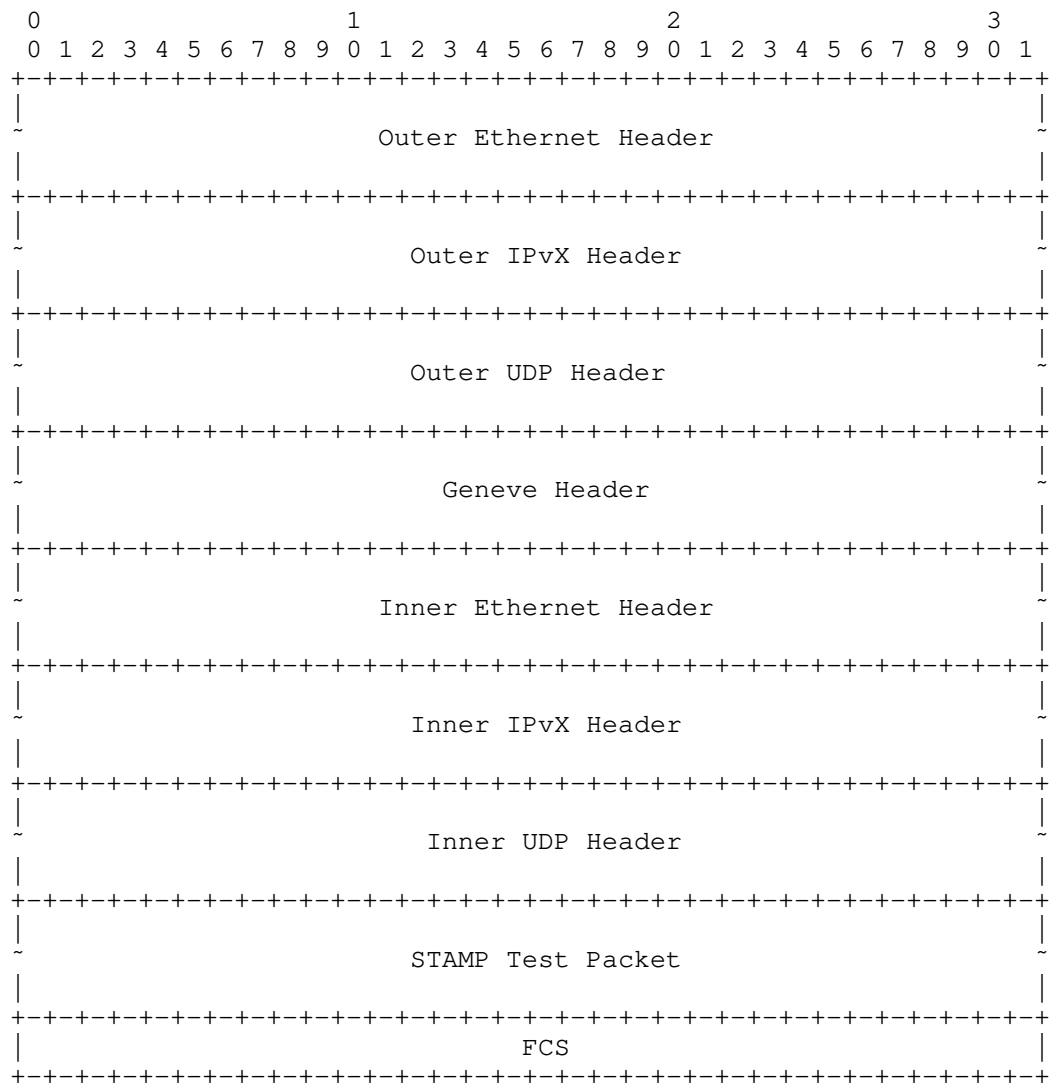


Figure 1: Geneve Encapsulation of PM Message With the Inner Ethernet/IP/UDP Header

The STAMP test packet MUST be carried inside the inner Ethernet frame of the Geneve packet, immediately after the inner IP/UDP headers. The inner Ethernet frame carrying the STAMP Test Packet has the following format:

The Ethernet header and IP header are encoded as specified in Section 4 of [I-D.ietf-bfd-vxlan].

The destination UDP port MUST be set the well-known port 862 as defined in [I-D.ietf-ippm-stamp].

The STAMP Test Packet SHOULD be unauthenticated STAMP Session-Sender test packet or unauthenticated STAMP Session-Reflector test packet. The STAMP Test Packet is encoded as specified in [I-D.ietf-ippm-stamp] and [I-D.ietf-ippm-stamp-option-tlv].

If the PM packets are encapsulated in Geneve as described above, the values in the Geneve header are set as follows:

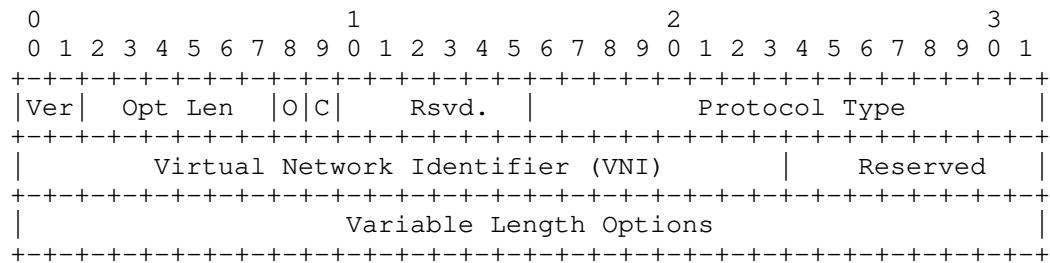


Figure 2: Geneve Header

Opt Len field MUST be set to 0 to indicate that the header does not include any variable-length options.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0.

Protocol Type field MUST be set to 0x6558 (Ethernet frame).

### 3.2. PM Encapsulation With Inner IP/UDP Headers

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that an inner IP header immediately follows the Geneve header, i.e., the Protocol Type equals to 0x0800 (IPv4) or 0x86DD (IPv6), then PM packets are encapsulated in Geneve as described below.

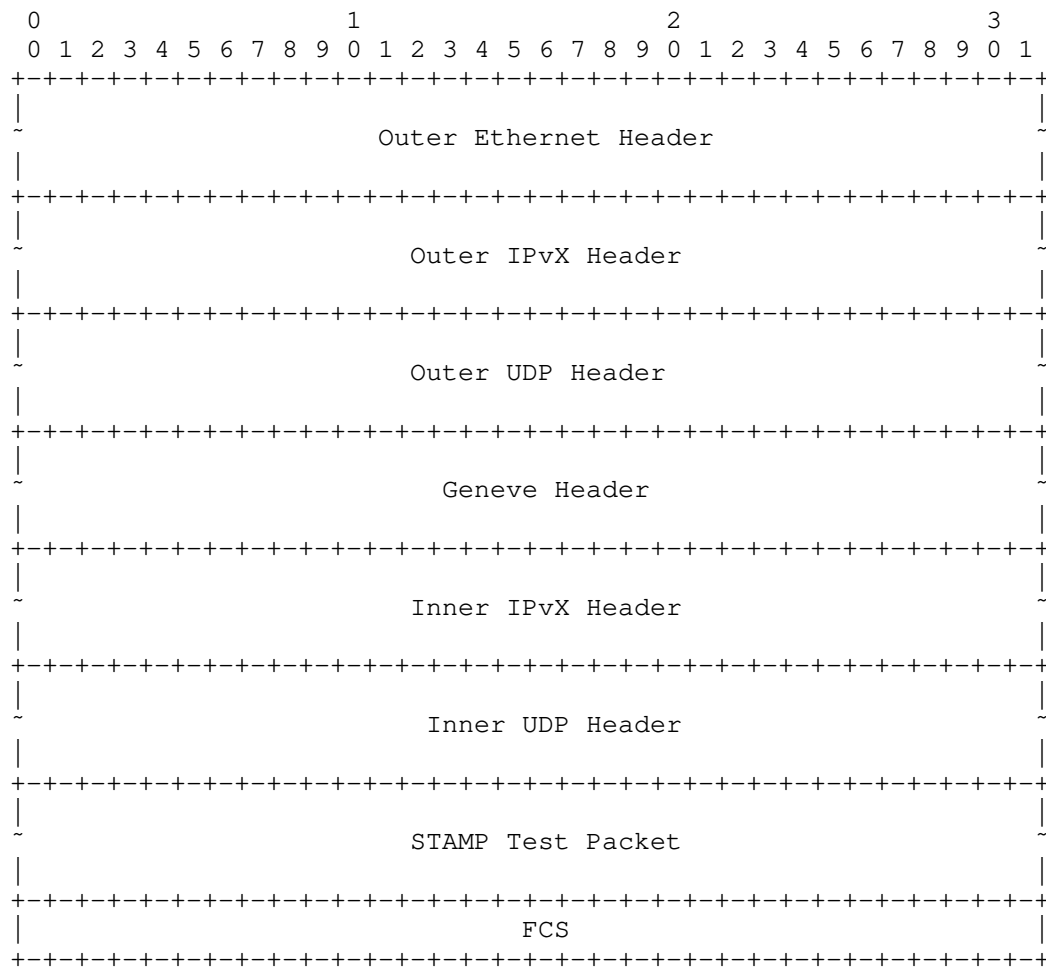


Figure 3: Geneve Encapsulation of PM Message With the Inner IP/UDP Header

A STAMP test packet MUST be carried inside the inner IP/UDP packet that immediately follows the Geneve header. The values in the inner IP packet carrying the STAMP Test Packet are as follows:

The IP header is encoded as specified in Section 3.2 of [I-D.xiao-nvo3-bfd-geneve].

The destination UDP port MUST be set the well-known port 862 as defined in [I-D.ietf-ippm-stamp].

The STAMP Test Packet SHOULD be unauthenticated STAMP Session-Sender test packet or unauthenticated STAMP Session-Reflector test packet. The STAMP Test Packet is encoded as specified in [I-D.ietf-ippm-stamp] and [I-D.ietf-ippm-stamp-option-tlv].

When the PM packets are encapsulated in Geneve in this way, the Geneve header follows the value set below.

Opt Len field MUST be set to 0 to indicate there isn't any variable-length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0.

Protocol Type field MUST be set to 0x0800 (IPv4) or 0x86DD (IPv6), depending on the address family of the inner IP packet.

### 3.3. PM Encapsulation With Inner MPLS Header

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that an MPLS label stack immediately follows the Geneve header, i.e., the Protocol Type equals to 0x8847 (MPLS) or 0x8848 (MPLS with the upstream-assigned label), then PM packets are encapsulated in Geneve, as described below.

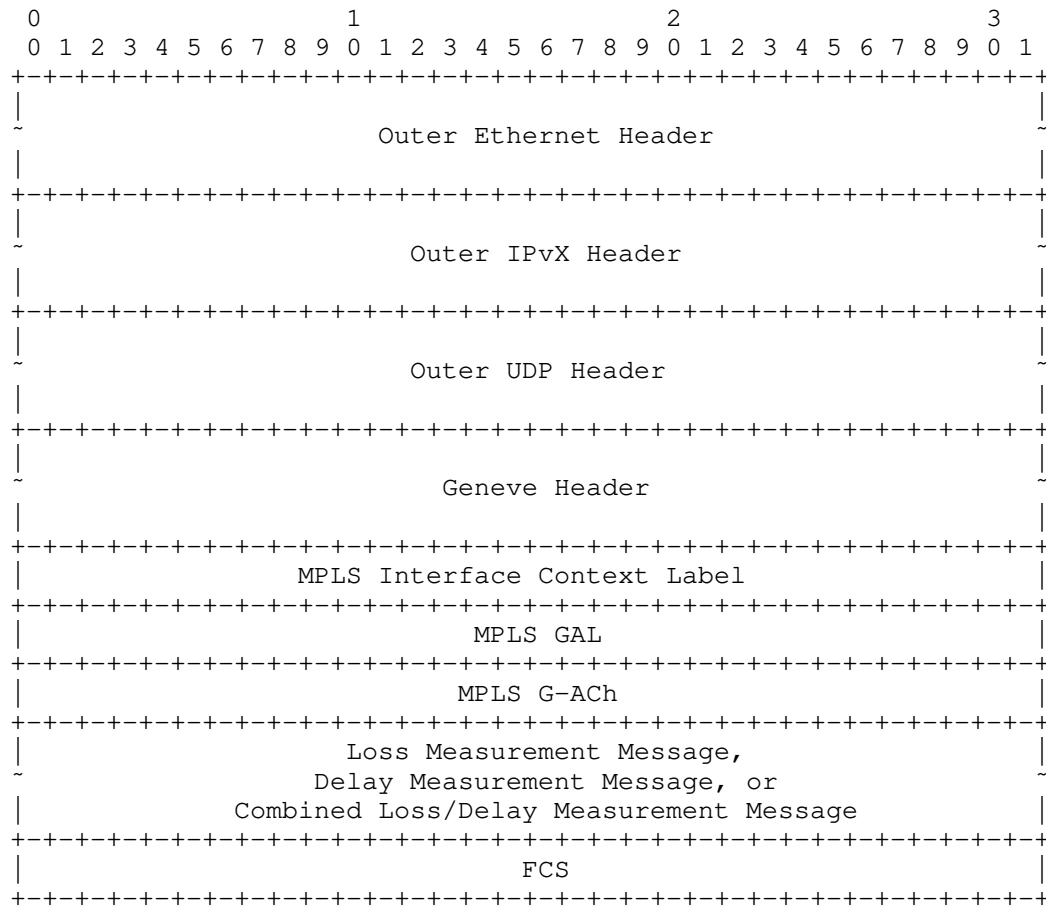


Figure 4: Geneve Encapsulation of PM Message With the Inner MPLS GAL/G-ACh

The Loss Measurement Message, Delay Measurement Message, or Combined Loss/Delay Measurement Message MUST be carried inside the inner MPLS packet that immediately follows the Geneve header. The values in the inner MPLS packet carrying the Loss Measurement Message, Delay Measurement Message, or Combined Loss/Delay Measurement Message are as follows:

The MPLS Interface Context Label and the MPLS GAL (Generic Associated Channel Label) are encoded as specified in Section 3.3 of [I-D.xiao-nvo3-bfd-geneve].

The MPLS G-ACh (Generic Associated Channel) is encoded as specified in [RFC5586], and the "Channel Type" field of MPLS G-ACh MUST be set to 0x000A, 0x000C or 0x000D requested by [RFC6374], respectively indicating "MPLS Direct Loss Measurement", "MPLS Delay Measurement" or "MPLS Direct Loss and Delay Measurement".

The Loss Measurement Message, Delay Measurement Message, and Combined Loss/Delay Measurement Message are encoded as specified in Sections 3.1 through 3.3 of [RFC6374].

When the PM packets are encapsulated in Geneve in this way, the Geneve header follows the value set below.

Opt Len field MUST be set to 0 to indicate there isn't any variable-length option.

O bit MUST be set to 1, which indicates this packet contains a control message.

C bit MUST be set to 0.

Protocol Type field MUST be set to 0x8847 (MPLS).

#### 4. Reception of PM packet from Geneve Tunnel

Once a packet is received, NVE MUST validate the packet as described in [I-D.ietf-nvo3-geneve] and Section 4 of [I-D.xiao-nvo3-bfd-geneve].

##### 4.1. Demultiplexing of the PM packet

Similar to BFD over Geneve, multiple PM sessions may be running between two NVEs, so there needs to be a mechanism for demultiplexing received PM packets to the proper session.

If the PM packet is received with Session Identifier value equals to 0, for different PM encapsulation, the procedure for demultiplexing the received PM packets is different, which would follow the procedure for a BFD packet with Your Discriminator equals to 0, as specified in Section 4.1 of [I-D.xiao-nvo3-bfd-geneve].

If the PM packet is received with a non-zero Session Identifier, then PM session MUST be demultiplexed only with Session Identifier as the key.

With respect to PM for Geneve, the use of the specific VNI would follow the principle as specified in Section 4.1 of [I-D.xiao-nvo3-bfd-geneve].

## 5. Security Considerations

This document does not raise any additional security issues beyond those of the specifications referred to in the list of normative references.

## 6. IANA Considerations

This document has no IANA action requested.

## 7. Acknowledgements

TBA.

## 8. Normative References

- [I-D.ietf-bfd-vxlan]  
Networks, J., Paragiri, S., Govindan, V., Mudigonda, M.,  
and G. Mirsky, "BFD for VXLAN", draft-ietf-bfd-vxlan-07  
(work in progress), May 2019.
- [I-D.ietf-ippm-stamp]  
Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple  
Two-way Active Measurement Protocol", draft-ietf-ippm-  
stamp-09 (work in progress), October 2019.
- [I-D.ietf-ippm-stamp-option-tlv]  
Mirsky, G., Xiao, M., Nydell, H., Foote, R., Masputra, A.,  
and E. Ruffini, "Simple Two-way Active Measurement  
Protocol Optional Extensions", draft-ietf-ippm-stamp-  
option-tlv-02 (work in progress), October 2019.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic  
Network Virtualization Encapsulation", draft-ietf-  
nvo3-geneve-14 (work in progress), September 2019.
- [I-D.xiao-nvo3-bfd-geneve]  
Xiao, M., Mirsky, G., and J. Networks, "BFD for Geneve",  
draft-xiao-nvo3-bfd-geneve-01 (work in progress), October  
2019.
- [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>>.



- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed.,  
"MPLS Generic Associated Channel", RFC 5586,  
DOI 10.17487/RFC5586, June 2009,  
<<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay  
Measurement for MPLS Networks", RFC 6374,  
DOI 10.17487/RFC6374, September 2011,  
<<https://www.rfc-editor.org/info/rfc6374>>.
- [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>>.

## Authors' Addresses

Xiao Min  
ZTE Corp.  
Nanjing  
China

Phone: +86 25 88013062  
Email: [xiao.min2@zte.com.cn](mailto:xiao.min2@zte.com.cn)

Greg Mirsky  
ZTE Corp.  
USA

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

Santosh Pallagatti  
VMware

Email: [santosh.pallagatti@gmail.com](mailto:santosh.pallagatti@gmail.com)

NVO3 Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 15 May 2022

X. Min  
ZTE Corp.  
G. Mirsky  
Ericsson  
S. Pallagatti  
VMware  
11 November 2021

Performance Measurement for Geneve  
draft-xiao-nvo3-pm-geneve-04

Abstract

This document describes the method to achieve Performance Measurement (PM) in point-to-point Generic Network Virtualization Encapsulation (Geneve) tunnels used to make up an overlay network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 May 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions Used in This Document . . . . .	2
2.1. Abbreviations . . . . .	2
2.2. Requirements Language . . . . .	3
3. PM Packet Transmission over Geneve Tunnel . . . . .	3
3.1. PM Encapsulation With Inner Ethernet/IP/UDP Header . . . . .	3
3.2. PM Encapsulation With Inner IP/UDP Header . . . . .	5
4. Reception of PM packet from Geneve Tunnel . . . . .	7
4.1. Demultiplexing of the PM packet . . . . .	7
5. Security Considerations . . . . .	7
6. IANA Considerations . . . . .	7
7. Acknowledgements . . . . .	7
8. References . . . . .	7
8.1. Normative References . . . . .	7
8.2. Informative References . . . . .	8
Authors' Addresses . . . . .	8

## 1. Introduction

"Generic Network Virtualization Encapsulation" (Geneve) [RFC8926] provides an encapsulation scheme that allows building an overlay network by decoupling the address space of the attached virtual hosts from that of the network.

This document describes the use of "Simple Two-way Active Measurement Protocol" (STAMP) [RFC8762] and "Simple Two-Way Active Measurement Protocol Optional Extensions" (STAMP Optional Extensions) [RFC8972], to enable measuring the performance of the path between two Geneve tunnel endpoints, like delay, delay variation, and packet loss.

Analogous to [I-D.ietf-nvo3-bfd-geneve], in this document, Network Virtualization Edge (NVE) represents the Geneve tunnel endpoint, Tenant System (TS) represents the physical or virtual device attached to a Geneve tunnel endpoint from the outside, Virtual Access Point (VAP) represents the NVE side of the interface between the NVE and the TS, the usage of Management Virtual Network Identifier (VNI) is described in [I-D.ietf-nvo3-geneve-oam] and outside the scope of this document.

## 2. Conventions Used in This Document

## 2.1. Abbreviations

Geneve: Generic Network Virtualization Encapsulation

NVE: Network Virtualization Edge

PM: Performance Measurement

SSID: STAMP Session Identifier

STAMP: Simple Two-way Active Measurement Protocol

TS: Tenant System

VAP: Virtual Access Point

VNI: Virtual Network Identifier

## 2.2. Requirements Language

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. PM Packet Transmission over Geneve Tunnel

PM session is originated and terminated at VAP of an NVE, selection of the PM packet encapsulation is based on how the VAP encapsulates the data packets. This document defines two formats of PM packet encapsulation in Geneve:

- \* with Ethernet and IP/UDP encapsulation;
- \* with IP/UDP encapsulation.

### 3.1. PM Encapsulation With Inner Ethernet/IP/UDP Header

If the VAP that originates the PM packets is used to encapsulate Ethernet data frames, then PM packets are encapsulated in Geneve as described below, here the PM packets are STAMP test packets.

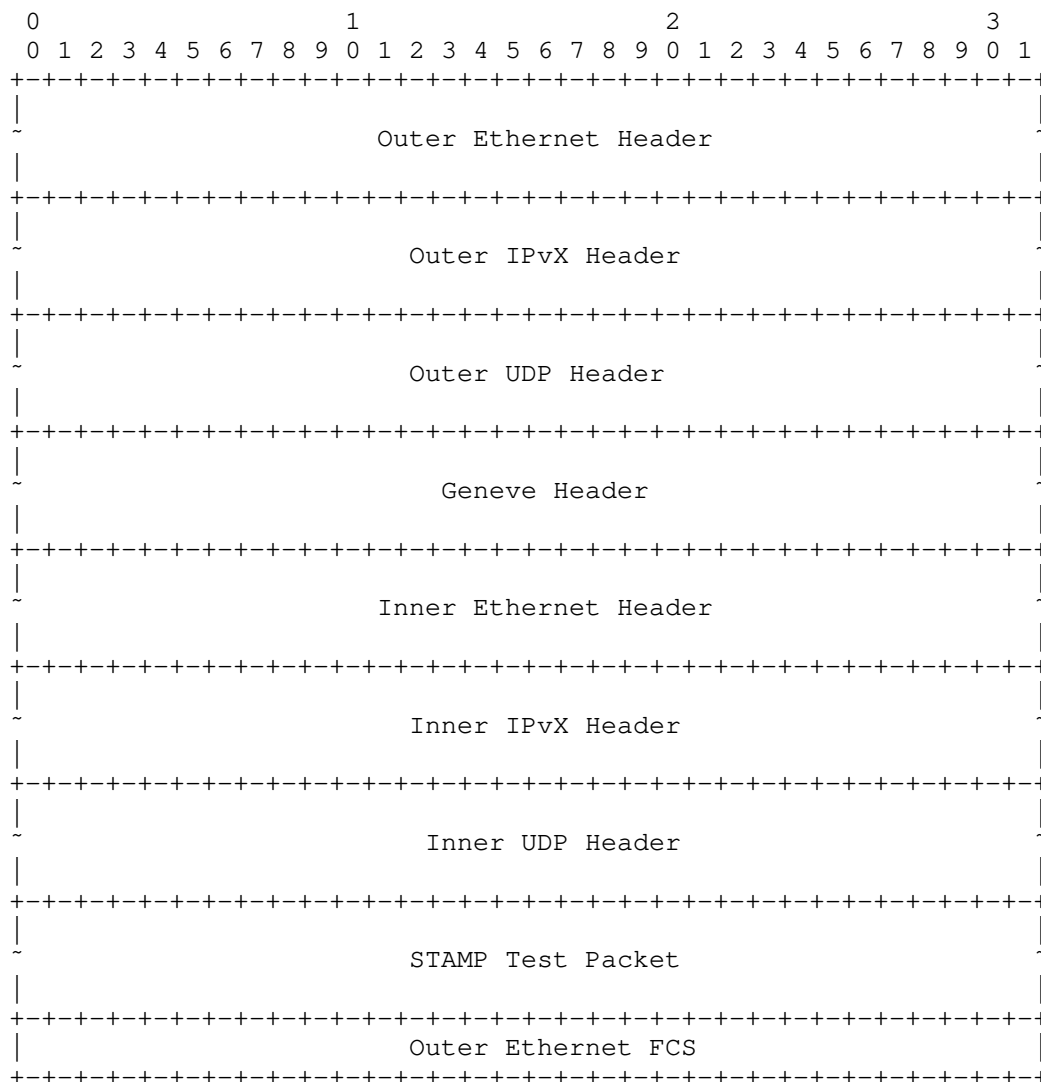


Figure 1: Geneve Encapsulation of PM Packet With the Inner Ethernet/IP/UDP Header

The STAMP test packet MUST be carried inside the inner Ethernet frame of the Geneve packet, immediately after the inner Ethernet/IP/UDP header. The inner Ethernet frame carrying the STAMP test packet has the following format:

The Ethernet header and IP header are encoded as defined in Section 3.1 of [I-D.ietf-nvo3-bfd-geneve].

The destination UDP port MUST follow the STAMP UDP port usage defined in Section 4.1 of [RFC8762].

The STAMP test packet can be STAMP Session-Sender test packet or STAMP Session-Reflector test packet. The STAMP test packet is encoded as specified in [RFC8762] and [RFC8972].

When the PM packets are encapsulated in Geneve in this way, the values in the Geneve header are set as specified in Section 3.1 of [I-D.ietf-nvo3-bfd-geneve].

### 3.2. PM Encapsulation With Inner IP/UDP Header

If the VAP that originates the PM packets is used to encapsulate IP data packets, then PM packets are encapsulated in Geneve as described below, here the PM packets are STAMP test packets.

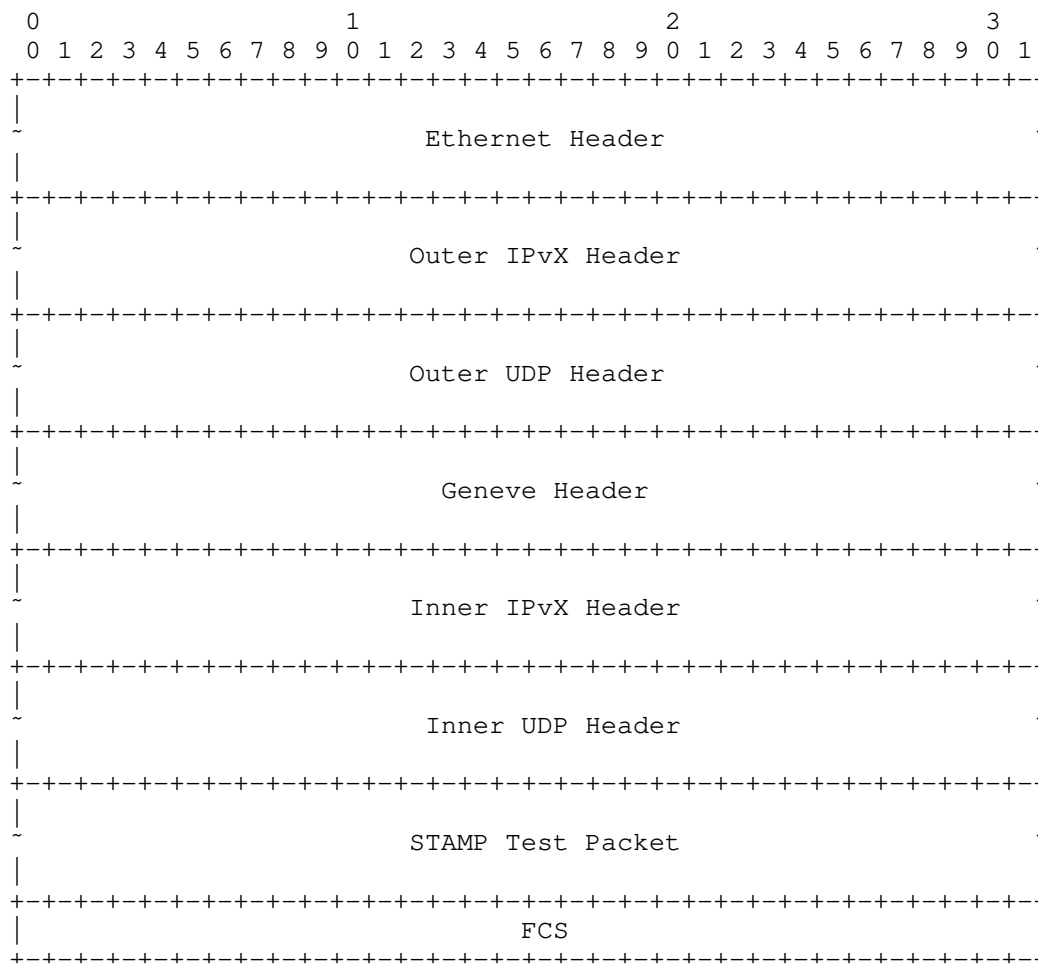


Figure 2: Geneve Encapsulation of PM Packet With the Inner IP/UDP Header

The STAMP test packet MUST be carried inside the inner IP packet that immediately follows the Geneve header. The inner IP packet carrying the STAMP test packet has the following format:

The IP header is encoded as defined in Section 3.2 of [I-D.ietf-nvo3-bfd-geneve].

The destination UDP port MUST follow the STAMP UDP port usage defined in Section 4.1 of [RFC8762].

The STAMP test packet can be STAMP Session-Sender test packet or STAMP Session-Reflector test packet. The STAMP test packet is encoded as specified in [RFC8762] and [RFC8972].

When the PM packets are encapsulated in Geneve in this way, the values in the Geneve header are set as specified in Section 3.2 of [I-D.ietf-nvo3-bfd-geneve].

#### 4. Reception of PM packet from Geneve Tunnel

Once a packet is received, the NVE MUST validate the packet as specified in Section 4 of [I-D.ietf-nvo3-bfd-geneve], except that the received STAMP test packet would be processed by STAMP Session-Sender or STAMP Session-Reflector, instead of BFD.

##### 4.1. Demultiplexing of the PM packet

Analogous to BFD over Geneve, multiple STAMP sessions for the same VNI may be running between two NVEs, so there needs to be a mechanism for demultiplexing received STAMP test packets to the proper session.

If the STAMP test packet is received with STAMP Session Identifier (SSID) equals to 0, the procedure for demultiplexing the received STAMP test packets would follow the procedure for demultiplexing the received BFD packets with Your Discriminator equals to 0, which is specified in Section 4.1 of [I-D.ietf-nvo3-bfd-geneve].

If the STAMP test packet is received with a non-zero SSID, then the STAMP session MUST be demultiplexed only with SSID as the key.

#### 5. Security Considerations

This document does not raise any additional security issues beyond those of the specifications referred to in the list of references.

#### 6. IANA Considerations

This document has no IANA action requested.

#### 7. Acknowledgements

TBA.

#### 8. References

##### 8.1. Normative References



[I-D.ietf-nvo3-bfd-geneve]

Min, X., Mirsky, G., Pallagatti, S., Tantsura, J., and S. Aldrin, "BFD for Geneve", Work in Progress, Internet-Draft, draft-ietf-nvo3-bfd-geneve-05, 11 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-nvo3-bfd-geneve-05.txt>>.

[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>>.

[RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <<https://www.rfc-editor.org/info/rfc8762>>.

[RFC8926] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed., "Geneve: Generic Network Virtualization Encapsulation", RFC 8926, DOI 10.17487/RFC8926, November 2020, <<https://www.rfc-editor.org/info/rfc8926>>.

[RFC8972] Mirsky, G., Min, X., Nydell, H., Foote, R., Masputra, A., and E. Ruffini, "Simple Two-Way Active Measurement Protocol Optional Extensions", RFC 8972, DOI 10.17487/RFC8972, January 2021, <<https://www.rfc-editor.org/info/rfc8972>>.

## 8.2. Informative References

[I-D.ietf-nvo3-geneve-oam]

Mirsky, G., Boutros, S., Black, D., and S. Pallagatti, "OAM for use in GENEVE", Work in Progress, Internet-Draft, draft-ietf-nvo3-geneve-oam-03, 8 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-nvo3-geneve-oam-03.txt>>.

## Authors' Addresses

Xiao Min  
ZTE Corp.  
Nanjing  
China

Phone: +86 25 88013062  
Email: xiao.min2@zte.com.cn

Greg Mirsky  
Ericsson  
United States of America  
  
Email: gregimirsky@gmail.com

Santosh Pallagatti  
VMware  
  
Email: santosh.pallagatti@gmail.com