Network Working Group — F. Maino, Ed.
Internet-Draft — Cisco Systems
Intended status: Informational — L. Kreeger, Ed.
Expires: April 16, 2020 — 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

Copyright Notice

Table of Contents

1. Introduction

Network Working Group — F. Maino, Ed.
Internet-Draft — Cisco Systems
Intended status: Informational — L. Kreeger, Ed.
Expires: June 7, 2020 — Arrcus
U. Elzur, Ed.
Intel
December 5, 2019

Generic Protocol Extension for VXLAN
draft-ietf-nvo3-vxlan-gpe-09

Abstract

This draft 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.

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

Copyright Notice

Table of Contents

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	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
skipping to change at *page 3, line 20*	**skipping to change at *page 3, line 30***
identifiers for this purpose. VXLAN GPE is intended to extend the existing VXLAN protocol to provide protocol typing, OAM, and versioning capabilities.	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.	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.	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	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.	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.	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.	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.
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.	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.
`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                   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`
skipping to change at *page 6, line 12*	**skipping to change at *page 6, line 7***
Flag bit 5 was chosen as the P bit because this flag bit is currently reserved in VXLAN.	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.	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:	This draft defines the following Next Protocol values:
0x1 : IPv4	0x01 : 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:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |   Reserved    | Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                    Protocol Specific Fields                   ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

0x02 : IPv6

0x03 : Ethernet

0x04 : Network Service Header [RFC8300]

0x05 to 0x7F: Unassigned.

0x80 to 0xFF: Unassigned (shim headers).

Next protocol values from 0x80 to 0xFF 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 Lenght, and a Next Protocol field. When a shim header is used with other protocols identified by next protocol values from 0x0 to 0x7F, the shim header MUST come before the further protocol, and the next header of the shim will indicate which protocol follows the shim header.

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:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |   Reserved    | Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                    Protocol Specific Fields                   ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

skipping to change at *page 8, line 34* | **skipping to change at *page 8, line 20***

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.

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.

skipping to change at *page 12, line 30* | **skipping to change at *page 12, line 7***

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

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.

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 in Section 5.3.1 are met.

5.3.1. UDP Zero Checksum Handling with IPv6

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

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.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |    NP = IPv4  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
```

skipping to change at *page 14, line 17*

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |NP = Ethernet  |
```

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.

7. VXLAN GPE Examples

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

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |    NP = IPv4  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
```

skipping to change at *page 17, line 5*

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|0|0|I|1|R|0|       Reserved                |NP = Ethernet  |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Original Ethernet Frame                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

skipping to change at *page 15, line 44*

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)

skipping to change at *page 17, line 4*

Bit 4 - Instance ID (I bit)

Bit 5 - Next Protocol (P bit)


```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Original Ethernet Frame                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

skipping to change at *page 18, line 44*

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.

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..0x7F	Unassigned	
0x80..0xFF	Unassigned (shim headers)	

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)

skipping to change at *page 19, line 48*

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]

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.

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>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <https://www.rfc-editor.org/info/rfc2983>.

[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., 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

Larry Kreeger (editor)
Arrcus

Email: lkreeger@gmail.com

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-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

Larry Kreeger (editor)
Arrcus

Email: lkreeger@gmail.com

End of changes. 54 change blocks.

158 lines changed or deleted	*321 lines changed or added*