INTERNET-DRAFT
Intended Status: Informational

Sami Boutros(Ed.)
VMware

Ignas Bagdonas
Equinix

Sam Aldrin
Google

Matthew Bocci
Nokia

Uri Elzur
Ilango Ganga
Intel

Pankaj Garg
Microsoft

Rajeev Manur
Broadcom

Tal Mizrahi
Marvell

David Mozes
Mellanox

Erik Nordmark
Arista Networks

Michael Smith
Cisco

Expires: September 13, 2017

March 12, 2017

NVO3 Encapsulation Considerations
draft-dt-nvo3-encap-01

Abstract

As communicated by WG Chairs, the IETF NVO3 chairs and Routing Area director have chartered a design team to take forward the encapsulation discussion and see if there is potential to design a common encapsulation that addresses the various technical concerns.

There are implications of different encapsulations in real environments consisting of both software and hardware implementations and spanning multiple data centers. For example, OAM functions such as path MTU discovery become challenging with multiple encapsulations along the data path.

The design team recommend Geneve with few modifications as the common encapsulation, more details are described in section 7.

Status of this Memo

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

Copyright and License Notice

Table of Contents

1. Problem Statement

As communicated by WG Chairs, the NVO3 WG charter states that it may produce requirements for network virtualization data planes based on encapsulation of virtual network traffic over an IP-based underlay data plane. Such requirements should consider OAM and security. Based on these requirements the WG will select, extend, and/or develop one or more data plane encapsulation format(s).

This has led to drafts describing three encapsulations being adopted by the working group:

- draft-ietf-nvo3-geneve-03

- draft-ietf-nvo3-gue-04

- draft-ietf-nvo3-vxlan-gpe-02

Discussion on the list and in face-to-face meetings has identified a number of technical problems with each of these encapsulations. Furthermore, there was clear consensus at the IETF meeting in Berlin that it is undesirable for the working group to progress more than one data plane encapsulation. Although consensus could not be reached on the list, the overall consensus was for a single encapsulation (RFC2418, Section 3.3). Nonetheless there has been resistance to converging on a single encapsulation format.

2. Design Team Goals

As communicated by WG Chairs, the design team should take one of the proposed encapsulations and enhance it to address the technical concerns. Backwards compatibility with the chosen encapsulation and the simple evolution of deployed networks as well as applicability to all locations in the NVO3 architecture are goals. The DT should specifically avoid a design that is burdensome on hardware implementations, but should allow future extensibility. The chosen design should also operate well with ICMP and in ECMP environments. If further extensibility is required, then it should be done in such a manner that it does not require the consent of an entity outside of the IETF.

3. 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].

4. Abbreviations

NVO3 Network Virtualization Overlays over Layer 3

OAM Operations, Administration, and Maintenance

TLV Type, Length, and Value

VNI Virtual Network Identifier

NVE Network Virtualization Edge

NVA Network Virtualization Authority

NIC Network interface card

Transit device Underlay network devices between NVE(s).

5. Issues with current Encapsulations

As summarized by WG Chairs.

5.1 Geneve

- Can’t be implemented cost-effectively in all use cases because variable length header and order of the TLVs makes is costly (in terms of number of gates) to implement in hardware

- Fork-lift upgrade from widely deployed VXLAN (no backwards compatibility mechanisms)

- Header doesn’t fit into largest commonly available parse buffer (256 bytes in NIC). Cannot justify doubling buffer size unless it is mandatory for hardware to process additional option fields.

5.2 GUE

- There were a significant number of objections related to the complexity of implementation in hardware, similar to those noted for Geneve above.

- In addition, there were concerns raised that GUE does not support a sufficient number of extensions due to its reliance on a limited flags field, which is already almost 45% allocated.

5.3 VXLAN-GPE

- GPE is not day-1 backwards compatible with VXLAN. Although the

frame format is similar, it uses a different UDP port, so would require changes to existing implementations even if the rest of the GPE frame is the same.

- GPE is insufficiently extensible. Numerous extensions and options have been designed for GUE and Geneve. Note that these have not yet been validated by the WG.

- Security e.g. of the VNI has not been addressed by GPE. Although a shim header could be used for security and other extensions, this has not been defined yet and its implications on offloading in NICs are not understood.

6. Common Encapsulation Considerations

6.1 Current Encapsulations

Appendix A includes a detailed comparison between the three proposed encapsulations. The comparison indicates several common properties, but also three major differences among the encapsulations:

- Extensibility: Geneve and GUE were defined with built-in extensibility, while VXLAN-GPE is not inherently extensible. Note that any of the three encapsulations can be extended using the Network Service Header (NSH).

- Extension method: Geneve is extensible using Type/Length/Value (TLV) fields, while GUE uses a small set of possible extensions, and a set of flags that indicate which extension is present.

- Length field: Geneve and GUE include a Length field, indicating the length of the encapsulation header, while VXLAN-GPE does not include such a field.

6.2 Useful Extensions Use cases

Non vendor specific TLV MUST follow the standardization process. The following use cases for extensions shows that there is a strong requirement to support variable length extensions with possible different subtypes.

6.2.1. Telemetry extensions.

In several scenarios it is beneficial to make information about the path a packet took through the network or through a network device as well as associated telemetry information available to the operator.

This includes not only tasks like debugging, troubleshooting, as well as network planning and network optimization but also policy or service level agreement compliance checks.

Packet scheduling algorithms, especially for balancing traffic across equal cost paths or links, often leverage information contained within the packet, such as protocol number, IP-address or MAC-address. Probe packets would thus either need to be sent from the exact same endpoints with the exact same parameters, or probe packets would need to be artificially constructed as "fake" packets and inserted along the path. Both approaches are often not feasible from an operational perspective, be it that access to the end-system is not feasible, or that the diversity of parameters and associated probe packets to be created is simply too large. An in-bound telemetry mechanism in extensions is an alternative in those cases.

6.2.2. Security/Integrity extensions

Since the currently proposed NVO3 encapsulations do not protect their headers a single bit corruption in the VNI field could deliver a packet to the wrong tenant. Extensions are needed to use any sophisticated security.

The possibility of VNI spoofing with an NVO3 protocol is exacerbated by the use of UDP. Systems typically have no restrictions on applications being able to send to any UDP port so an unprivileged application can trivially spoof for instance, VXLAN packets, including using arbitrary VNIs.

One can envision HMAC-like support in some NVO3 extension to authenticate the header and the outer IP addresses, thereby preventing attackers from injecting packets with spoofed VNIs.

An other aspect of security is payload security. Essentially this is to make packets that look like IP|UDP|NVO3 Encap|DTLS Extension|payload. This is nice since we still have the UDP header for ECMP, the NVO3 header is in plain text so it can by read by network elements, and different security or other payload transforms can be supported on a single UDP port (we don’t need a separate UDP for DTLS).

6.2.3. Group Base Policy

Another use case would be to carry the Group Based Policy (GBP) source group information within a NVO3 header extension in a similar manner as has been implemented for VXLAN [VXLAN-GBP]. This allows various forms of policy such as access control and QoS to be applied

between abstract groups rather than coupled to specific endpoint addresses.

6.3 Hardware Considerations

Hardware restrictions should be taken into consideration along with future hardware enhancements that may provide more flexible metadata processing. However, the set of options that need to and will be implemented in hardware will be a subset of what is implemented in software, since software NVEs are likely to grow features, and hence option support, at a more rapid rate.

We note that it is hard to predict which options will be implemented in which piece of hardware and when. That depends on whether the hardware will be in the form of a NIC providing increasing offload capabilities to software NVEs, or a switch chip being used as an NVE gateway towards non-NVO3 parts of the network, or even an transit devices that participates in the NVO3 dataplane e.g. for OAM purposes.

A result of this is that it doesn’t look useful to prescribe some order of the option so that the ones that are likely to be implemented in hardware come first; we can’t decide such an order when we define the options, however a control plane can enforce such order for some hardware implementations.

We do know that hardware needs to initially be able to efficiently skip over the NVO3 header to find the inner payload. That is needed for both NICs doing e.g. TCP offload and transit devices and NVEs applying policy/ACLs to the inner payload.

6.4 Extension Size

Extension header length has a significant impact to hardware and software implementations. A total header length that is too small will unnecessarily constrained software flexibility. A total header length that is too large will place a nontrivial cost on hardware implementations. Thus, the design team recommends that there be a minimum and maximum total extension header length selected. The maximum total header length is determined by the bits allocated for the total extension header length field. The risk with this approach is that it may be difficult to extend the total header size in the future. The minimum total header length is determined by a requirement in the specifications that all implementations must meet. The risk with this approach is that all implementations will only implement the minimum total header length which would then become the de facto maximum total header length. The recommended minimum total header length is 64 bytes.

Single Extension size should always be 4 bytes aligned.

The maximum length of a single option should be large enough to meet the different extension use case requirements e.g. in-band telemetry and future use.

6.5 Extension Ordering

In order to support hardware nodes at the tunnel endpoint or at the transit that can process one or few extensions TLVs in TCAM. A control plane in such a deployment can signal a capability to ensure a specific TLV will always appear in a specific order for example the first one in the packet.

The order of the TLVs should be HW friendly for both the sender and the receiver and possibly the transit node too.

A transit node may need to process some extensions like telemetry and/or OAM inband extensions.

6.6 TLV vs Bit Fields

If there is a well-known initial set of options that are likely to be implemented in software and in hardware, it can be efficient to use the bit-field approach as in GUE. However, as described in section 6.3, if options are added over time and different subsets of options are likely to be implemented in different pieces of hardware, then it would be hard for the IETF to specify which options should get the early bit fields. TLVs are a lot more flexible, which avoids the need to determine the relative importance different options. However, general TLV of arbitrary order, size, and repetition of the same order is difficult to implement in hardware. A middle ground is to use TLV with restrictions on the size and alignment, observing that individual TLVs can have a fixed length, and support in the control plane such that an NVE will only receive options that to needs and implements. The control plane approach can potentially be used to control the order of the TLVs sent to a particular NVE. Note that transit devices are not likely to participate in the control plane hence to the extent that they need to participate in option processing they need more effort, But transit devices would have issues with future GUE bits being defined for future options as well.

A benefit of TLVs from a HW perspective is that they are self describing i.e., all the information is in the TLV. In a Bit fields approach the hardware needs to look up the bit to determine the length of the data associated with the bit through some separate

table, which would add hardware complexity.

There are use cases where multiple modules of software are running on NVE. This can be modules such as a diagnostic module by one vendor that does packet sampling and another module from a different vendor that does a firewall. Using a TLV format, it is easier to have different software modules process different TLVs, which could be standard extensions or vendor specific extensions defined by the different vendors, without conflicting with each other. This can help with hardware modularity as well.

6.7 Control Plane Considerations

Given that we want to allow large flexibility and extensibility for e.g. software NVEs, yet be able to support key extensions in less flexible e.g. hardware NVEs, it is useful to consider the control plane. By control plane in this context we mean both protocols such as EVPN and others, and also deployment specific configuration.

If each NVE can express in the control plane that they only care about particular extensions (could be a single extension, or a few), and the source NVEs only include requested extensions in the NVO3 packets, then the target NVE can both use a simpler parser (e.g., a TCAM might be usable to look for a single NVO3 extension) and the depth of the inner payload in the NVO3 packet will be minimized. Furthermore, if the target NVE cares about a few extensions and can express in the control plane the desired order of those extensions in the NVO3 packets, then it can provide useful functionality with minimal hardware requirements.

Note that transit devices that are not aware of the NVO3 extensions somewhat benefit from such an approach, since the inner payload is less deep in the packet if no extraneous extensions are included in the packet. However, in general a transit device is not likely to participate in the NVO3 control plane. (However, configuration mechanisms can take into account limitations of the transit devices used in particular deployments.)

Note that in this approach different NVEs could desire different (sets of) extensions, which means that the source NVE needs to be able to place different sets of extensions in different NVO3 packets, and perhaps in different order. It also assumes that underlay multicast or replication servers are not used together with NVO3 extensions.

There is a need to consider mandatory extensions versus optional extensions. Mandatory extensions require the receiver to drop the packet if the extension is unknown. A control plane mechanism can

prevent the need for dropping unknown extensions, since they would not be included to targets that do not support them.

The control planes defined today need to add the ability to describe the different encapsulations. Thus perhaps EVPN, and any other control plane protocol that the IETF defines, should have a way to enumerate the supported NVO3 extensions and their order.

6.8 Split NVE

If the working group sees a need for having the hosts send and receive options in a split NVE case, this is possible using any of the existing extensible encapsulations (Geneve, GUE, GPE+NSH) by defining a way to carry those over other transports. NSH can already be used over different transports.

If we need to do this with other encapsulations it can be done by defining an Ether type for other encapsulations so that it can be carried over Ethernet and 802.1Q.

If we need to carry other encapsulations over MPLS, it would require an EVPN control plane to signal that other encapsulation header + options will be present in front of the L2 packet. The VNI can be ignored in the header, and the MPLS label will be the one used to identify the EVPN L2 instance.

6.9 Larger VNI Considerations

We discussed whether we should make VNI 32-bits or larger. The benefit of 24-bit VNI would be to avoid unnecessary changes with existing proposals and implementations that are almost all, if not all, are using 24-bit VNI. If we need a larger VNI, an extension can be used to support that.

7. Design team recommendations

We concluded that Geneve is most suitable as a starting point for proposed standard for network virtualization, for the following reasons:

1. We studied whether VNI should be in base header or in extensions and whether it should be 24-bit or 32-bit. The design team agreed that VNI is critical information for network virtualization and MUST be present in all packets. Design team also agreed that 24-bit VNI

matches the existing widely used encapsulation format i.e. VxLAN and NVGRE and hence more suitable to use going forward.

2. Geneve has the total options length that allow skipping over the options for NIC offload operations, and will allow transit devices to view flow information in the inner payload.

3. We considered the option of using NSH with VxLAN-GPE but given that NSH is targeted at service chaining and contains service chaining information, it is less suitable for the network virtualization use case. The other downside for VxLAN-GPE was lack of header length in VxLAN-GPE and hence makes skipping over the headers to process inner payload more difficult. Total Option Length is present in Geneve. It is not possible to skip any options in the middle with VxLAN-GPE. In principle a split between a base header and a header with options is interesting (whether that options header is NSH or some new header without ties to a service path). We explored whether it would make sense to either use NSH for this, or define a new NVO3 options header. However, we observed that this makes it slightly harder to find the inner payload since the length field is not in the NVO3 header itself. Thus one more field would have to be extracted to compute the start of the inner payload. Also, if the experience with IPv6 extension headers is a guidance, there would be a risk that key pieces of hardware might not implement the options header, resulting in future calls to deprecate its use. Making the options part of the base NVO3 header has less of those issues. Even though the implementation of any particular option can not be predicted ahead of time, the option mechanism and ability to skip the options is likely to be broadly implemented.

4. We compared the TLV vs Bit-fields style extension and it was deemed that parsing both TLV and bit-fields is expensive and while bit-fields may be simpler to parse, it is also more restrictive and requires guessing which extensions will be widely implemented so they can get early bit assignments for efficiency, as well Bit-fields are not flexible enough to address the requirement of variable length and different subtypes of the same option. While TLV are more flexible, a control plane can restrict the number of option TLVs as well the order and size of the TLVs to make it simpler for a dataplane implementation to handle.

5. We briefly discussed multi-vendor NVE case, and the need to allow vendors to put their own extensions in the NVE header. This is possible with TLVs.

6. We also agreed that the C bit in Geneve is helpful to allow receiver NVE to easily decide whether to process options or not. For example a UUID based packet trace and how an optional extension such

as that can be ignored by receiver NVE and thus make it easy for NVE to skip over the options. Thus the C-bit remains as defined in Geneve.

7. There are already some extensions that are being discussed (see section 6.2) of varying sizes, by using Geneve option it is possible to get in band parameters like: switch id, ingress port, egress port, internal delay, and queue in telemetry defined extension TLV from switches. It is also possible to add Security extension TLVs like HMAC and DTLS to authenticate the Geneve packet header and secure the Geneve packet payload by software or hardware tunnel endpoints. As well, a Group Based Policy extension TLV can be carried.

There seems to be interest to standardize some well known secure option TLVs to secure the header and payload to guarantee encapsulation header integrity and tenant data privacy. The design team recommends that the working group consider standardizing such option(s).

We recommend the following enhancements to Geneve to make it more suitable to hardware and yet provide the flexibility for software:

We would propose a text such as, while TLV are more flexible, a control plane can restrict the number of option TLVs as well the order and size of the TLVs to make it simpler for a data plane implementation in software or hardware to handle. For example, there may be some critical information such as secure hash that must be processed in certain order at lowest latency.

A control plane can negotiate a subset of option TLVs and certain TLV ordering, as well can limit the total number of option TLVs present in the packet, for example, to allow hardware capable of processing fewer options. Hence, the control planes need to have the ability to describe the supported TLVs subset and their order.

The Geneve draft could specify that the subset and order of option TLVs should be configurable for each remote NVE in the absence of a protocol control plane.

8. Acknowledgements

Tom Herbert provided the motivation for the Security/Integrity extension.

9. Security Considerations

This document does not introduce any additional security constraints.

10. References

10.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2 Informative References

[Geneve] Generic Network Virtualization Encapsulation [I-D.ietf-nvo3-geneve]
[GUE] Generic UDP Encapsulation [I-D.ietf-nvo3-gue]
[NSH] Network Service Header [I-D.ietf-sfc-nsh]
[VXLAN-GPE] Virtual eXtensible Local Area Network - Generic Protocol Extension [I-D.ietf-nvo3-vxlan-gpe]

[VXLAN-GBP] VXLAN Group Policy Option - [I-D.draft-smith-vxlan-group-policy-03]

11. Appendix A

11.1. Overview

This section presents a comparison of the three NVO3 encapsulation proposals, Geneve, GUE, and VXLAN-GPE. The three encapsulations use an outer UDP/IP transport. Geneve and VXLAN-GPE use an 8-octet header, while GUE uses a 4-octet header. In addition to the base header, optional extensions may be included in the encapsulation, as discussed in Section 3.2 below.

11.2. Extensibility

11.2.1. Native Extensibility Support

The Geneve and GUE encapsulations both enable optional headers to be incorporated at the end of the base encapsulation header.

VXLAN-GPE does not provide native support for header extensions. However, as discussed in [I-D.ietf-nvo3-vxlan-gpe], extensibility can be attained to some extent if the Network Service Header (NSH) [I-D.ietf-sfc-nsh] is used immediately following the VXLAN-GPE header. NSH supports either a fixed-size extension (MD Type 1), or a variable-size TLV-based extension (MD Type 2). It should be noted

that NSH-over-VXLAN-GPE implies an additional overhead of the 8-octets NSH header, in addition to the VXLAN-GPE header.

11.2.2. Extension Parsing

The Geneve Variable Length Options are defined as Type/Length/Value(TLV) extensions. Similarly, VXLAN-GPE, when using NSH, can include NSH TLV-based extensions. In contrast, GUE defines a small set of possible extension fields (proposed in [I-D.herbert-gue-extensions]), and a set of flags in the GUE header that indicate for each extension type whether it is present or not.

TLV-based extensions, as defined in Geneve, provide the flexibility for a large number of possible extension types. Similar behavior can be supported in NSH-over-VXLAN-GPE when using MD Type 2. The flag-based approach taken in GUE strives to simplify implementations by defining a small number of possible extensions, used in a fixed order.

The Geneve and GUE headers both include a length field, defining the total length of the encapsulation, including the optional extensions.

The length field simplifies the parsing of transit devices that skip the encapsulation header without parsing its extensions.

11.2.3. Critical Extensions

The Geneve encapsulation header includes the ’C’ field, which indicates whether the current Geneve header includes critical options, which must be parsed by the tunnel endpoint. If the endpoint is not able to process the critical option, the packet is discarded.

11.2.4. Maximal Header Length

The maximal header length in Geneve, including options, is 260 octets. GUE defines the maximal header to be 128 octets. VXLAN-GPE uses a fixed-length header of 8 octets, unless NSH-over-VXLAN-GPE is used, yielding an encapsulation header of up to 264 octets.

11.3. Encapsulation Header

11.3.1. Virtual Network Identifier (VNI)

The Geneve and VXLAN-GPE headers both include a 24-bit VNI field. GUE, on the other hand, enables the use of a 32-bit field called VNID; this field is not included in the GUE header, but was defined

as an optional extension in [I-D.herbert-gue-extensions].

The VXLAN-GPE header includes the 'I' bit, indicating that the VNI field is valid in the current header. A similar indicator is defined as a flag in the GUE header [I-D.herbert-gue-extensions].

11.3.2. Next Protocol

The three encapsulation headers include a field that specifies the type of the next protocol header, which resides after the NVO3 encapsulation header. The Geneve header includes a 16-bit field that uses the IEEE Ethertype convention. GUE uses an 8-bit field, which uses the IANA Internet protocol numbering. The VXLAN-GPE header incorporates an 8-bit Next Protocol field, using a VXLAN-GPE-specific registry, defined in [I-D.ietf-nvo3-vxlan-gpe].

The VXLAN-GPE header also includes the 'P' bit, which explicitly indicates whether the Next Protocol field is present in the current header.

11.3.3. Other Header Fields

The OAM bit, which is defined in Geneve and in VXLAN-GPE, indicates whether the current packet is an OAM packet. The GUE header includes a similar field, but uses different terminology; the GUE 'C-bit' specifies whether the current packet is a control packet. Note that the GUE control bit can potentially be used in a large set of protocols that are not OAM protocols. However, the control packet examples discussed in [I-D.ietf-nvo3-gue] are OAM-related.

Each of the three NVO3 encapsulation headers includes a 2-bit Version field, which is currently defined to be zero.

The Geneve and VXLAN-GPE headers include reserved fields; 14 bits in the Geneve header, and 27 bits in the VXLAN-GPE header are reserved.

11.4. Comparison Summary

The following table summarizes the comparison between the three NVO3 encapsulations.

	Geneve	GUE	VXLAN-GPE
Outer transport	UDP/IP	UDP/IP	UDP/IP

Base header length	8 octets	4 octets	8 octets (16 octets using NSH)
Extensibility	Variable length options	Extension fields	No native extensibility. Extensible using NSH.
Extension parsing method	TLV-based	Flag-based	TLV-based (using NSH with MD Type 2)
Extension order	Variable	Fixed	Variable (using NSH)
Length field	+	+	-
Max Header Length	260 octets	128 octets	8 octets (264 using NSH)
Critical extension bit	+	-	-
VNI field size	24 bits	32 bits (extension)	24 bits
Next protocol field	16 bits Ethertype registry	8 bits Internet protocol registry	8 bits New registry
Next protocol indicator	-	-	+
OAM / control field	OAM bit	Control bit	OAM bit
Version field	2 bits	2 bits	2 bits
Reserved bits	14 bits	-	27 bits

Figure 1: NVO3 Encapsulation Comparison

Authors' Addresses (In alphabetical order)

Sam Aldrin
Google
Email: aldrin.ietf@gmail.com

Ignas Bagdonas
Equinix
Email: ibagdona.ietf@gmail.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com

Sami Boutros
VMware
Email: sboutros@vmware.com

Uri Elzur
Intel
Email: uri.elzur@intel.com

Ilango Ganga
Intel
Email: ilango.s.ganga@intel.com

Pankaj Garg
Microsoft
Email: pankajg@microsoft.com

Rajeev Manur
Broadcom
Email: rajeev.manur@broadcom.com

Tal Mizrahi
Marvell
Email: talmi@marvell.com

David Mozes
Mellanox
Email: davidm@mellanox.com

Erik Nordmark
Arista Networks
Email: nordmark@sonic.net

Michael Smith
Cisco
Email: michsmit@cisco.com

NVO3 L. Huang, Ed.
Internet-Draft S. Hu
Intended status: Informational China Mobile
Expires: September 2, 2017 March 1, 2017

VxLAN Extension Requirement for Signaling Exchange Between Control and User Plane of vBras
draft-huang-nvo3-vxlan-extension-for-vbras-00

Abstract

This document briefly describes the architecture of control plane and user plane separated BRAS and the VxLAN extension requirement for Signaling Exchange between control plane and user plane. It also describes some possible solutions.

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). 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 September 2, 2017.

Table of Contents

1. Introduction

For migration of vBRAS, one way is separating the control and user plane of traditional BRAS. Control plane is deployed in centrolized cloud DC and user plane is fulfilled by high performance hardware device, e.g. router, switch, etc. VxLAN is used to transfer some of signaling packets between CP and user UP. For carrying information of access user VxLAN need to be extended or combined with other protocols/mechanisms.

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 [RFC2119].

3. Requirement

The architecture of C/U separated BRAS is shown as the following figure.

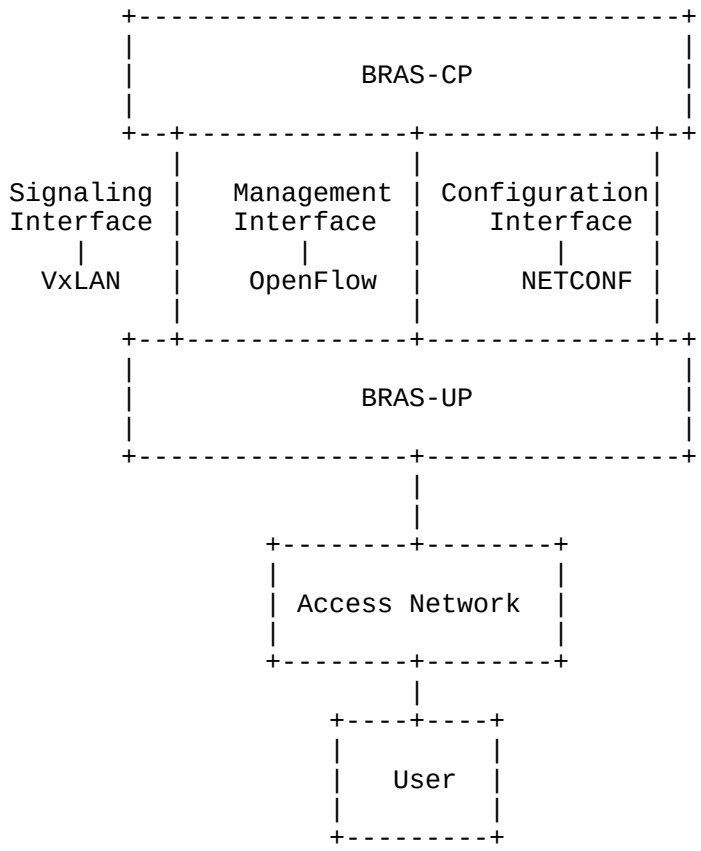


Figure 1: Architecture of C/U separated vBRAS

In this architecture, control plane (CP) is responsible for user access authentication and setting forwarding entries of user plane if authentication is successful. User plane (UP) need to relay PPPoE/ IPoE signaling packets between users and CP and forward PPPoE/IPoE data packets to Internet based on the forwarding entries set by CP. CP should do some basic configurations on UP, e.g. service configuration.

There are three interfaces between CP and UP. Configuration interface is used by CP to carry out basic configurations of user plane through NETCONF. Management interface is used for seting forwarding entries of user plane through OpenFlow. Signaling interface is used to transfering PPPoE/IPoE signaling packets between user plane and control plane. VxLAN is chosen for signaling interface since it’s a relatively mature technology and can carry L2

packets through L3 network. For user access authentication, CP need to know which port of UP the user is connected to for the authentication of access location because a specfic user is only permitted on specific port. Usually the following information is necessary: device ID, slot ID, subcard ID and port ID, which need about 16 to 32 bits totally. The access port information should be carried in VxLAN packets encapsulated by UP. So an entension on VxLAN or some other mechanism is necessary for this requirement.

4. Analysis on solutions

4.1. VxLAN header solution

In VxLAN header, two possible fields can carry the required information.

One is VNI field in which 16 bits is used for carrying port information and 8 bits for the real VxLAN ID. The advantage is no amendment on VxLAN header is required and most of current devices can work as UP based on standard VxLAN, e.g. routers, switches. The disadvantage is too many VxLAN tunnels must be built and it’s hard for CP to build VxLAN tunnel with UP since it doesn’t know VNI before receiving VxLAN packets from UP.

Another is the reserved 32 bits in VxLAN header. The advantage is the VNI and port info could be arranged separately and no further field is introduce in. The disadvantage is the available reserved bits is limited and the usage of reserved bits might conflict with other draft. In addition, current devices need to upgrade to support this new field.

At last, maybe we can extended VxLAN header. For example, we can define a extension sign in current reverved bits and extend more fielded beyond current format of VxLAN. The advantagethe is VNI and port info could be arranged flexibly. The disadvantage is VxLAN header should be amended and current devices need to upgrade to support the amendment. The following figure is an example.

Please view in a fixed-width font such as Courier.

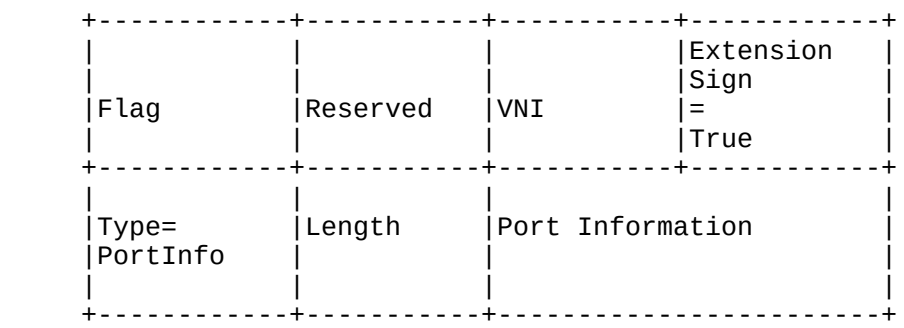


Figure 2: Example of extension of VxLAN header

4.2. Combination of VxLAN and NSH

As mentioned in Network Service Header [I-D.ietf-sfc-nsh] and Generic Protocol Extension for VXLAN [I-D.ietf-nvo3-vxlan-gpe], the field of "Context Headers" in NSH can be used for carrying the required information. The advantage is VNI and port info could be arranged flexibly. The disadvantage is that introducing NSH in is a little heavier than required and more functional requirements are put on UP device. Furthermore, VxLAN-GPE and NSH are still in draft status.

4.3. Assign VNIs for every port

This method is also based on VNI. When UP is connected to CP, CP will assign different VNI for every port. Both CP and UP should keep a mapping table of VNI and port. Then CP can encapsulate PPPoE/IPoE signaling packets with a specific VNI based on receiving port. CP can get the port information through VNI and the mapping table. The advantage is no amendment on VxLAN header is required and most of current devices can work as UP based on standard VxLAN. The disadvantage is too many VxLAN tunnels must be built and extra complex mechanisms should be included in for VNI assignment and the storage of mapping table.

4.4. Summury of these soluitlons

By comparison of the above solutions, we prefer to extend VxLAN header to meet the requirement. Because it seems leverage the flexibility and complexity. So some kind of extension on VxLAN header need to be standardized.

5. Security Considerations

None.

6. IANA Considerations

None.

7. Normative References

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-03 (work in progress), October 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

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

[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, <http://www.rfc-editor.org/info/rfc7348>.

Authors' Addresses

Lu Huang (editor)
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: hlisname@yahoo.com

Shujun Hu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: 13488683482@139.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

J. Gross, Ed.
I. Ganga, Ed.
Intel
T. Sridhar, Ed.
VMware
March 13, 2017

Geneve: Generic Network Virtualization Encapsulation
draft-ietf-nvo3-geneve-04

Abstract

Network virtualization involves the cooperation of devices with a wide variety of capabilities such as software and hardware tunnel endpoints, transit fabrics, and centralized control clusters. As a result of their role in tying together different elements in the system, the requirements on tunnels are influenced by all of these components. Flexibility is therefore the most important aspect of a tunnel protocol if it is to keep pace with the evolution of the system. This draft describes Geneve, a protocol designed to recognize and accommodate these changing capabilities and needs.

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 September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

Networking has long featured a variety of tunneling, tagging, and other encapsulation mechanisms. However, the advent of network virtualization has caused a surge of renewed interest and a corresponding increase in the introduction of new protocols. The large number of protocols in this space, ranging all the way from VLANs [IEEE.802.1Q-2014] and MPLS [RFC3031] through the more recent VXLAN [RFC7348], NVGRE [RFC7637], and STT [I-D.davie-stt], often leads to questions about the need for new encapsulation formats and what it is about network virtualization in particular that leads to their proliferation.

While many encapsulation protocols seek to simply partition the underlay network or bridge between two domains, network virtualization views the transit network as providing connectivity between multiple components of a distributed system. In many ways this system is similar to a chassis switch with the IP underlay network playing the role of the backplane and tunnel endpoints on the edge as line cards. When viewed in this light, the requirements placed on the tunnel protocol are significantly different in terms of the quantity of metadata necessary and the role of transit nodes.

Current work such as [VL2] and the NVO3 working group [I-D.ietf-nvo3-dataplane-requirements] have described some of the properties that the data plane must have to support network virtualization. However, one additional defining requirement is the need to carry system state along with the packet data. The use of some metadata is certainly not a foreign concept - nearly all protocols used for virtualization have at least 24 bits of identifier space as a way to partition between tenants. This is often described as overcoming the limits of 12-bit VLANs, and when seen in that context, or any context where it is a true tenant identifier, 16 million possible entries is a large number. However, the reality is that the metadata is not exclusively used to identify tenants and encoding other information quickly starts to crowd the space. In fact, when compared to the tags used to exchange metadata between line cards on a chassis switch, 24-bit identifiers start to look quite small. There are nearly endless uses for this metadata, ranging from storing input ports for simple security policies to service based context for interposing advanced middleboxes.

Existing tunnel protocols have each attempted to solve different aspects of these new requirements, only to be quickly rendered out of date by changing control plane implementations and advancements. Furthermore, software and hardware components and controllers all have different advantages and rates of evolution - a fact that should be viewed as a benefit, not a liability or limitation. This draft

describes Geneve, a protocol which seeks to avoid these problems by providing a framework for tunneling for network virtualization rather than being prescriptive about the entire system.

1.1. 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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

1.2. Terminology

The NVO3 framework [RFC7365] defines many of the concepts commonly used in network virtualization. In addition, the following terms are specifically meaningful in this document:

Checksum offload. An optimization implemented by many NICs which enables computation and verification of upper layer protocol checksums in hardware on transmit and receive, respectively. This typically includes IP and TCP/UDP checksums which would otherwise be computed by the protocol stack in software.

Clos network. A technique for composing network fabrics larger than a single switch while maintaining non-blocking bandwidth across connection points. ECMP is used to divide traffic across the multiple links and switches that constitute the fabric. Sometimes termed "leaf and spine" or "fat tree" topologies.

ECMP. Equal Cost Multipath. A routing mechanism for selecting from among multiple best next hop paths by hashing packet headers in order to better utilize network bandwidth while avoiding reordering a single stream.

Geneve. Generic Network Virtualization Encapsulation. The tunnel protocol described in this draft.

LRO. Large Receive Offload. The receive-side equivalent function of LSO, in which multiple protocol segments (primarily TCP) are coalesced into larger data units.

NIC. Network Interface Card. A NIC could be part of a tunnel endpoint or transit device and can either process Geneve packets or aid in the processing of Geneve packets.

OAM. Operations, Administration, and Management. A suite of tools used to monitor and troubleshoot network problems.

Transit device. A forwarding element along the path of the tunnel making up part of the Underlay Network. A transit device MAY be capable of understanding the Geneve packet format but does not originate or terminate Geneve packets.

LSO. Large Segmentation Offload. A function provided by many commercial NICs that allows data units larger than the MTU to be passed to the NIC to improve performance, the NIC being responsible for creating smaller segments of size less than or equal to the MTU with correct protocol headers. When referring specifically to TCP/IP, this feature is often known as TSO (TCP Segmentation Offload).

Tunnel endpoint. A component performing encapsulation and decapsulation of packets, such as Ethernet frames or IP datagrams, in Geneve headers. As the ultimate consumer of any tunnel metadata, endpoints have the highest level of requirements for parsing and interpreting tunnel headers. Tunnel endpoints may consist of either software or hardware implementations or a combination of the two. Endpoints are frequently a component of an NVE but may also be found in middleboxes or other elements making up an NVO3 Network.

VM. Virtual Machine.

2. Design Requirements

Geneve is designed to support network virtualization use cases, where tunnels are typically established to act as a backplane between the virtual switches residing in hypervisors, physical switches, or middleboxes or other appliances. An arbitrary IP network can be used as an underlay although Clos networks composed using ECMP links are a common choice to provide consistent bisectional bandwidth across all connection points. Figure 1 shows an example of a hypervisor, top of rack switch for connectivity to physical servers, and a WAN uplink connected using Geneve tunnels over a simplified Clos network. These tunnels are used to encapsulate and forward frames from the attached components such as VMs or physical links.

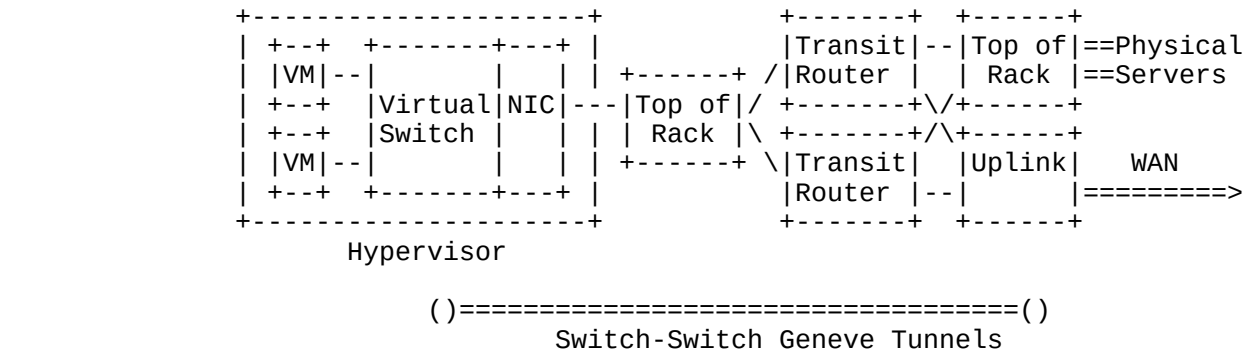


Figure 1: Sample Geneve Deployment

To support the needs of network virtualization, the tunnel protocol should be able to take advantage of the differing (and evolving) capabilities of each type of device in both the underlay and overlay networks. This results in the following requirements being placed on the data plane tunneling protocol:

- o The data plane is generic and extensible enough to support current and future control planes.
- o Tunnel components are efficiently implementable in both hardware and software without restricting capabilities to the lowest common denominator.
- o High performance over existing IP fabrics.

These requirements are described further in the following subsections.

2.1. Control Plane Independence

Although some protocols for network virtualization have included a control plane as part of the tunnel format specification (most notably, the original VXLAN spec prescribed a multicast learning-based control plane), these specifications have largely been treated as describing only the data format. The VXLAN packet format has actually seen a wide variety of control planes built on top of it.

There is a clear advantage in settling on a data format: most of the protocols are only superficially different and there is little advantage in duplicating effort. However, the same cannot be said of control planes, which are diverse in very fundamental ways. The case for standardization is also less clear given the wide variety in requirements, goals, and deployment scenarios.

As a result of this reality, Geneve aims to be a pure tunnel format specification that is capable of fulfilling the needs of many control planes by explicitly not selecting any one of them. This simultaneously promotes a shared data format and increases the chances that it will not be obsoleted by future control plane enhancements.

2.2. Data Plane Extensibility

Achieving the level of flexibility needed to support current and future control planes effectively requires an options infrastructure to allow new metadata types to be defined, deployed, and either finalized or retired. Options also allow for differentiation of products by encouraging independent development in each vendor's core specialty, leading to an overall faster pace of advancement. By far the most common mechanism for implementing options is Type-Length-Value (TLV) format.

It should be noted that while options can be used to support non-wirespeed control packets, they are equally important on data packets as well to segregate and direct forwarding (for instance, the examples given before of input port based security policies and service interposition both require tags to be placed on data packets). Therefore, while it would be desirable to limit the extensibility to only control packets for the purposes of simplifying the datapath, that would not satisfy the design requirements.

2.2.1. Efficient Implementation

There is often a conflict between software flexibility and hardware performance that is difficult to resolve. For a given set of functionality, it is obviously desirable to maximize performance. However, that does not mean new features that cannot be run at that speed today should be disallowed. Therefore, for a protocol to be efficiently implementable means that a set of common capabilities can be reasonably handled across platforms along with a graceful mechanism to handle more advanced features in the appropriate situations.

The use of a variable length header and options in a protocol often raises questions about whether it is truly efficiently implementable in hardware. To answer this question in the context of Geneve, it is important to first divide "hardware" into two categories: tunnel endpoints and transit devices.

Endpoints must be able to parse the variable header, including any options, and take action. Since these devices are actively participating in the protocol, they are the most affected by Geneve.

However, as endpoints are the ultimate consumers of the data, transmitters can tailor their output to the capabilities of the recipient. As new functionality becomes sufficiently well defined to add to endpoints, supporting options can be designed using ordering restrictions and other techniques to ease parsing.

Transit devices MAY be able to interpret the options and participate in Geneve packet processing. However, as non-terminating devices, they do not originate or terminate the Geneve packet. The participation of transit devices in Geneve packet processing is OPTIONAL.

Further, either tunnel endpoints or transit devices MAY use offload capabilities of NICs such as checksum offload to improve the performance of Geneve packet processing. The presence of a Geneve variable length header SHOULD NOT prevent the tunnel endpoints and transit devices from using such offload capabilities.

2.3. Use of Standard IP Fabrics

IP has clearly cemented its place as the dominant transport mechanism and many techniques have evolved over time to make it robust, efficient, and inexpensive. As a result, it is natural to use IP fabrics as a transit network for Geneve. Fortunately, the use of IP encapsulation and addressing is enough to achieve the primary goal of delivering packets to the correct point in the network through standard switching and routing.

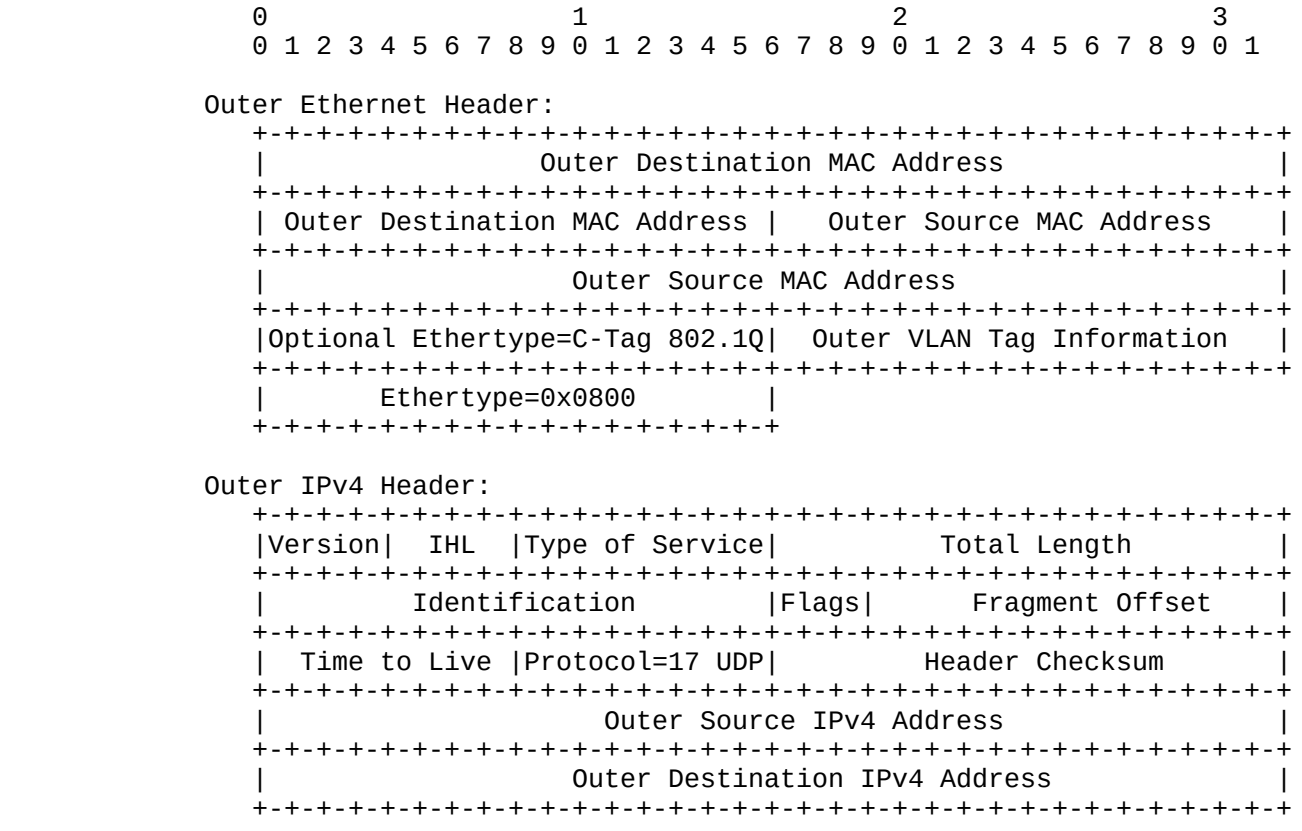
In addition, nearly all underlay fabrics are designed to exploit parallelism in traffic to spread load across multiple links without introducing reordering in individual flows. These equal cost multipathing (ECMP) techniques typically involve parsing and hashing the addresses and port numbers from the packet to select an outgoing link. However, the use of tunnels often results in poor ECMP performance without additional knowledge of the protocol as the encapsulated traffic is hidden from the fabric by design and only endpoint addresses are available for hashing.

Since it is desirable for Geneve to perform well on these existing fabrics, it is necessary for entropy from encapsulated packets to be exposed in the tunnel header. The most common technique for this is to use the UDP source port, which is discussed further in Section 3.3.

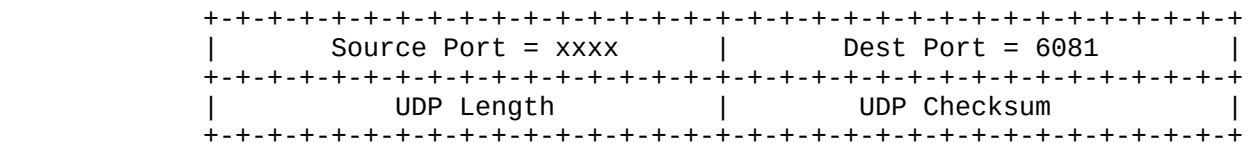
3. Geneve Encapsulation Details

The Geneve packet format consists of a compact tunnel header encapsulated in UDP over either IPv4 or IPv6. A small fixed tunnel header provides control information plus a base level of functionality and interoperability with a focus on simplicity. This header is then followed by a set of variable options to allow for future innovation. Finally, the payload consists of a protocol data unit of the indicated type, such as an Ethernet frame. Section 3.1 and Section 3.2 illustrate the Geneve packet format transported (for example) over Ethernet along with an Ethernet payload.

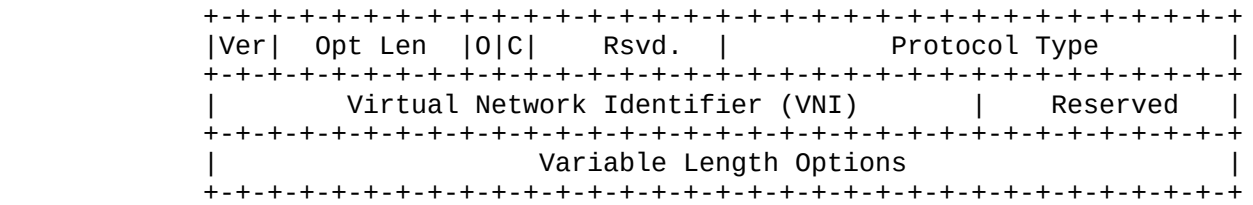
3.1. Geneve Packet Format Over IPv4



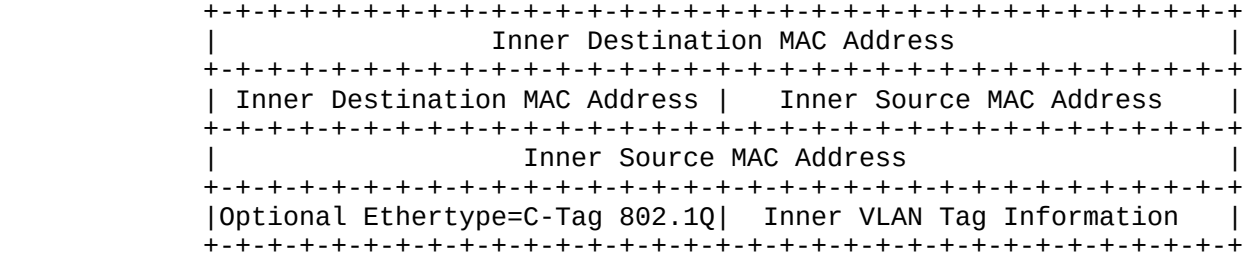
Outer UDP Header:



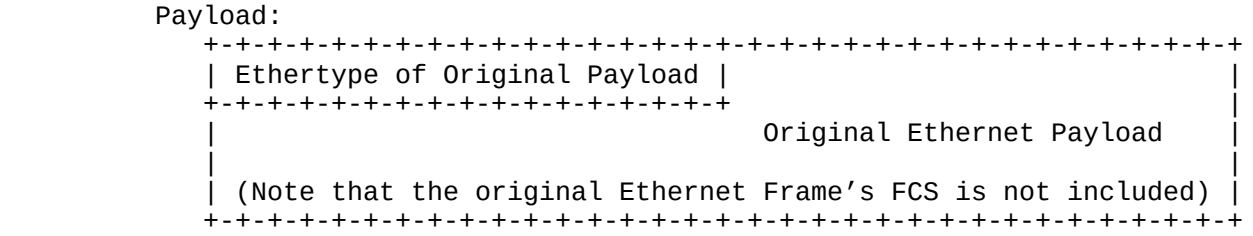
Geneve Header:



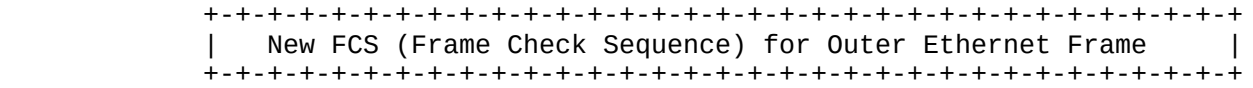
Inner Ethernet Header (example payload):



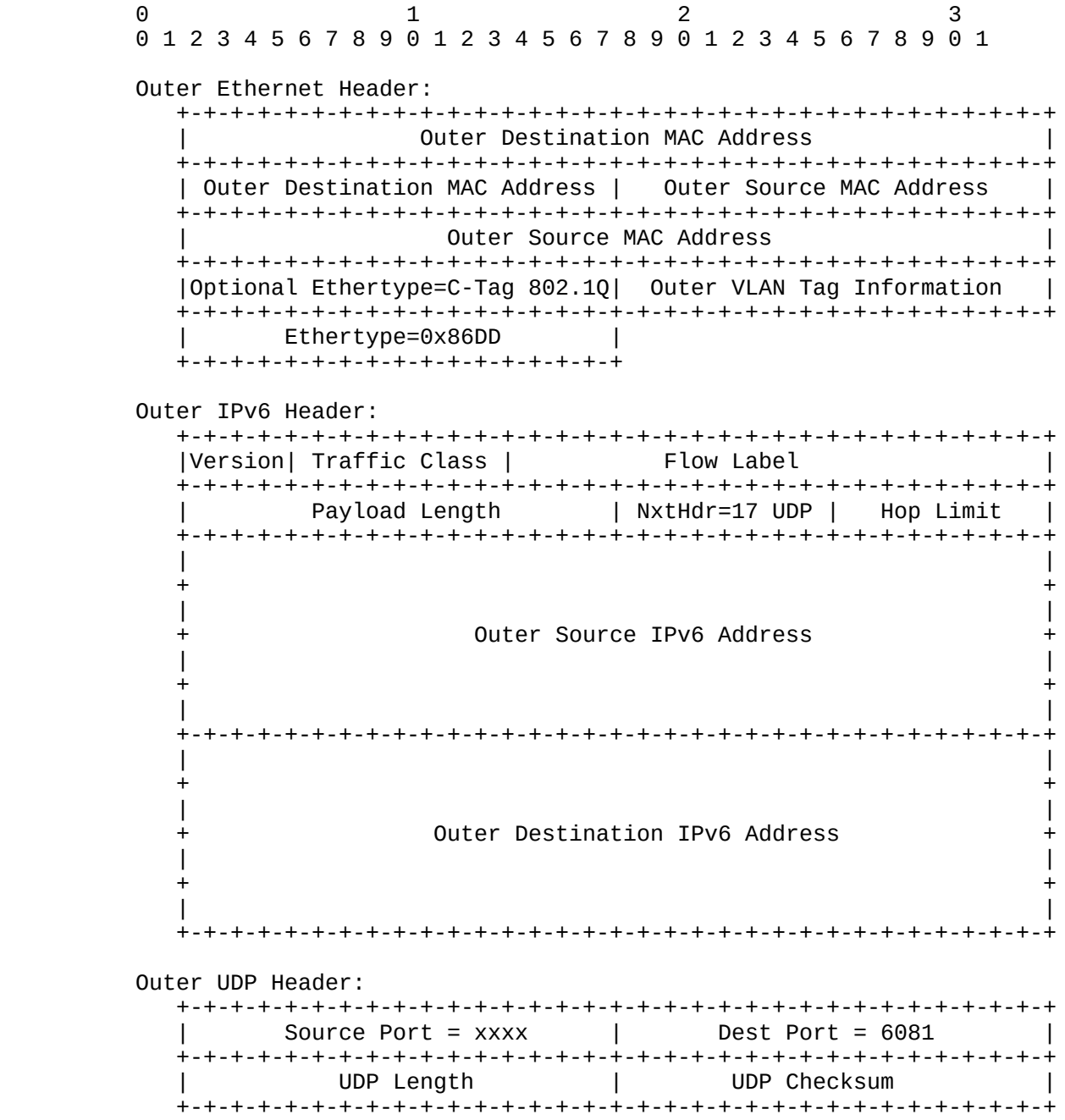
Payload:



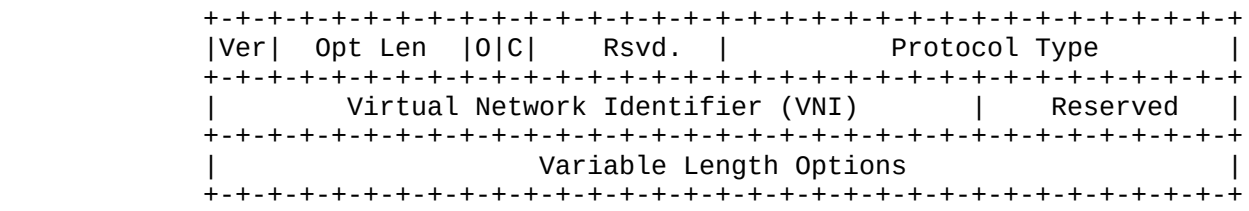
Frame Check Sequence:



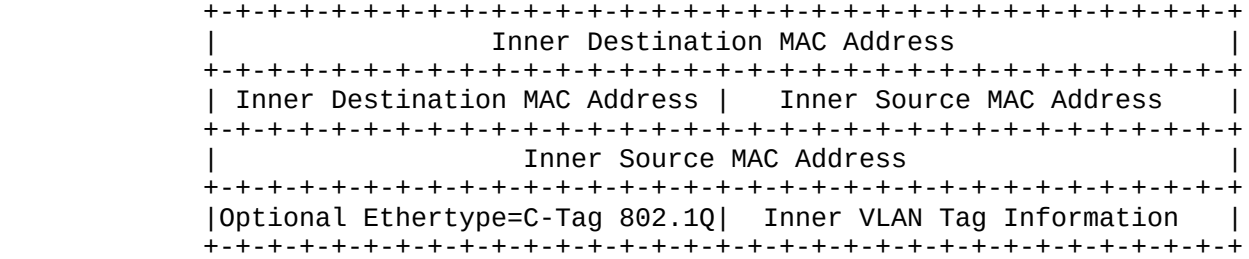
3.2. Geneve Packet Format Over IPv6


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
Optional Ethertype=C-Tag 802.1Q
Outer VLAN Tag Information
Ethertype=0x86DD
Outer IPv6 Header:
Version
Traffic Class
Flow Label
Payload Length
NxtHdr=17 UDP
Hop Limit
Outer Source IPv6 Address
Outer Destination IPv6 Address
Outer UDP Header:
Source Port = xxxx
Dest Port = 6081
UDP Length
UDP Checksum

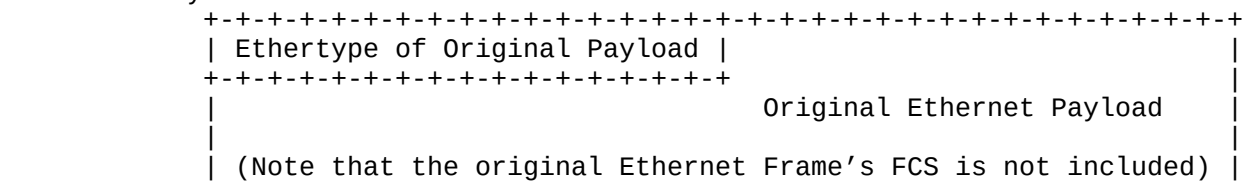
Geneve Header:



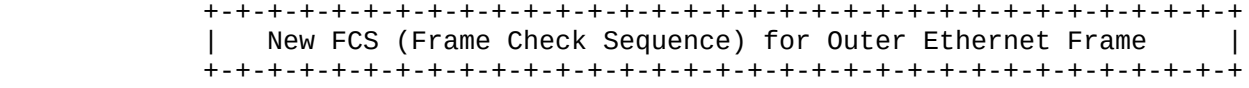
Inner Ethernet Header (example payload):



Payload:



Frame Check Sequence:



3.3. UDP Header

The use of an encapsulating UDP [RFC0768] header follows the connectionless semantics of Ethernet and IP in addition to providing entropy to routers performing ECMP. The header fields are therefore interpreted as follows:

Source port: A source port selected by the originating tunnel endpoint. This source port SHOULD be the same for all packets belonging to a single encapsulated flow to prevent reordering due to the use of different paths. To encourage an even distribution of flows across multiple links, the source port SHOULD be calculated using a hash of the encapsulated packet headers using, for example, a traditional 5-tuple. Since the port represents a

flow identifier rather than a true UDP connection, the entire 16-bit range MAY be used to maximize entropy.

Dest port: IANA has assigned port 6081 as the fixed well-known destination port for Geneve. Although the well-known value should be used by default, it is RECOMMENDED that implementations make this configurable. The chosen port is used for identification of Geneve packets and MUST NOT be reversed for different ends of a connection as is done with TCP.

UDP length: The length of the UDP packet including the UDP header.

UDP checksum: The checksum MAY be set to zero on transmit for packets encapsulated in both IPv4 and IPv6 [RFC6935]. When a packet is received with a UDP checksum of zero it MUST be accepted and decapsulated. If the originating tunnel endpoint optionally encapsulates a packet with a non-zero checksum, it MUST be a correctly computed UDP checksum. Upon receiving such a packet, the egress endpoint MUST validate the checksum. If the checksum is not correct, the packet MUST be dropped, otherwise the packet MUST be accepted for decapsulation. It is RECOMMENDED that the UDP checksum be computed to protect the Geneve header and options in situations where the network reliability is not high and the packet is not protected by another checksum or CRC.

3.4. Tunnel Header Fields

Ver (2 bits): The current version number is 0. Packets received by an endpoint with an unknown version MUST be dropped. Non-terminating devices processing Geneve packets with an unknown version number MUST treat them as UDP packets with an unknown payload.

Opt Len (6 bits): The length of the options fields, expressed in four byte multiples, not including the eight byte fixed tunnel header. This results in a minimum total Geneve header size of 8 bytes and a maximum of 260 bytes. The start of the payload headers can be found using this offset from the end of the base Geneve header.

O (1 bit): OAM packet. This packet contains a control message instead of a data payload. Endpoints MUST NOT forward the payload and transit devices MUST NOT attempt to interpret or process it. Since these are infrequent control messages, it is RECOMMENDED that endpoints direct these packets to a high priority control queue (for example, to direct the packet to a general purpose CPU from a forwarding ASIC or to separate out control traffic on a

NIC). Transit devices MUST NOT alter forwarding behavior on the basis of this bit, such as ECMP link selection.

C (1 bit): Critical options present. One or more options has the critical bit set (see Section 3.5). If this bit is set then tunnel endpoints MUST parse the options list to interpret any critical options. On endpoints where option parsing is not supported the packet MUST be dropped on the basis of the 'C' bit in the base header. If the bit is not set tunnel endpoints MAY strip all options using 'Opt Len' and forward the decapsulated packet. Transit devices MUST NOT drop or modify packets on the basis of this bit.

Rsvd. (6 bits): Reserved field which MUST be zero on transmission and ignored on receipt.

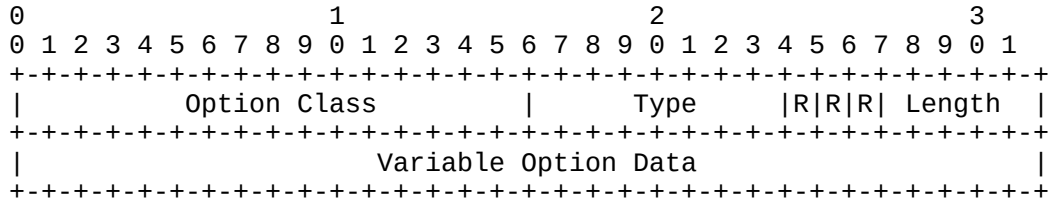
Protocol Type (16 bits): The type of the protocol data unit appearing after the Geneve header. This follows the EtherType [ETYPES] convention with Ethernet itself being represented by the value 0x6558.

Virtual Network Identifier (VNI) (24 bits): An identifier for a unique element of a virtual network. In many situations this may represent an L2 segment, however, the control plane defines the forwarding semantics of decapsulated packets. The VNI MAY be used as part of ECMP forwarding decisions or MAY be used as a mechanism to distinguish between overlapping address spaces contained in the encapsulated packet when load balancing across CPUs.

Reserved (8 bits): Reserved field which MUST be zero on transmission and ignored on receipt.

Transit devices MUST maintain consistent forwarding behavior irrespective of the value of 'Opt Len', including ECMP link selection. These devices SHOULD be able to forward packets containing options without resorting to a slow path.

3.5. Tunnel Options



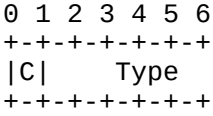
Geneve Option

The base Geneve header is followed by zero or more options in Type-Length-Value format. Each option consists of a four byte option header and a variable amount of option data interpreted according to the type.

Option Class (16 bits): Namespace for the 'Type' field. IANA will be requested to create a "Geneve Option Class" registry to allocate identifiers for organizations, technologies, and vendors that have an interest in creating types for options. Each organization may allocate types independently to allow experimentation and rapid innovation. It is expected that over time certain options will become well known and a given implementation may use option types from a variety of sources. In addition, IANA will be requested to reserve specific ranges for standardized and experimental options.

Type (8 bits): Type indicating the format of the data contained in this option. Options are primarily designed to encourage future extensibility and innovation and so standardized forms of these options will be defined in a separate document.

The high order bit of the option type indicates that this is a critical option. If the receiving endpoint does not recognize this option and this bit is set then the packet MUST be dropped. If the critical bit is set in any option then the 'C' bit in the Geneve base header MUST also be set. Transit devices MUST NOT drop packets on the basis of this bit. The following figure shows the location of the 'C' bit in the 'Type' field:



The requirement to drop a packet with an unknown critical option applies to the entire tunnel endpoint system and not a particular component of the implementation. For example, in a system comprised of a forwarding ASIC and a general purpose CPU, this does not mean that the packet must be dropped in the ASIC. An implementation may send the packet to the CPU using a rate-limited control channel for slow-path exception handling.

R (3 bits): Option control flags reserved for future use. MUST be zero on transmission and ignored on receipt.

Length (5 bits): Length of the option, expressed in four byte multiples excluding the option header. The total length of each option may be between 4 and 128 bytes. Packets in which the total

length of all options is not equal to the 'Opt Len' in the base header are invalid and MUST be silently dropped if received by an endpoint.

Variable Option Data: Option data interpreted according to 'Type'.

3.5.1. Options Processing

Geneve options are primarily intended to be originated and processed by tunnel endpoints. However, options MAY be processed by transit devices along the tunnel path as well. Transit devices not processing Geneve headers SHOULD process Geneve packets as any other UDP packet and maintain consistent forwarding behavior.

In tunnel endpoints, the generation and interpretation of options is determined by the control plane, which is out of the scope of this document. However, to ensure interoperability between heterogeneous devices some requirements are imposed on options and the devices that process them:

- Receiving endpoints MUST drop packets containing unknown options with the 'C' bit set in the option type. Conversely, transit devices MUST NOT drop packets as a result of encountering unknown options, including those with the 'C' bit set.

- Some options may be defined in such a way that the position in the option list is significant. Therefore, options MUST NOT be reordered by transit devices.

- An option MUST NOT affect the parsing or interpretation of any other option.

When designing a Geneve option, it is important to consider how the option will evolve in the future. Once an option is defined it is reasonable to expect that implementations may come to depend on a specific behavior. As a result, the scope of any future changes must be carefully described upfront.

Unexpectedly significant interoperability issues may result from changing the length of an option that was defined to be a certain size. A particular option is specified to have either a fixed length, which is constant, or a variable length, which may change over time or for different use cases. This property is part of the definition of the option and conveyed by the 'Type'. For fixed length options, some implementations may choose to ignore the length field in the option header and instead parse based on the well known length associated with the type. In this case, redefining the length will impact not only parsing of the option in question but also any

options that follow. Therefore, options that are defined to be fixed length in size MUST NOT be redefined to a different length. Instead, a new 'Type' should be allocated.

4. Implementation and Deployment Considerations

4.1. Encapsulation of Geneve in IP

As an IP-based tunnel protocol, Geneve shares many properties and techniques with existing protocols. The application of some of these are described in further detail, although in general most concepts applicable to the IP layer or to IP tunnels generally also function in the context of Geneve.

4.1.1. IP Fragmentation

To prevent fragmentation and maximize performance, the best practice when using Geneve is to ensure that the MTU of the physical network is greater than or equal to the MTU of the encapsulated network plus tunnel headers. Manual or upper layer (such as TCP MSS clamping) configuration can be used to ensure that fragmentation never takes place, however, in some situations this may not be feasible.

It is strongly RECOMMENDED that Path MTU Discovery ([RFC1191], [RFC1981]) be used by setting the DF bit in the IP header when Geneve packets are transmitted over IPv4 (this is the default with IPv6). The use of Path MTU Discovery on the transit network provides the encapsulating endpoint with soft-state about the link that it may use to prevent or minimize fragmentation depending on its role in the virtualized network.

Note that some implementations may not be capable of supporting fragmentation or other less common features of the IP header, such as options and extension headers.

4.1.2. DSCP and ECN

When encapsulating IP (including over Ethernet) packets in Geneve, there are several considerations for propagating DSCP and ECN bits from the inner header to the tunnel on transmission and the reverse on reception.

[RFC2983] provides guidance for mapping DSCP between inner and outer IP headers. Network virtualization is typically more closely aligned with the Pipe model described, where 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). 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 Geneve.

4.1.3. Broadcast and Multicast

Geneve tunnels may either be point-to-point unicast between two endpoints or may utilize broadcast or multicast addressing. It is not required that inner and outer addressing match in this respect. For example, in physical networks that do not support multicast, encapsulated multicast traffic may be replicated into multiple unicast tunnels or forwarded by policy to a unicast location (possibly to be replicated there).

With physical networks that do support multicast it may be desirable to use this capability to take advantage of hardware replication for encapsulated packets. In this case, multicast addresses may be allocated in the physical network corresponding to tenants, encapsulated multicast groups, or some other factor. The allocation of these groups is a component of the control plane and therefore outside of the scope of this document. When physical multicast is in use, the 'C' bit in the Geneve header may be used with groups of devices with heterogeneous capabilities as each device can interpret only the options that are significant to it if they are not critical.

4.1.4. Unidirectional Tunnels

Generally speaking, a Geneve tunnel is a unidirectional concept. IP is not a connection oriented protocol and it is possible for two endpoints to communicate with each other using different paths or to have one side not transmit anything at all. As Geneve is an IP-based protocol, the tunnel layer inherits these same characteristics.

It is possible for a tunnel to encapsulate a protocol, such as TCP, which is connection oriented and maintains session state at that layer. In addition, implementations MAY model Geneve tunnels as connected, bidirectional links, such as to provide the abstraction of a virtual port. In both of these cases, bidirectionality of the tunnel is handled at a higher layer and does not affect the operation of Geneve itself.

4.2. Constraints on Protocol Features

Geneve is intended to be flexible to a wide range of current and future applications. As a result, certain constraints may be placed on the use of metadata or other aspects of the protocol in order to optimize for a particular use case. For example, some applications may limit the types of options which are supported or enforce a maximum number or length of options. Other applications may only handle certain encapsulated payload types, such as Ethernet or IP. This could be either globally throughout the system or, for example, restricted to certain classes of devices or network paths.

These constraints may be communicated to tunnel endpoints either explicitly through a control plane or implicitly by the nature of the application. As Geneve is defined as a data plane protocol that is control plane agnostic, the exact mechanism is not defined in this document.

4.2.1. Constraints on Options

While Geneve options are more flexible, a control plane may restrict the number of option TLVs as well as the order and size of the TLVs, between tunnel endpoints, to make it simpler for a data plane implementation in software or hardware to handle [I-D.dt-nvo3-encap]. For example, there may be some critical information such as a secure hash that must be processed in a certain order to provide lowest latency.

A control plane may negotiate a subset of option TLVs and certain TLV ordering, as well may limit the total number of option TLVs present in the packet, for example, to accommodate hardware capable of processing fewer options [I-D.dt-nvo3-encap]. Hence, a control plane needs to have the ability to describe the supported TLVs subset and their order to the tunnel end points. In the absence of a control plane, alternative configuration mechanisms may be used for this purpose. The exact mechanism is not defined in this document.

4.3. NIC Offloads

Modern NICs currently provide a variety of offloads to enable the efficient processing of packets. The implementation of many of these offloads requires only that the encapsulated packet be easily parsed (for example, checksum offload). However, optimizations such as LSO and LRO involve some processing of the options themselves since they must be replicated/merged across multiple packets. In these situations, it is desirable to not require changes to the offload logic to handle the introduction of new options. To enable this,

some constraints are placed on the definitions of options to allow for simple processing rules:

- When performing LSO, a NIC MUST replicate the entire Geneve header and all options, including those unknown to the device, onto each resulting segment. However, a given option definition may override this rule and specify different behavior in supporting devices. Conversely, when performing LRO, a NIC MAY assume that a binary comparison of the options (including unknown options) is sufficient to ensure equality and MAY merge packets with equal Geneve headers.

- Options MUST NOT be reordered during the course of offload processing, including when merging packets for the purpose of LRO.

- NICs performing offloads MUST NOT drop packets with unknown options, including those marked as critical.

There is no requirement that a given implementation of Geneve employ the offloads listed as examples above. However, as these offloads are currently widely deployed in commercially available NICs, the rules described here are intended to enable efficient handling of current and future options across a variety of devices.

4.4. Inner VLAN Handling

Geneve is capable of encapsulating a wide range of protocols and therefore a given implementation is likely to support only a small subset of the possibilities. However, as Ethernet is expected to be widely deployed, it is useful to describe the behavior of VLANs inside encapsulated Ethernet frames.

As with any protocol, support for inner VLAN headers is OPTIONAL. In many cases, the use of encapsulated VLANs may be disallowed due to security or implementation considerations. However, in other cases trunking of VLAN frames across a Geneve tunnel can prove useful. As a result, the processing of inner VLAN tags upon ingress or egress from a tunnel endpoint is based upon the configuration of the endpoint and/or control plane and not explicitly defined as part of the data format.

5. Interoperability Issues

Viewed exclusively from the data plane, Geneve does not introduce any interoperability issues as it appears to most devices as UDP packets. However, as there are already a number of tunnel protocols deployed in network virtualization environments, there is a practical question of transition and coexistence.

Since Geneve is a superset of the functionality of the three most common protocols used for network virtualization (VXLAN, NVGRE, and STT) it should be straightforward to port an existing control plane to run on top of it with minimal effort. With both the old and new packet formats supporting the same set of capabilities, there is no need for a hard transition - endpoints directly communicating with each other use any common protocol, which may be different even within a single overall system. As transit devices are primarily forwarding packets on the basis of the IP header, all protocols appear similar and these devices do not introduce additional interoperability concerns.

To assist with this transition, it is strongly suggested that implementations support simultaneous operation of both Geneve and existing tunnel protocols as it is expected to be common for a single node to communicate with a mixture of other nodes. Eventually, older protocols may be phased out as they are no longer in use.

6. Security Considerations

As UDP/IP packets, Geneve does not have any inherent security mechanisms. As a result, an attacker with access to the underlay network transporting the IP packets has the ability to snoop or inject packets. Legitimate but malicious tunnel endpoints may also spoof identifiers in the tunnel header to gain access to networks owned by other tenants.

Within a particular security domain, such as a data center operated by a single provider, the most common and highest performing security mechanism is isolation of trusted components. Tunnel traffic can be carried over a separate VLAN and filtered at any untrusted boundaries. In addition, tunnel endpoints should only be operated in environments controlled by the service provider, such as the hypervisor itself rather than within a customer VM.

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 Geneve encapsulation. If the remote tunnel endpoint is not completely trusted, for example it resides on a customer premises, then it may also be necessary to sanitize any tunnel metadata to prevent tenant-hopping attacks.

Geneve does not otherwise affect the security of the encapsulated packets.

7. IANA Considerations

IANA has allocated UDP port 6081 as the well-known destination port for Geneve. Upon publication, the registry should be updated to cite this document. The original request was:

Service Name: geneve
Transport Protocol(s): UDP
Assignee: Jesse Gross <jgross@vmware.com>
Contact: Jesse Gross <jgross@vmware.com>
Description: Generic Network Virtualization Encapsulation (Geneve)
Reference: This document
Port Number: 6081

In addition, IANA is requested to create a "Geneve Option Class" registry to allocate Option Classes. This shall be a registry of 16-bit hexadecimal values along with descriptive strings. The identifiers 0x0-0xFF are to be reserved for standardized options for allocation by IETF Review [RFC5226] and 0xFFF0-0xFFFF for Experimental Use. Otherwise, identifiers are to be assigned to any organization with an interest in creating Geneve options on a First Come First Served basis. The registry is to be populated with the following initial values:

Option Class	Description
0x0000..0x00FF	Unassigned - IETF Review
0x0100	Linux
0x0101	Open vSwitch
0x0102	Open Virtual Networking (OVN)
0x0103	In-band Network Telemetry (INT)
0x0104	VMware
0x0105..0xFFEF	Unassigned - First Come First Served
0xFFF0..FFFF	Experimental

8. Contributors

The following individuals were authors of an earlier version of this document and made significant contributions:

Pankaj Garg
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Chris Wright
Red Hat Inc.
1801 Varsity Drive
Raleigh, NC 27606
USA

Email: chrisw@redhat.com

Puneet Agarwal
Innovium, Inc.
6001 America Center Drive
San Jose, CA 95002
USA

Email: puneet@innovium.com

Kenneth Duda
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: kduda@arista.com

Dinesh G. Dutt
Cumulus Networks
140C S. Whisman Road
Mountain View, CA 94041
USA

Email: ddutt@cumulusnetworks.com

Jon Hudson
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Ariel Hendel
Broadcom Limited
3151 Zanker Road
San Jose, CA 95134
USA

Email: ariel.hendel@broadcom.com

9. Acknowledgements

The authors wish to thank Martin Casado, Bruce Davie and Dave Thaler for their input, feedback, and helpful suggestions.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

10.2. Informative References

[ETYPES] The IEEE Registration Authority, "IEEE 802 Numbers", 2013, <http://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xml>.

[I-D.davie-stt] Davie, B. and J. Gross, "A Stateless Transport Tunneling Protocol for Network Virtualization (STT)", draft-davie-stt-08 (work in progress), April 2016.

[I-D.dt-nvo3-encap] Boutros, S., Aldrin, S., Elzur, U., Ganga, I., Manur, R., Mozes, D., and M. Smith, "NVO3 Encapsulation Considerations", draft-dt-nvo3-encap-01 (work in progress), March 2017.

[I-D.ietf-nvo3-dataplane-requirements]
Bitar, N., Lasserre, M., Balus, F., Morin, T., Jin, L., and B. Khasnabish, "NVO3 Data Plane Requirements", draft-ietf-nvo3-dataplane-requirements-03 (work in progress), April 2014.

[IEEE.802.1Q-2014]
IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks", IEEE Std 802.1Q, 2014.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://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, <http://www.rfc-editor.org/info/rfc1981>.

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <http://www.rfc-editor.org/info/rfc3031>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <http://www.rfc-editor.org/info/rfc6935>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7365>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <http://www.rfc-editor.org/info/rfc7637>.

[VL2] Greenberg et al, , "VL2: A Scalable and Flexible Data Center Network", 2009.

Proc. ACM SIGCOMM 2009

Authors' Addresses

Jesse Gross (editor)

Email: jesse@kernel.org

Ilango Ganga (editor)
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
USA

Email: ilango.s.ganga@intel.com

T. Sridhar (editor)
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: tsridhar@vmware.com

NVO3 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

G. Mirsky
ZTE Corp.
N. Kumar
D. Kumar
Cisco Systems, Inc.
M. Chen
Y. Li
Huawei Technologies
D. Dolson
Sandvine

March 10, 2017

Echo Request and Echo Reply for Overlay Networks
draft-ooamdt-rtgwg-demand-cc-cv-03

Abstract

This document defines Overlay Echo Request and Echo Reply that enable on-demand Continuity Check, Connectivity Verification among other operations in overlay networks.

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 September 11, 2017.

Copyright Notice

Table of Contents

1. Introduction

Operations, Administration, and Maintenance (OAM) toolset provides methods for fault management and performance monitoring in each layer of the network, in order to improve their ability to support services with guaranteed and strict Service Level Agreements (SLAs) while reducing operational costs.

1.1. Conventions used in this document

1.1.1. Terminology

Term "Overlay OAM" used in this document interchangeably with longer version "set of OAM protocols, methods and tools for Overlay networks". And "Overlay ping" is used interchangeably with longer version Overlay Echo Request/Reply.

CC Continuity Check

CV Connectivity Verification

ECMP Equal Cost Multipath

FM Fault Management

Geneve Generic Network Virtualization Encapsulation

GUE Generic UDP Encapsulation

MPLS Multiprotocol Label Switching

NVO3 Network Virtualization Overlays

OAM Operations, Administration, and Maintenance

SFC Service Function Chaining

SFP Service Function Path

VXLAN Virtual eXtensible Local Area Network

VXLAN-GPE Generic Protocol Extension for VXLAN

1.1.2. 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 [RFC2119].

2. On-demand Continuity Check and Connectivity Verification

2.1. Requirements Towards On-demand CC/CV OAM

Availability, not as performance metric, is understood as ability to reach the node, i.e. the fact that path between ingress and egress does exist. Such OAM mechanism also referred as Continuity Check (CC). Connectivity Verification (CV) extends Continuity Check functionality in order to provide confirmation that the desired source is connected to the desired sink.

Echo Request/Reply OAM mechanism enables detection of the loss of continuity defect, its localization and collection information in order to discover root cause. These are requirements considered:

REQ#1: MUST support fault localization of Loss of Continuity check at Overlay layer.

REQ#2: MAY support fault localization of Loss of Continuity check at transport layer.

REQ#3: MUST support tracing path in overlay network through the overlay nodes.

REQ#4: MAY support tracing path in underlay network connecting overlay border nodes.

REQ#5: MAY support verification of the mapping between its data plane state and client layer services.

REQ#6: MUST have the ability to discover and exercise equal cost multipath (ECMP) paths in its underlay network.

REQ#7: MUST be able to trigger on-demand FM with responses being directed towards initiator of such proxy request.

2.2. Proposed Solution

The format of the Echo Request/Echo Reply control packet is to support ping and traceroute functionality in overlay networks Figure 1 resembles the format of MPLS LSP Ping [RFC4379] with some exceptions.

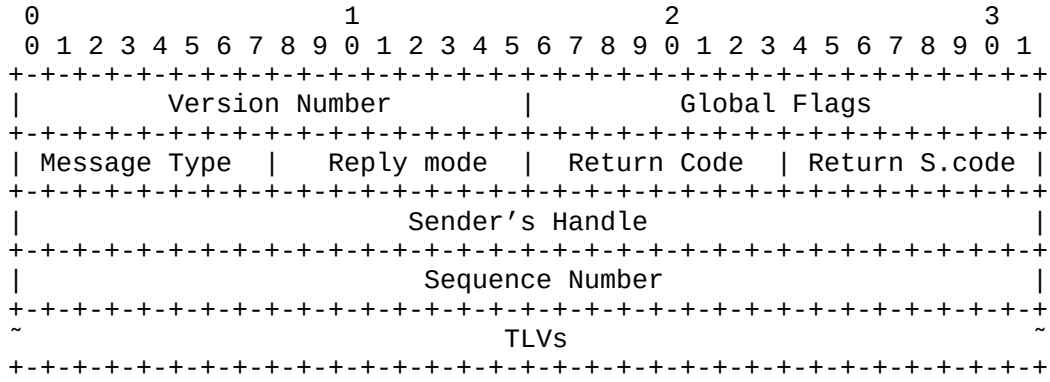


Figure 1: Overlay OAM Ping format

The interpretation of the fields is as following:

The Version reflects the current version. The version number is to be incremented whenever a change is made that affects the

ability of an implementation to correctly parse or process control packet.

The Global Flags is a bit vector field

The Message Type filed reflects the type of the packet. Value TBA2 identifies Echo Request and TBA3 - Echo Reply

The Reply Mode defines the type of the return path requested by the sender of the Echo Request.

Return Codes and Subcodes can be used to inform the sender about result of processing its request.

The Sender's Handle is filled in by the sender, and returned unchanged by the receiver in the Echo Reply.

The Sequence Number is assigned by the sender and can be (for example) used to detect missed replies.

TLVs (Type-Length-Value tuples) have the two octets long Type field, two octets long Length field that is length of the Value field in octets.

2.3. Overlay Echo Request Transmission

Overlay Echo Request control packet MUST use the appropriate encapsulation of the monitored overlay network. Overlay network encapsulation MUST identify Echo Request as OAM packet. Overlay encapsulation uses different methods to identify OAM payload [I-D.ietf-nvo3-vxlan-gpe], [I-D.ietf-nvo3-gue], [I-D.ietf-nvo3-geneve], [I-D.ietf-sfc-nsh],[I-D.ietf-bier-mpls-encapsulation]. Overlay network's header MUST be immediately followed by the Overlay OAM Header [I-D.ooamdt-rtgwg-ooam-header]. Message Type field in the Overlay OAM Header MUST be set to Overlay Echo Request value (TBA2).

Value of the Reply Mode field MAY be set to:

- Do Not Reply (TBA4) if one-way monitoring is desired. If Echo Request is used to measure synthetic packet loss, the receiver MAY report loss measurement results to a remote node.
- Reply via an IPv4/IPv6 UDP Packet (TBA5) value likely will be the most used.
- Reply via Application Level Control Channel (TBA6) value if the overlay network MAY have bi-directional paths.

- Reply via Specified Path (TBA7) value in order to enforce use of the particular return path specified in the included TLV to verify bi-directional continuity and also increase robustness of the monitoring by selecting more stable path.

2.4. Overlay Echo Request Reception

2.5. Overlay Echo Reply Transmission

The Reply Mode field directs whether and how the Echo Reply message should be sent. The sender of the Echo Request MAY use TLVs to request that corresponding Echo Reply be sent using the specified path. Value TBA3 is referred as "Do not reply" mode and suppresses transmission of Echo Reply packet. Default value (TBA5) for the Reply mode field requests the responder to send the Echo Reply packet out-of-band as IPv4 or IPv6 UDP packet. [Selection of destination and source IP addresses and UDP port numbers to be provided in the next update.]

2.6. Overlay Echo Reply Reception

3. IANA Considerations

3.1. Overlay Echo Request/Echo Reply Type

IANA is requested to assign new type from the Overlay OAM Protocol Types registry as follows:

Value	Description	Reference
TBA1	Overlay Echo Request/Echo Reply	This document

Table 1: Overlay Echo Request/Echo Reply Type

3.2. Overlay Ping Parameters

IANA is requested to create new Overlay Echo Request/Echo Reply Parameters registry.

3.3. Overlay Echo Request/Echo Reply Message Types

IANA is requested to create in the Overlay Echo Request/Echo Reply Parameters registry the new sub-registry Message Types. All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] and assign values as follows:

Value	Description	Reference
0	Reserved	
TBA2	Overlay Echo Request	This document
TBA3	Overlay Echo Reply	This document
TBA3+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 2: Overlay Echo Request/Echo Reply Message Types

3.4. Overlay Echo Reply Modes

IANA is requested to create in the Overlay Echo Request/Echo Reply Parameters registry the new sub-registry Reply Modes All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] and assign values as follows:

Value	Description	Reference
0	Reserved	
TBA4	Do Not Reply	This document
TBA5	Reply via an IPv4/IPv6 UDP Packet	This document
TBA6	Reply via Application Level Control Channel	This document
TBA7	Reply via Specified Path	This document
TBA7+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 3: Overlay Echo Reply Modes

4. Security Considerations

Overlay Echo Request/Reply operates within the domain of the overlay network and thus inherits any security considerations that apply to the use of that overlay technology and, consequently, underlay data plane. Also, the security needs for Overlay Echo Request/Reply are

similar to those of ICMP ping [RFC0792], [RFC4443] and MPLS LSP ping [I-D.ietf-mpls-rfc4379bis].

There are at least three approaches of attacking a node in the overlay network using the mechanisms defined in the document. One is a Denial-of-Service attack, by sending Overlay ping to overload a node in the overlay network. The second may use spoofing, hijacking, replying, or otherwise tampering with Overlay Echo Requests and/or Replies to misrepresent, alter operator's view of the state of the overlay network. The third is an unauthorized source using an Overlay Echo Request/Reply to obtain information about the overlay and/or underlay network.

To mitigate potential Denial-of-Service attacks, it is RECOMMENDED that implementations throttle the Overlay ping traffic going to the control plane.

Reply and spoofing attacks involving faking or replying Overlay Echo Reply messages would have to match the Sender's Handle and Sequence Number of an outstanding Overlay Echo Request message which is highly unlikely. Thus the non-matching reply would be discarded. But since "even a broken clock is right twice a day" implementations MAY use Timestamp control block [I-D.ooamdt-rtgwg-ooam-header] to validate the TimeStamp Sent by requiring an exact match on this field.

To protect against unauthorized sources trying to obtain information about the overlay and/or underlay an implementation MAY check that the source of the Echo Request is indeed part of the overlay domain.

5. Contributors

Work on this documented started by Overlay OAM Design Team with contributions from:

Carlos Pignataro

Cisco Systems, Inc.

cpignata@cisco.com

Santosh Pallagatti

santosh.pallagatti@gmail.com

Erik Nordmark

Arista Networks

nordmark@acm.org

Ignas Bagdonas

ibagdona@gmail.com

David Mozes

Mellanox Technologies Ltd.

davidm@mellanox.com

6. Acknowledgment

TBD

7. References

7.1. Normative References

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS and non-MPLS Networks", draft-ietf-bier-mpls-encapsulation-06 (work in progress), December 2016.

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-03 (work in progress), September 2016.

[I-D.ietf-nvo3-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-05 (work in progress), October 2016.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-03 (work in progress), October 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

[I-D.ooamdt-rtgwg-ooam-header]
Mirsky, G., Kumar, N., Kumar, D., Chen, M., Yizhou, L., Mozes, D., and D. Dolson, "OAM Header for use in Overlay Networks", draft-ooamdt-rtgwg-ooam-header-02 (work in progress), February 2017.

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

7.2. Informative References

[I-D.ietf-mpls-rfc4379bis]
Kompella, K., Swallow, G., Pignataro, C., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", draft-ietf-mpls-rfc4379bis-09 (work in progress), October 2016.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Nagendra Kumar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Deepak Kumar
Cisco Systems, Inc.

Email: dekumar@cisco.com

Mach Chen
Huawei Technologies

Email: mach.chen@huawei.com

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

David Dolson
Sandvine

Email: ddolson@sandvine.com

NVO3 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

G. Mirsky
ZTE Corp.
N. Kumar
D. Kumar
Cisco Systems, Inc.
M. Chen
Y. Li
Huawei Technologies
D. Dolson
Sandvine

March 10, 2017

OAM Header for use in Overlay Networks
draft-ooamdt-rtgwg-ooam-header-03

Abstract

This document introduces Overlay Operations, Administration, and Maintenance (OOAM) Header to be used in overlay networks to create Overlay Associated Channel (OAC) to ensure that OOAM control packets are in-band with user traffic and de-multiplex OOAM protocols.

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 September 11, 2017.

Copyright Notice

Table of Contents

1. Introduction

New protocols that support overlay networks like VxLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], GUE [I-D.ietf-nvo3-gue], Geneve [I-D.ietf-nvo3-geneve], BIER [I-D.ietf-bier-mpls-encapsulation], and NSH [I-D.ietf-sfc-nsh] support multi-protocol payload, e.g. Ethernet, IPv4/IPv6, and recognize Operations, Administration, and Maintenance (OAM) as one of distinct types. That ensures that Overlay OAM (OOAM)packets are sharing fate with Overlay data packet traversing the underlay.

This document introduces generic requirements to OAM protocols used in overlay networks and defines OOAM Header to be used in overlay networks to de-multiplex OOAM protocols.

1.1. Conventions used in this document

1.1.1. Terminology

Term "Overlay OAM" used in this document interchangeably with longer version "set of OAM protocols, methods and tools for Overlay networks".

NTP Network Time Protocol

OAC Overlay Associated Channel

OAM Operations, Administration, and Maintenance

OOAM Overlay OAM

PTP Precision Time Protocol

1.1.2. 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 [RFC2119].

2. General Requirements to OAM Protocols in Overlay Networks

OAM protocols, whether it is part of fault management or performance monitoring, intended to provide reliable information that can be used to identify defect, localize it and apply corrective actions. One of the main challenges that network operators may encounter is interpretations of reports of the defect or service degradation and correlation to affected services. In order to improve reliability of the correlation process we set forth the following requirements:

REQ#1: Overlay OAM packets SHOULD be fate sharing with data traffic, i.e. in-band with the monitored traffic, i.e. follow exactly the same overlay and transport path as data plane traffic, in forward direction, i.e. from ingress toward egress end point(s) of the OAM test.

REQ#2: Encapsulation of OAM control message and data packets in underlay network MUST be indistinguishable from underlay network forwarding point of view.

REQ#3: Presence of OAM control message in overlay packet MUST be unambiguously identifiable.

REQ#4: It MUST be possible to express entropy for underlay Equal Cost Multipath in overlay encapsulation in order to avoid using data packet content by underlay transient nodes.

3. Associated Channel in Overlay Networks

Associated channel in the overlay network is the channel that, by using the same encapsulation as user traffic, follows the same path through the underlay network as user traffic. In other words, the associated channel is in-band with user traffic. Creating notion of the overlay associated channel (OAC) in the overlay network ensures that control packets of active OAM protocols carried in the OAC are in-band with user traffic. Additionally, OAC allows development of OAM tools that, from operational point of view, function in essentially the same manner in any type of overlay.

4. Overlay OAM Header

OOAM Header immediately follows the header of the overlay and identifies OAC. The format of the OOAM Header is:

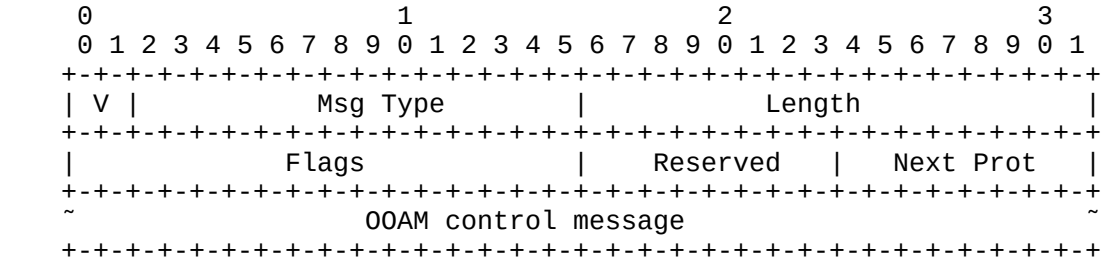


Figure 1: Overlay OAM Header format

The OAM Header consists of the following fields:

- V - two bits long field indicates the current version of the Overlay OAM Header. The current value is 0;
- Msg Type - 14 bits long field identifies OAM protocol, e.g. Echo Request/Reply, BFD, Performance Measurement;
- Length - two octets long field that is length of the OOAM control packet in octets;
- Flags -two octets long field carries bit flags that define optional capability and thus processing of the OOAM control packet;

- Reserved - one octet field that MUST be zeroed on transmit and ignored on receipt;

- Next Prot - one octet long field that defines optional payload that is present after the OOAM Control Packet.

The format of the Flags field is:

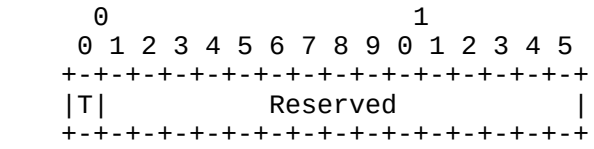


Figure 2: Flags field format

where:

- T - Timestap block flag.

- Reserved - must be set to all zeroes on transmission and ignored on receipt.

The OOAM header may be followed by the Timestamp control block Figure 3 and then by OOAM Control Packet identified by the Msg Type field.

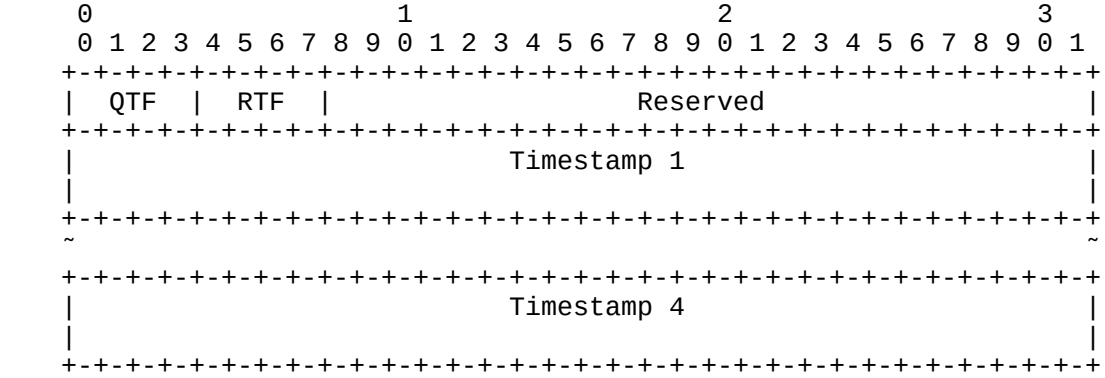


Figure 3: Timestamp block format

where:

QTF - Querier timestamp format

RTF - Responder timestamp format

Timestamp 1-4 - 64-bit timestamp values

Network Time Protocol (NTP), described in [RFC5905], is widely used and has long history of deployment. But it is the IEEE 1588 Precision Time Protocol (PTP) [IEEE.1588.2008] that is being broadly used to achieve high-quality clock synchronization. Converging between NTP and PTP time formats is possible but is not trivial and does come with cost, particularly when it is required to be performed in real time without loss of accuracy. And recently protocols that supported only NTP time format, like One-Way Active Measurement Protocol [RFC4656] and Two-Way Active Measurement Protocol [RFC5357], have been enhanced to support the PTP time format as well [I-D.ietf-ippm-twamp-time-format]. This document proposes to select PTP time format as default time format for Overlay OAM performance measurement. Hence QTF, RTF fields MUST be set to 0 if querier or responder use PTP time format respectively. If the querier or responder use the NTP time format, then QTF and/or RTF MUST be set to 1. Use of other values MUST be considered as error and MAY be reported.

4.1. Use of OOAM Header in Active OAM

Active OAM methods, whether used for fault management or performance monitoring, generate dedicated test packets [RFC7799]. Format of an OAM test packet in overlay network presented in Figure 4.

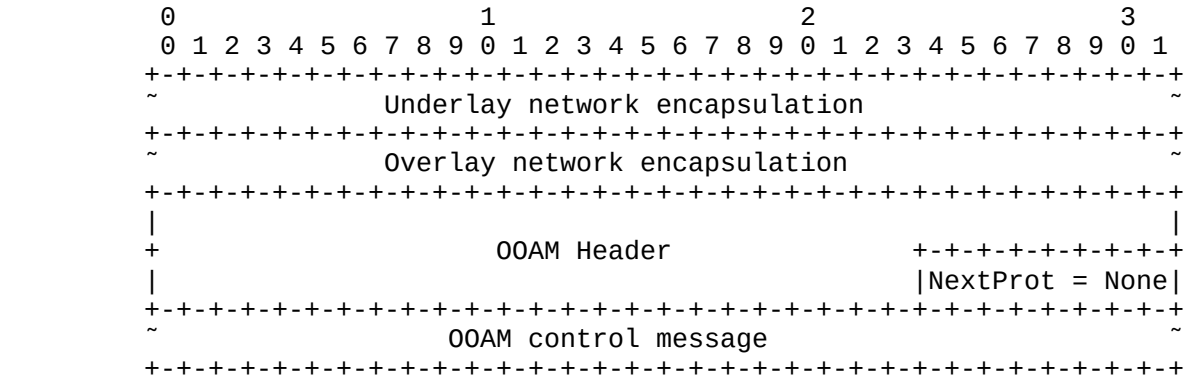


Figure 4: Overlay OAM Header in Active OAM Control Packet

Because active OAM method uses only OAM protocol value of Next Prot field in the OOAM header is set to None indicating that there’s no content from other protocol immediately after OOAM control message in the packet.

4.2. Use of OOAM Header in Hybrid OAM

Hybrid OAM Type I methods, whether used for fault management or performance monitoring, modify user data packets [RFC7799]. Format of such modified packet in overlay network presented in Figure 5.

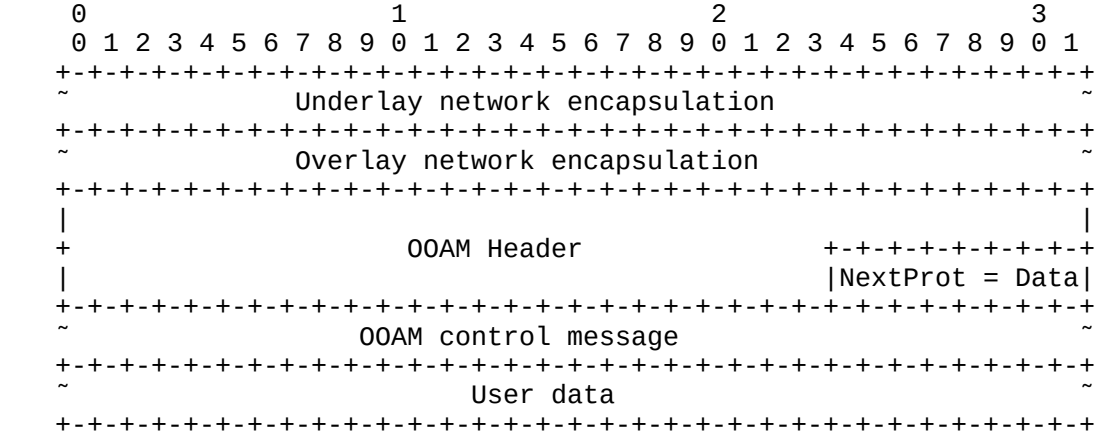


Figure 5: Overlay OAM Header in Hybrid OAM Control Packet

In case when OOAM header used for Hybrid Type I OAM method value of the Next Prot field is set to the value associated with the protocol of the user data.

5. IANA Considerations

IANA is requested to create new registry called "Overlay OAM".

5.1. OOAM Message Types

IANA is requested to create new sub-registry called "Overlay OAM Protocol Types" in the "Overlay OAM" registry. All code points in the range 1 through 15615 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] . Remaining code points are allocated according to the Table 1:

Value	Description	Reference
0	Reserved	
1 - 15615	Unassigned	IETF Review
15616 - 16127	Unassigned	First Come First Served
16128 - 16143	Experimental	This document
16144 - 16382	Private Use	This document
16383	Reserved	This document

Table 1: Overlay OAM Protocol type

5.2. OOAM Header Flags

IANA is requested to create sub-registry "Overlay OAM Header Flags" in "Overlay OAM" registry. Two flags are defined in this document. New values are assigned via Standards Action [RFC5226].

Flags bit	Description	Reference
Bit 0	Timestamp field	This document
Bit 1-15	Unassigned	

Table 2: Overlay OAM Flags

6. Security Considerations

TBD

7. Contributors

Work on this documented started by Overlay OAM Design Team with contributions from:

Carlos Pignataro

Cisco Systems, Inc.

cpignata@cisco.com

Erik Nordmark

Arista Networks

nordmark@acm.org

Ignas Bagdonas

ibagdona@gmail.com

David Mozes

Mellanox Technologies Ltd.

davidm@mellanox.com

8. Acknowledgement

TBD

9. References

9.1. Normative References

[IEEE.1588.2008]
"Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Standard 1588, July 2008.

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

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

9.2. Informative References

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS and non-MPLS Networks", draft-ietf-bier-mpls-encapsulation-06 (work in progress), December 2016.

[I-D.ietf-ippm-twamp-time-format]
Mirsky, G. and I. Meilik, "Support of IEEE-1588 time stamp format in Two-Way Active Measurement Protocol (TWAMP)", draft-ietf-ippm-twamp-time-format-05 (work in progress), March 2017.

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-03 (work in progress), September 2016.

[I-D.ietf-nvo3-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-05 (work in progress), October 2016.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-03 (work in progress), October 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <http://www.rfc-editor.org/info/rfc4656>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <http://www.rfc-editor.org/info/rfc5357>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <http://www.rfc-editor.org/info/rfc7799>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Nagendra Kumar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Deepak Kumar
Cisco Systems, Inc.

Email: dekumar@cisco.com

Mach Chen
Huawei Technologies

Email: mach.chen@huawei.com

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

David Dolson
Sandvine

Email: ddolson@sandvine.com