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

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