Transport Area Working Group
Internet-Draft
Updates: 6040, 2661, 1701, 2784, 2637, 3931 (if approved)
Intended status: Standards Track
Expires: January 9, 2017

B. Briscoe
Simula Research Laboratory

July 8, 2016

Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim
draft-briscoe-tsvwg-rfc6040bis-01

Abstract

RFC 6040 on "Tunnelling of Explicit Congestion Notification" made the rules for propagation of ECN consistent for all forms of IP in IP tunnel. This specification extends the scope of RFC 6040 to include tunnels where two IP headers are separated by a shim header that cannot stand alone.

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 January 9, 2017.

Copyright Notice

Table of Contents

1. Scope of RFC 6040

RFC 6040 on "Tunnelling of Explicit Congestion Notification" [RFC6040] made the rules for propagation of Explicit Congestion Notification (ECN [RFC3168]) consistent for all forms of IP in IP tunnel. The scope of RFC 6040 was stated as

> "...ECN field processing at encapsulation and decapsulation for any IP-in-IP tunnelling, whether IPsec or non-IPsec tunnels. It applies irrespective of whether IPv4 or IPv6 is used for either the inner or outer headers. ..."

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. To clear up confusion, this specification clarifies that the scope of RFC 6040 includes any IP-in-IP tunnel, including those with shim header(s) between the IP headers.

2. Terminology

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

3. IP-in-IP Tunnels with Tightly Coupled Shim Headers

In many cases the shim header(s) and the outer IP header are always added (or removed) as part of the same process. We call this a tightly coupled shim header. Processing the shim and outer together is often necessary because the shim(s) are not sufficient for packet forwarding in their own right; not unless complemented by an outer header.

For all such tightly coupled shim headers, the rules in [RFC6040] for propagating the ECN field SHOULD be applied directly between the inner and outer IP headers. This specification therefore updates the following specifications of tightly coupled shim headers by adding that RFC 6040 SHOULD apply when the shim header is used between IP headers:

- L2TPv2 [RFC2661], L2TPv3 [RFC3931]
- GRE [RFC1701], [RFC2784]
- PPTP [RFC2637]
- GTP [GTPv1], [GTPv1-U], [GTPv2-C]
- VXLAN [RFC7348].

Geneve [I-D.ietf-nvo3-geneve] and Generic UDP Encapsulation (GUE) [I-D.ietf-nvo3-gue] are also tightly coupled shim headers, but their specifications already refer to RFC 6040 for ECN encapsulation.

The above is written as a 'SHOULD' not a 'MUST' to allow for the possibility that the structure of some pre-existing tunnel implementations might make it hard to predict what other headers will be added or removed subsequently.

Although the definition of the various GTP shim headers is under the control of the 3GPP, it is hard to determine whether the 3GPP or the IETF controls standardization of the _process_ of adding both a GTP and an IP header to an inner IP header. Nonetheless, the present specification is provided so that the 3GPP can refer to it from any of its own specifications of GTP and IP header processing.

Similarly, VXLAN is not under the control of the IETF, but the present specification is provided so that the authors of any future update to the VXLAN specification can refer to it.

More generally, whatever form IP-in-IP tunnelling takes, the ECN field SHOULD be propagated according to the rules in RFC 6040 wherever possible. Otherwise [I-D.ietf-tsvwg-ecn-encap-guidelines] gives more general guidance on how to propagate ECN to and from protocols that encapsulate IP.

4. IANA Considerations (to be removed by RFC Editor)

This memo includes no request to IANA.

5. Security Considerations

The Security Considerations in RFC 6040 apply equally to the wider scope defined by the present specification.

6. Comments Solicited

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group mailing list <tsvwg@ietf.org>, and/or to the authors.

7. Normative References

[GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface", Technical Specification TS 29.060.

[GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.

[GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.

[I-D.ietf-nvo3-geneve] Gross, J. and I. Ganga, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-01 (work in progress), January 2016.

[I-D.ietf-nvo3-gue] Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-04 (work in progress), July 2016.

[I-D.ietf-tsvwg-ecn-encap-guidelines] Briscoe, B., Kaippallimalil, J., and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", draft-ietf-tsvwg-ecn-encap-guidelines-06 (work in progress), July 2016.

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <http://www.rfc-editor.org/info/rfc1701>.

[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>.

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <http://www.rfc-editor.org/info/rfc2637>.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <http://www.rfc-editor.org/info/rfc2661>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <http://www.rfc-editor.org/info/rfc3931>.

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

[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>.

Author’s Address

Bob Briscoe
Simula Research Laboratory
UK

EMail: ietf@bobbriscoe.net
URI: http://bobbriscoe.net/

INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: December 2016

T. Herbert
Facebook
L. Yong
Huawei
F. Templin
Boeing

June 21, 2016

Extensions for Generic UDP Encapsulation
draft-herbert-gue-extensions-00

Abstract

This specification defines a set of the fundamental extensions for Generic UDP Encapsulation (GUE). The extensions defined in this specification are the security option, payload transform option, checksum option, fragmentation option, and the remote checksum offload option.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

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

Copyright and License Notice

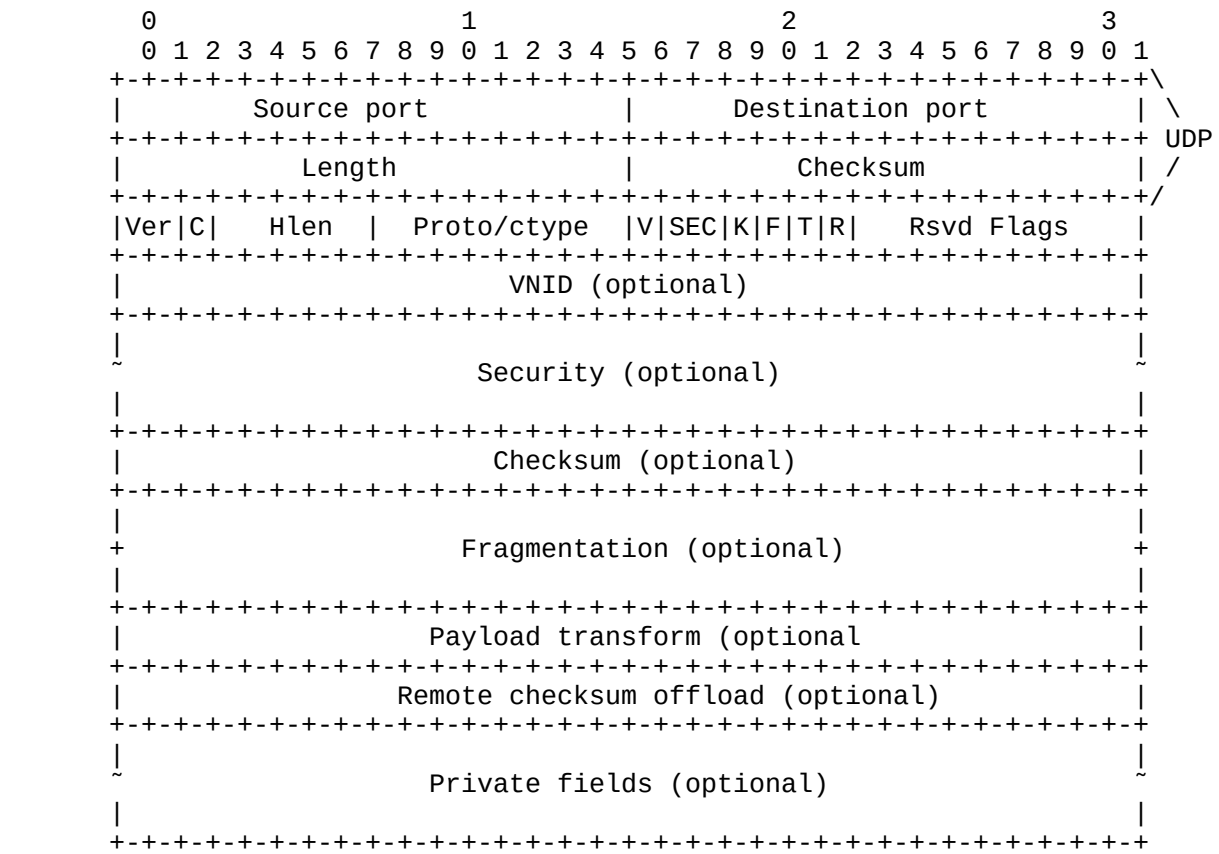
Table of Contents

1. Introduction

Generic UDP Encapsulation [I.D.nvo3-gue] is a generic and extensible encapsulation protocol. This specification defines a basic set of extensions for GUE. These extensions are the security option, payload transform option, checksum option, fragmentation option, and the remote checksum offload option.

2. GUE header format with options

The general format of GUE with the options defined in this specification is:



The contents of the UDP are described in [I.D.herbert-gue].

The GUE header consists of:

- Ver: Version. Set to 0x0 to indicate GUE encapsulation header.

Note that version 0x1 does not allow options.

- C: Control bit. May be set or unset. GUE options can be used with either control or data packets unless otherwise specified in the option definition.

- Hlen: Length in 32-bit words of the GUE header, including optional fields but not the first four bytes of the header. Computed as (header_len - 4) / 4. The length of the encapsulated packet is determined from the UDP length and the Hlen: encapsulated_packet_length = UDP_Length - 8 - GUE_Hlen.

- Proto/ctype: If the C bit is not set this indicates IP protocol number for the packet in the payload, else if the C bit is set this type of control message for the payload. The next header begins at the offset provided by Hlen. When the fragmentation option or payload transform option is used this field may be set to protocol number 59 for a data message, or a value of 0 for a control message, to indicate no next header for the payload.

- V: Indicates the network virtualization option (VNID) field is present. The VNID option is described in [I.D.hy-nvo3-gue-4-nvo].

- SEC: Indicates security option field is present. The security option is described in section 5.

- K: Indicates checksum option field is present. The checksum option is described in section 3.

- F: Indicates fragmentation option field is present. The fragmentation option is described in section 4.

- T: Indicates transform option field is present. The transform option is described in section 5.

- R: Indicates the remote checksum option field is present. The remote checksum offload option is described in section 6.

- Private fields are described in [I.D.nvo3-gue].

3. Checksum option

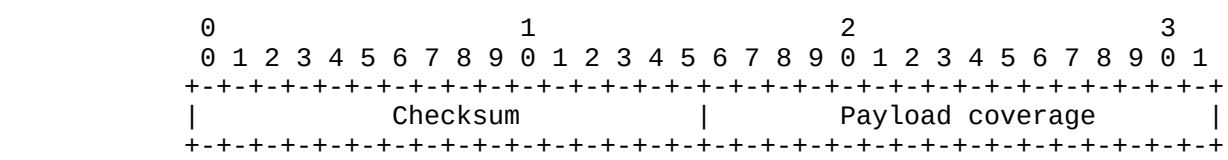
The GUE checksum option provides a checksum that covers the GUE header, a GUE pseudo header, and optionally part or all of the GUE payload. The GUE pseudo header includes the corresponding IP addresses as well as the UDP ports of the encapsulating headers. This

checksum should provide adequate protection against address corruption in IPv6 when the UDP checksum is zero. Additionally, the GUE checksum provides protection of the GUE header when the UDP checksum is set to zero with either IPv4 or IPv6. In particular, the GUE checksum can provide protection for some sensitive data, such as the virtual network identifier ([I.D.hy-nvo3-gue-4-nvo]), which when corrupted could lead to mis-delivery of a packet to the wrong virtual network.

3.1. Option format

The presence of the GUE checksum option is indicated by the K bit in the GUE header.

The format of the checksum option is:



The fields of the option are:

- Checksum: Computed checksum value. This checksum covers the GUE header (including fields and private data covered by Hlen), the GUE pseudo header, and optionally all or part of the payload (encapsulated packet).

- Payload coverage: Number of bytes of payload to cover in the checksum. Zero indicates that the checksum only covers the GUE header and GUE pseudo header. If the value is greater than the encapsulated payload length, the packet must be dropped.

3.2. Requirements

The GUE header checksum must be set on transmit when using a zero UDP checksum with IPv6.

The GUE header checksum must be set when the UDP checksum is zero for IPv4 if the GUE header includes data that when corrupted can lead to misdelivery or other serious consequences, and there is no other mechanism that provides protection (no security field for instance). Otherwise the GUE header checksum should be used with IPv4 when the UDP checksum is zero.

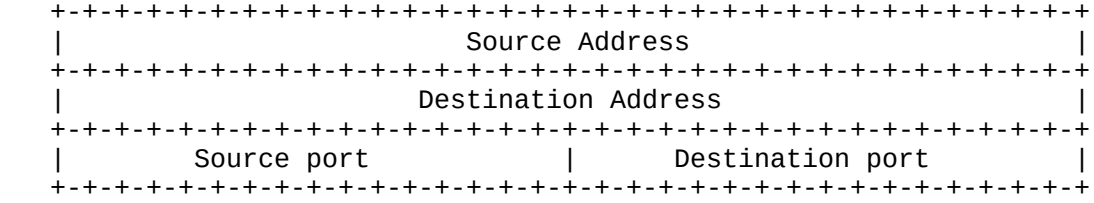
The GUE header checksum should not be set when the UDP checksum is non-zero. In this case the UDP checksum provides adequate protection

and this avoids convolutions when a packet traverses NAT that does address translation (in that case the UDP checksum is required).

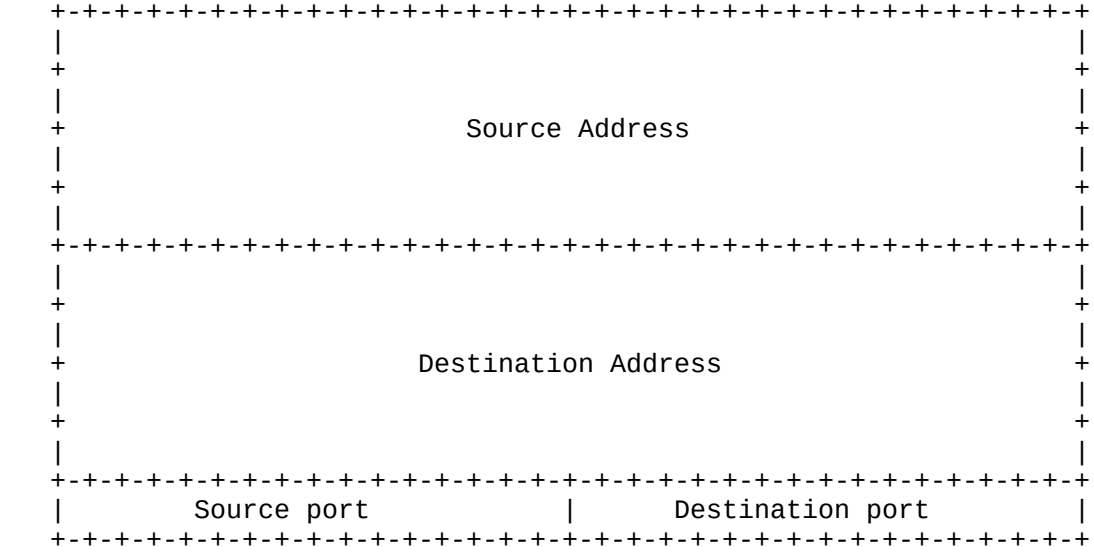
3.3. GUE checksum pseudo header

The GUE pseudo header checksum is included in the GUE checksum to provide protection for the IP and UDP header elements which when corrupted could lead to misdelivery of the GUE packet. The GUE pseudo header checksum is similar to the standard IP pseudo header defined in [RFC0768] and [RFC0793] for IPv4, and in [RFC2460] for IPv6.

The GUE pseudo header for IPv4 is:



The GUE pseudo header for IPv6 is:



Note that the GUE pseudo header does not include payload length or protocol as in the standard IP pseudo headers. The length field is deemed unnecessary because:

- If the length is corrupted this will usually be detected by a

checksum validation failure on the inner packet.

- Fragmentation of packets in a tunnel should occur on the inner packet before being encapsulated or GUE fragmentation (section 4) may be performed at tunnel ingress. GUE packets are not expected to be fragmented when using IPv6. See RFC6936 for considerations of payload length and IPv6 checksum.

- A corrupted length field in itself should not lead to misdelivery of a packet.

- Without the length field, the GUE pseudo header checksum is the same for all packets of flow. This is a useful property for optimizations such as TCP Segment Offload (TSO).

3.4. Checksum computation

The GUE checksum is computed and verified following the standard process for computing the Internet checksum [RFC1071]. Checksum computation may be optimized per the mathematical properties including parallel computation and incremental updates.

3.5. Transmitter operation

The procedure for setting the GUE checksum on transmit is:

1) Create the GUE header including the checksum and payload coverage fields. The checksum field is initially set to zero.

2) Calculate the 1's complement checksum of the GUE header from the start of the GUE header through the its length as indicated in GUE Hlen.

3) Calculate the checksum of the GUE pseudo header for IPv4 or IPv6.

4) Calculate checksum of payload portion if payload coverage is enabled (payload coverage field is non-zero). If the length of the payload coverage is odd, logically append a single zero byte for the purposes of checksum calculation.

5) Add and fold the computed checksums for the GUE header, GUE pseudo header and payload coverage. Set the bitwise not of the result in the GUE checksum field.

3.6. Receiver operation

If the GUE checksum option is present, the receiver must validate the

checksum before processing any other fields or accepting the packet.

The procedure for verifying the checksum is:

1) If the payload coverage length is greater than the length of the encapsulated payload then drop the packet.

2) Calculate the checksum of the GUE header from the start of the header to the end as indicated by Hlen.

3) Calculate the checksum of the appropriate GUE pseudo header.

4) Calculate the checksum of payload if payload coverage is enabled (payload coverage is non-zero). If the length of the payload coverage is odd logically append a single zero byte for the purposes of checksum calculation.

5) Sum the computed checksums for the GUE header, GUE pseudo header, and payload coverage. If the result is all 1 bits (-0 in 1's complement arithmetic), the checksum is valid and the packet is accepted; otherwise the checksum is considered invalid and the packet must be dropped.

3.7. Security Considerations

The checksum option is only a mechanism for corruption detection, it is not a security mechanism. To provide integrity checks or authentication of the GUE header, the GUE security option should be used.

4. Fragmentation option

The fragmentation option allows an encapsulator to perform fragmentation of packets being ingress to a tunnel. Procedures for fragmentation and reassembly are defined in this section. This specification adapts the procedures for IP fragmentation and reassembly described in [RFC0791] and [RFC2460]. Fragmentation may be performed on both data and control messages in GUE.

4.1. Motivation

This section describes the motivation for having a fragmentation option in GUE.

MTU and fragmentation issues with In-the-Network Tunneling are described in [RFC4459]. Considerations need to be made when a packet is received at a tunnel ingress point which may be too large to traverse the path between tunnel endpoints.

There are four suggested alternatives in [RFC4459] to deal with this:

1) Fragmentation and Reassembly by the Tunnel Endpoints

2) Signaling the Lower MTU to the Sources

3) Encapsulate Only When There is Free MTU

4) Fragmentation of the Inner Packet

Many tunneling protocol implementations have assumed that fragmentation should be avoided, and in particular alternative #3 seems preferred for deployment. In this case, it is assumed that an operator can configure the MTUs of links in the paths of tunnels to ensure that they are large enough to accommodate any packets and required encapsulation overhead. This method, however, may not be feasible in certain deployments and may be prone to misconfiguration in others.

Similarly, the other alternatives have drawbacks that are described in [RFC4459]. Alternative #2 implies use of something like Path MTU Discovery which is not known to be sufficiently reliable. Alternative #4 is not permissible with IPv6 or when the DF bit is set for IPv4, and it also introduces other known issues with IP fragmentation.

For alternative #1, fragmentation and reassembly at the tunnel endpoints, there are two possibilities: encapsulate the large packet and then perform IP fragmentation, or segment the packet and then encapsulate each segment (a non-IP fragmentation approach).

Performing IP fragmentation on an encapsulated packet has the same issues as that of normal IP fragmentation. Most significant of these is that the Identification field is only sixteen bits in IPv4 which introduces problems with wraparound as described in [RFC4963].

The second possibility follows the suggestion expressed in [RFC2764] and the fragmentation feature described in the AERO protocol [I.D.templin-aerolink], that is for the tunneling protocol itself to incorporate a segmentation and reassembly capability that operates at the tunnel level. In this method fragmentation is part of the encapsulation and an encapsulation header contains the information for reassembly. This is different from IP fragmentation in that the IP headers of the original packet are not replicated for each fragment.

Incorporating fragmentation into the encapsulation protocol has some advantages:

- o At least a 32 bit identifier can be defined to avoid issues of the 16 bit Identification in IPv4.

- o Encapsulation mechanisms for security and identification, such as virtual network identifiers, can be applied to each segment.

- o This allows the possibility of using alternate fragmentation and reassembly algorithms (e.g. fragmentation with Forward Error Correction).

- o Fragmentation is transparent to the underlying network so it is unlikely that fragmented packet will be unconditionally dropped as might happen with IP fragmentation.

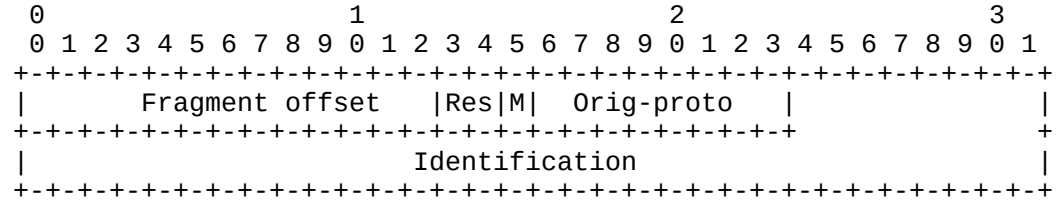
4.2. Scope

This specification describes the mechanics of fragmentation in Generic UDP Encapsulation. The operational aspects and details for higher layer implementation must be considered for deployment, but are considered out of scope for this document. The AERO protocol [I.D.templin-aerolink] defines one use case of fragmentation with encapsulation.

4.3. Option format

The presence of the GUE fragmentation option is indicated by the F bit in the GUE header.

The format of the fragmentation option is:



The fields of the option are:

- o Fragment offset: This field indicates where in the datagram this fragment belongs. The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.

- o Res: Two bit reserved field. Must be set to zero for transmission. If set to non-zero in a received packet then the packet MUST be dropped.

- M: More fragments bit. Set to 1 when there are more fragments following in the datagram, set to 0 for the last fragment.

- Orig-proto: The control type (when C bit is set) or the IP protocol (when C bit is not set) of the fragmented packet.

- Identification: 40 bits. Identifies fragments of a fragmented packet.

Pertinent GUE header fields to fragmentation are:

- C: This bit is set for each fragment based on the whether the original packet being fragmented is a control or data message.

- Proto/ctype - For the first fragment (fragment offset is zero) this is set to that of the original packet being fragmented (either will be a control type or IP protocol). For other fragments, this is set to zero for a control message being fragmented, or to "No next header" (protocol number 59) for a data message being fragmented.

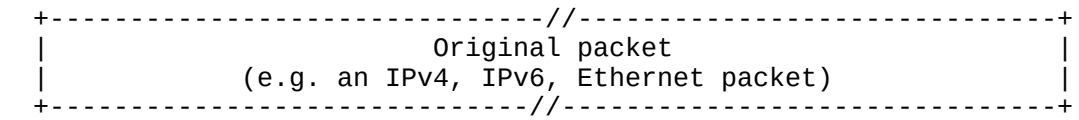
- F bit - Set to indicate presence of the fragmentation option field.

4.4. Fragmentation procedure

If an encapsulator determines that a packet must be fragmented (eg. the packet's size exceeds the Path MTU of the tunnel) it should divide the packet into fragments and send each fragment as a separate GUE packet, to be reassembled at the decapsulator (tunnel egress).

For every packet that is to be fragmented, the source node generates an Identification value. The Identification must be different than that of any other fragmented packet sent within the past 60 seconds (Maximum Segment Lifetime) with the same tunnel identification-- that is the same outer source and destination addresses, same UDP ports, same orig-proto, and same virtual network identifier if present.

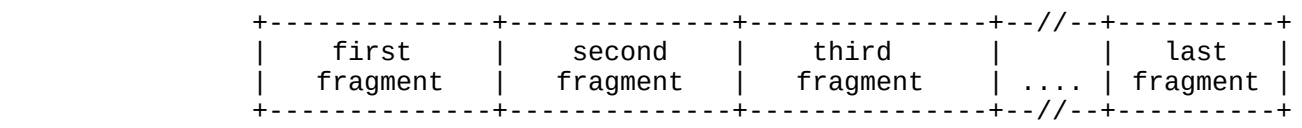
The initial, unfragmented, and unencapsulated packet is referred to as the "original packet". This will be a layer 2 packet, layer 3 packet, or the payload of a GUE control message:



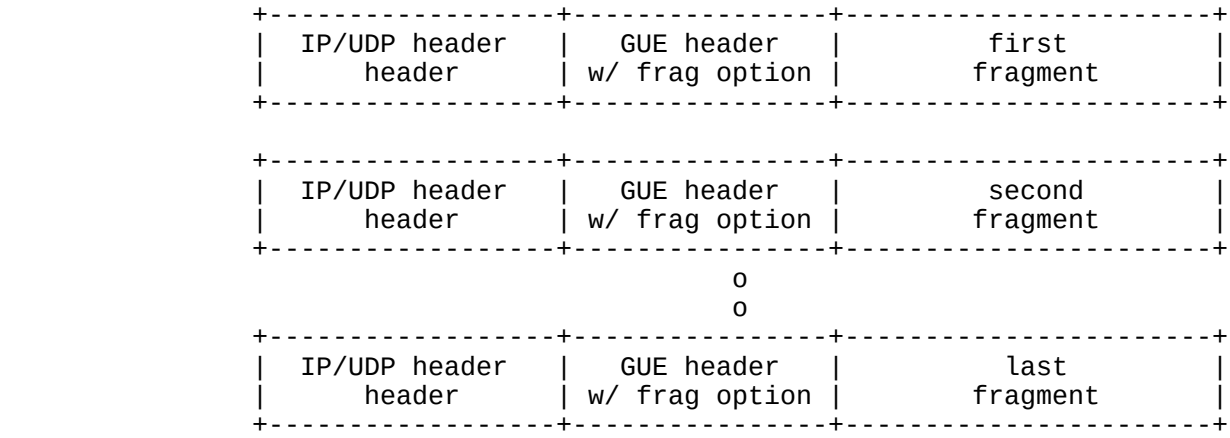
Fragmentation and encapsulation are performed on the original packet

in sequence. First the packet is divided up in to fragments, and then each fragment is encapsulated. Each fragment, except possibly the last ("rightmost") one, is an integer multiple of 8 octets long. Fragments MUST be non-overlapping. The number of fragments should be minimized, and all but the last fragment should be approximately equal in length.

The fragments are transmitted in separate "fragment packets" as:



Each fragment is encapsulated as the payload of a GUE packet. This is illustrated as:



Each fragment packet is composed of:

(1) Outer IP and UDP headers as defined for GUE encapsulation.

- The IP addresses and UDP destination port must be the same for all fragments of a fragmented packet.
- The source port selected for the inner flow identifier must be the same value for all fragments of a fragmented packet.

(2) A GUE header that contains:

- The C bit which is set to the same value for all the fragments of a fragmented packet based on whether a control message or data message was fragmented.

- o A proto/ctype. In the first fragment this is set to the value corresponding to the next header of the original packet and will be either an IP protocol or a control type. For subsequent fragments, this field is set to 0 for a fragmented control message or 59 (no next header) for a fragmented data messages.

- o The F bit is set and fragment option is present.

- o Other GUE options. Note that options apply to the individual GUE packet. For instance, the security option would be validated before reassembly.

(3) The GUE fragmentation option. The option contents include:

- o Orig-proto that identifies the first header of the original packet.

- o A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the of the original packet. The Fragment Offset of the first ("leftmost") fragment is 0.

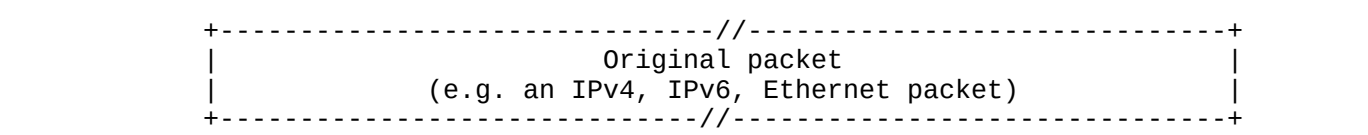
- o An M flag value of 0 if the fragment is the last ("rightmost") one, else an M flag value of 1.

- o The Identification value generated for the original packet.

(4) The fragment itself.

4.5. Reassembly procedure

At the destination, fragment packets are decapsulated and reassembled into their original, unfragmented form, as illustrated:



The following rules govern reassembly:

The IP/UDP/GUE headers of each packet are retained until all fragments have arrived. The reassembled packet is then composed of the decapsulated payloads in the GUE fragments, and the IP/UDP/GUE headers are discarded.

When a GUE packet is received with the fragment option, the proto/ctype in the GUE header must be validated. In the case that the packet is a first fragment (fragment offset is zero), the proto/ctype in the GUE header must equal the orig-proto value in the fragmentation option. For subsequent fragments (fragment offset is non-zero) the proto/ctype in the GUE header must be 0 for a control message or 59 (no-next-hdr) for a data message. If the proto/ctype value is invalid then for a received packet it MUST be dropped.

An original packet is reassembled only from GUE fragment packets that have the same outer Source Address, Destination Address, UDP source port, UDP destination port, GUE header C bit, virtual network identifier if present, orig-proto value in the fragmentation option, and Fragment Identification. The protocol type or control message type (depending on the C bit) for the reassembled packet is the value of the GUE header proto/ctype field in the first fragment.

The following error conditions may arise when reassembling fragmented packets with GUE encapsulation:

If insufficient fragments are received to complete reassembly of a packet within 60 seconds (or a configurable period) of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded.

If the payload length of a fragment is not a multiple of 8 octets and the M flag of that fragment is 1, then that fragment must be discarded.

If the length and offset of a fragment are such that the payload length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded.

If a fragment overlaps another fragment already saved for reassembly then the new fragment that overlaps the existing fragment MUST be discarded

If the first fragment is too small then it is possible that it does not contain the necessary headers for a stateful firewall. Sending small fragments like this has been used as an attack on IP fragmentation. To mitigate this problem, an implementation should ensure that the first fragment contains the headers of the encapsulated packet at least through the transport header.

A GUE node must be able to accept a fragmented packet that, after reassembly and decapsulation, is as large as 1500 octets. This means that the node must configure a reassembly buffer that is at least as large as 1500 octets plus the maximum-sized encapsulation headers that may be inserted during encapsulation. Implementations may find it more convenient and efficient to configure a reassembly buffer size of 2KB which is large enough to accommodate even the largest set of encapsulation headers and provides a natural memory page size boundary.

4.6. Security Considerations

Exploits that have been identified with IP fragmentation are conceptually applicable to GUE fragmentation.

Attacks on GUE fragmentation can be mitigated by:

- Hardened implementation that applies applicable techniques from implementation of IP fragmentation.
- Application of GUE security (section 5) or IPsec [RFC4301]. Security mechanisms can prevent spoofing of fragments from unauthorized sources.
- Implement fragment filter techniques for GUE encapsulation as described in [RFC1858] and [RFC3128].
- Do not accepted data in overlapping segments.
- Enforce a minimum size for the first fragment.

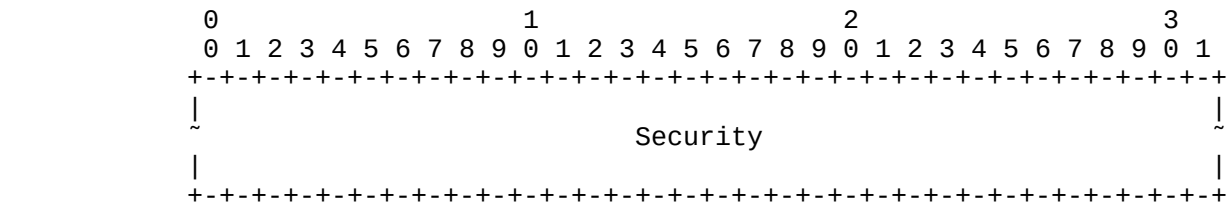
5. Security and payload transform options

The security option and the payload transform option are used to provide security for the GUE headers and payload. The GUE security option provides origin authentication and integrity protection of the GUE header at tunnel end points to guarantee isolation between tunnels and mitigate Denial of Service attacks. The payload transform option provides a means to perform encryption and authentication of the GUE packet that protects the payload from eavesdropping, tampering, or message forgery.

5.1. Security option format

The presence of the GUE security option is indicated in the SEC flag bits of the GUE header.

The format of the fragmentation option is:



The fields of the option are:

- Security (8, 16, or 32 octets). Contains the security information. The specific semantics and format of this field is expected to be negotiated between the two communicating nodes.

To provide security capability, the SEC flags MUST be set. Different sizes are allowed to provide different methods and extensibility. The use of the security field is expected to be negotiated out of band between two tunnel end points.

The possible values in the SEC flags are:

- 00 - No security field
- 01 - 64 bit security field
- 10 - 128 bit security field
- 11 - 256 bit security field

5.2. Security option usage

The GUE security field should be used to provide integrity and authentication of the GUE header. Security negotiation (interpretation of security field, key management, etc.) is expected to be negotiated out of band between two communicating hosts. Two possible uses for this field are outlined below, a more precise specification is deferred to other documents.

5.2.1. Cookies

The security field may be used as a cookie. This would be similar to cookie mechanism described in L2TP [RFC3931], and the general properties should be the same. The cookie may be used to validate the encapsulation. The cookie is a shared value between an encapsulator and decapsulator which should be chosen randomly and may be changed periodically. Different cookies may used for logical flows between the encapsulator and decapsulator, for instance packets sent with different VNIs in network virtualization [I.D.hy-nvo3-gue-4-nvo]

might have different cookies.

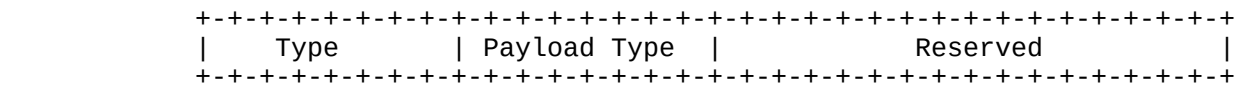
5.2.2. Secure hash

Strong authentication of the GUE header can be provided using a secure hash. This may follow the model of the TCP authentication option [RFC5925]. In this case the security field holds a message digest for the GUE header (e.g. 16 bytes from MD5). The digest might be done over static fields in IP and UDP headers per negotiation (addresses, ports, and protocols). In order to provide enough entropy, a random salt value in each packet might be added, for instance the security field could be a 256 bit value that contains 128 bits of salt value, and a 128 bit digest value. The use of secure hashes requires shared keys which are presumably negotiated and rotated as needed out of band.

5.3. Payload Transform Option format

The presence of the GUE payload transform option is indicated by the T bit in the GUE header.

The format of Payload Transform Field is:



The fields of the option are:

Type: Payload Transform Type or Code point. Each payload transform mechanism must have one code point registered in IANA. This document specifies:

0x01: for DTLS [RFC6347]

0x80~0xFF: for private payload transform types

A private payload transform type can be used for experimental purpose or vendor proprietary mechanisms.

Payload Type: Indicates the encrypted payload type. When payload transform option is present, proto/ctype in the base header should set to 59 ("No next header") for a data message and zero for a control message. The payload type (IP protocol or control message type) of the unencrypted payload must be encoded in this field.

The benefit of this rule is to prevent a middle box from

inspecting the encrypted payload according to GUE next protocol. The assumption here is that a middle box may understand GUE base header but does not understand GUE option flag definitions.

Reserved field for DTLS type MUST set to Zero. For other transformation types, the use of reserved field may be specified.

5.3.1. Payload transform option usage

The payload transform option provides a mechanism to transform or interpret the payload of a GUE packet. The Type field describes the format of the payload before transformation and Payload Type provides the protocol of the packet after transformation. The payload transformation option is generic so that it can have both security related uses (such as DTLS) as well as non security related uses (such as compression, CRC, etc.).

The payload of a GUE packet can be secured using Datagram Transport Layer Security [RFC6347]. An encapsulator would apply DTLS to the GUE payload so that the payload packets are encrypted and the GUE header remains in plaintext. The payload transform option is set to indicate that the payload should be interpreted as a DTLS record.

5.4. Operation of security mechanisms

GUE secure transport mechanisms apply to both IPv4 and IPv6 underlay networks. The outer IP address MUST be tunnel egress IP address (dst) and tunnel ingress IP address (src). The GUE security option provides origin authentication and integrity to GUE based tunnel; GUE payload encryption provides payload privacy over an IP delivery network or Internet. The two functions are processed separately at tunnel end points. A GUE tunnel can use both functions or use one of them.

When both encryption and security are required, an encapsulator must perform payload encryption first and then encapsulate the encrypted packet with security flag and payload transform flag set in GUE header; the security option field must be filled according Section 5.2 above and the payload transform field must be filled according to Section 5.3 above. The decapsulator must decapsulate the packet first, then perform the authentication validation. If the validation is successful, it performs the payload decryption according to the encryption information in the payload encryption field in the header. Else if the validation fails, the decapsulator must discard the packet and may generate an alert to the management system. These processing rules also apply when only one function, either encryption or security, is enabled, except the other function is not performed as stated above.

If GUE fragmentation is used in concert with the GUE security option and/or payload transform option, the security and playload transformation are performed after fragmentation at the encapsulator and before reassembly at the decapsulator.

In order to get flow entropy from the payload, an encapsulator must determine the flow entropy value (e.g. a hash over the 4-tuple of a TCP connection) before performing the payload encryption. The flow entropy value can then be set in UDP src port of the GUE packet.

DTLS [RFC6347] provides packet fragmentation capability. To avoid packets fragmentation being performed multiple times by an encapsulator, an encapsulator SHOULD only perform the packet fragmentation at part of the packet encapsulation process (e.g. using the GUE fragmentation option), not in payload encryption process (i.e. DTLS layer fragmentation should be avoided).

DTLS usage [RFC6347] is limited to a single DTLS session for any specific tunnel encapsulator/decapsulator pair (identified by source and destination IP addresses). Both IP addresses MUST be unicast addresses - multicast traffic is not supported when DTLS is used. A GUE tunnel decapsulator implementation that supports DTLS can establish DTLS session(s) with one or multiple tunnel encapsulators, and likewise a GUE tunnel encapsulator implementation can establish DTLS session(s) with one or multiple decapsulators.

5.5. Considerations of Using Other Security Tunnel Mechanisms

GUE may rely on other secure tunnel mechanisms such as DTLS [RFC6347] over the whole UDP payload for securing the whole GUE packet or IPsec [RFC4301] to achieve the secure transport over an IP network or Internet.

IPsec [RFC4301] was designed as a network security mechanism, and therefore it resides at the network layer. As such, if the tunnel is secured with IPsec, the UDP header would not be visible to intermediate routers anymore in either IPsec tunnel or transport mode. The big drawback here prohibits intermediate routers to perform load balancing based on the flow entropy in UDP header. In addition, this method prohibits any middle box function on the path.

By comparison, DTLS [RFC6347] was designed with application security and can better preserve network and transport layer protocol information than IPsec [RFC4301]. Using DTLS to secure the GUE tunnel, both GUE header and payload will be encrypted. In order to differentiate plaintext GUE header from encrypted GUE header, the destination port of the UDP header between two must be different, which essentially requires another standard UDP port for GUE with

DTLS. The drawback on this method is to prevent a middle box operation to GUE tunnel on the path.

Use of two independent tunnel mechanisms such as GUE and DTLS to carry a network protocol over an IP network adds some overlap and process complex. For example, fragmentation will be done twice.

As the result, a GUE tunnel SHOULD use the security mechanisms specified in this document to provide secure transport over an IP network or Internet when it is needed. GUE tunnels with the GUE security options can be used as a secure transport mechanism over an IP network and Internet.

6. Remote checksum offload option

Remote checksum offload is mechanism that provides checksum offload of encapsulated packets using rudimentary offload capabilities found in most Network Interface Card (NIC) devices. Many NIC implementations can only offload the outer UDP checksum in UDP encapsulation. Remote checksum offload is described in [UDPENCAP].

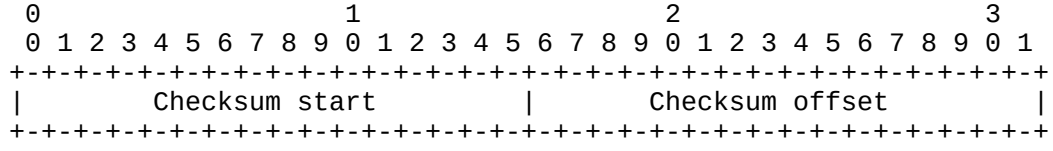
In remote checksum offload the outer header checksum, that is in the outer UDP header, is enabled in packets and, with some additional meta information, a receiver is able to deduce the checksum to be set for an inner encapsulated packet. Effectively this offloads the computation of the inner checksum. Enabling the outer checksum in encapsulation has the additional advantage that it covers more of the packet than the inner checksum including the encapsulation headers.

The remote offload checksum option should not be used when GUE fragmentation is also being performed. In this case the offload of the outer UDP checksum does not cover the whole transport segment so remote checksum offload would not properly set the inner transport layer checksum.

6.1. Option format

The presence of the GUE remote checksum offload option is indicated by the R bit in the GUE header.

The format of remote checksum offload field is:



The fields of the option are:

- o Checksum start: starting offset for checksum computation relative to the start of the encapsulated payload. This is typically the offset of a transport header (e.g. UDP or TCP).

- o Checksum offset: Offset relative to the start of the encapsulated packet where the derived checksum value is to be written. This typically is the offset of the checksum field in the transport header (e.g. UDP or TCP).

6.2. Transmitter operation

The typical actions to set remote checksum offload on transmit are:

1) Transport layer creates a packet and indicates in internal packet meta data that checksum is to be offloaded to the NIC (normal transport layer processing for checksum offload). The checksum field is populated with the bitwise not of the checksum of the pseudo header or zero as appropriate.

2) Encapsulation layer adds its headers to the packet including the offload meta data. The start offset and checksum offset are set accordingly.

3) Encapsulation layer arranges for checksum offload of the outer header checksum (e.g. UDP).

4) Packet is sent to the NIC. The NIC will perform transmit checksum offload and set the checksum field in the outer header. The inner header and rest of the packet are transmitted without modification.

6.3. Receiver operation

The typical actions a host receiver does to support remote checksum offload are:

1) Receive packet and validate outer checksum following normal processing (e.g. validate non-zero UDP checksum).

2) Validate the checksum option. If checksum start is greater than the length of the packet, then the packet must be dropped. If checksum offset is greater then the length of the packet minus two, then the packet must be dropped.

3) Deduce full checksum for the IP packet. If a NIC is capable of

receive checksum offload it will return either the full checksum of the received packet or an indication that the UDP checksum is correct. Either of these methods can be used to deduce the checksum over the IP packet [UDPENCAP].

4) From the packet checksum, subtract the checksum computed from the start of the packet (outer IP header) to the offset in the packet indicted by checksum start in the meta data. The result is the deduced checksum to set in the checksum field of the encapsulated transport packet.

   In pseudo code:

```
csum: initialized to checksum computed from start (outer IP
      header) to the end of the packet
start_of_packet: address of start of packet
encap_payload_offset: relative to start_of_packet
csum_start: value from meta data
checksum(start, len): function to compute checksum from start
      address for len bytes

csum -= checksum(start_of_packet, encap_payload_offset +
                 csum_start)
```

4) Write the resultant checksum value into the packet at the offset provided by checksum offset in the meta data.

   In pseudo code:

```
csum_offset: offset of checksum field

*(start_of_packet + encap_payload_offset +
  csum_offset) = csum
```

5) Checksum is verified at the transport layer using normal processing. This should not require any checksum computation over the packet since the complete checksum has already been provided.

6.4. Security Considerations

Remote checksum offload allows a means to change the GUE payload before being received at a decapsulator. In order to prevent misuse of this mechanism, a decapsulator should apply security checks on the GUE payload only after checksum remote offload has been processed.

7. Processing order of options

Options must be processed in a specific order for both sending and receive. Note that some options, such as the checksum option, depend on other fields in the GUE header to be set.

The order of processing options to send a GUE packet are:

1) Set VNID option.

2) Fragment if necessary and set fragmentation option. VNID is copied into each fragment. Note that if payload transformation will increase the size of the payload that must be accounted for when deciding how to fragment

3) Set Remote checksum offload.

4) Perform payload transform (potentially on each fragment) and set payload transform option.

5) Set security option.

6) Calculate GUE checksum and set checksum option.

On reception the order of actions is reversed.

1) Verify GUE checksum.

2) Verify security option.

3) Perform payload transformation (i.e. decrypt payload)

4) Adjust packet for remote checksum offload.

5) Perform reassembly.

6) Receive on virtual network indicated by VNID.

Note that the relative processing order of private fields is unspecified.

8. Security Considerations

Encapsulation of network protocol in GUE should not increase security risk, nor provide additional security in itself. GUE requires that the source port for UDP packets should be randomly seeded to mitigate some possible denial service attacks.

If the integrity and privacy of data packets being transported

through GUE is a concern, GUE security and payload encryption SHOULD be used to remove the concern. If the integrity is the only concern, the tunnel may consider use of GUE security only for optimization. Likewise, if the privacy is the only concern, the tunnel may use GUE encryption function only.

If GUE payload already provides secure mechanism, e.g., the payload is IPsec packets, it is still valuable to consider use of GUE security.

9. IANA Consideration

IANA is requested to assign flags for the extensions defined in this specification. Specifically, an assignment is requested for the V, SEC, K, F, T and R flags in the "GUE flag-fields" registry (proposed in [I.D.nvo3-gue]).

10. References

10.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[I.D.nvo3-gue] T. Herbert, L. Yong, and O. Zia, "Generic UDP Encapsulation" draft-ietf-nvo3-gue-03

10.2. Informative References

[RFC1071] Braden, R., Borman, D., and C. Partridge, "Computing the Internet checksum", RFC1071, September 1988.

[RFC1624] Rijsinghani, A., Ed., "Computation of the Internet Checksum via Incremental Update", RFC1624, May 1994.

[RFC1936] Touch, J. and B. Parham, "Implementing the Internet Checksum in Hardware", RFC1936, April 1996.

[RFC4459] MTU and Fragmentation Issues with In-the-Network Tunneling.

P. Savola. April 2006.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <http://www.rfc-editor.org/info/rfc4963>.

[RFC2764] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, A. Malis, "A Framework for IP Based Virtual Private Networks", RFC2764, February 2000.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security Considerations for IP Fragment Filtering", RFC 1858, October 1995.

[RFC3128] Miller, I., "Protection Against a Variant of the Tiny Fragment Attack (RFC 1858)", RFC 3128, June 2001.

[RFC3931] Lau, J., Townsley, W., et al, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC3931, 1999

[RFC5925] Touch, J., et al, "The TCP Authentication Option", RFC5925, June 2010.

[RFC6347] Rescoria, E., Modadugu, N., "Datagram Transport Layer Security Version 1.2", RFC6347, 2012.

[I.D.hy-nvo3-gue-4-nvo] Yong, L., Herbert, T., "Generic UDP Encapsulation (GUE) for Network Virtualization Overlay" draft-hy-nvo3-gue-4-nvo-03

[I.D.draft-mathis-frag-harmful] M. Mathis, J. Heffner, and B. Chandler, "Fragmentation Considered Very Harmful"

[I.D.templin-aerolink] F. Templin, "Transmission of IP Packets over AERO Links" draft-templin-aerolink-62.txt

[UDPENCAP] T. Herbert, "UDP Encapsulation in Linux", http://people.netfilter.org/pablo/netdev0.1/papers/UDP-Encapsulation-in-Linux.pdf

Authors' Addresses

Tom Herbert
Facebook
1 Hacker Way

Menlo Park, CA
USA

EMail: tom@herbertland.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75024
USA

Email: lucy.yong@huawei.com

Fred L. Templin
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Internet Area
Internet-Draft
Intended status: Informational
Expires: January 21, 2017

E. Baccelli
INRIA
C. Perkins
Futurewei

July 20, 2016

Multi-hop Ad Hoc Wireless Communication
draft-ietf-intarea-adhoc-wireless-com-02

Abstract

This document describes characteristics of communication between interfaces in a multi-hop ad hoc wireless network, that protocol engineers and system analysts should be aware of when designing solutions for ad hoc networks at the IP layer.

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 January 21, 2017.

Table of Contents

1. Introduction

Experience gathered with ad hoc routing protocol development, deployment and operation, shows that wireless communication presents specific challenges [RFC2501] [DoD01], which Internet protocol designers should be aware of, when designing solutions for ad hoc networks at the IP layer. This document does not prescribe solutions, but instead briefly describes these challenges in hopes of increasing that awareness.

As background, RFC 3819 [RFC3819] provides an excellent reference for higher-level considerations when designing protocols for shared media. From MTU to subnet design, from security to considerations about retransmissions, RFC 3819 provides guidance and design rationale to help with many aspects of higher-level protocol design.

The present document focuses more specifically on challenges in multi-hop ad hoc wireless networking. For example, in that context, even though a wireless link may experience high variability as a communications channel, such variation does not mean that the link is "broken". Many layer-2 technologies serve to reduce error rates by various means. Nevertheless, such errors as noted in this document may still become visible above layer-2 and so become relevant to the operation of higher layer protocols.

2. Multi-hop Ad Hoc Wireless Networks

For the purposes of this document, a multi-hop ad hoc wireless network will be considered to be a collection of devices that each have at least one radio transceiver (i.e., wireless network interface), and that are moreover configured to self-organize and provide store-and-forward functionality as needed to enable

communications. This document focuses on the characteristics of communications through such a network interface.

Although the characteristics of packet transmission over multi-hop ad hoc wireless networks, described below, are not the typical characteristics expected by IP [RFC6250], it is desirable and possible to run IP over such networks, as demonstrated in certain deployments currently in operation, such as Freifunk [FREIFUNK], and Funkfeuer [FUNKFEUER]. These deployments use routers running IP protocols e.g., OLSR (Optimized Link State Routing [RFC3626]) on top of IEEE 802.11 in ad hoc mode with the same ESSID (Extended Service Set Identification) at the link layer. Multi-hop ad hoc wireless networks may also run on link layers other than IEEE 802.11, and may use routing protocols other than OLSR. The following documents provide a number of examples: AODV [RFC3561], OLSRv2 [RFC7181], TBRPF [RFC3684], OSPF ([RFC5449], [RFC5820] and [RFC7137]), or DSR [RFC4728].

Note that in contrast, devices communicating via an IEEE 802.11 access point in infrastructure mode do not form a multi-hop ad hoc wireless network, since the central role of the access point is predetermined, and devices other than the access point do not generally provide store-and-forward functionality.

3. Common Packet Transmission Characteristics in Multi-hop Ad Hoc Wireless Networks

In the following, we will consider several devices in a multi-hop ad hoc wireless network N. Each device will be considered only through its own wireless interface to network N. For conciseness and readability, this document uses the expressions "device A" (or simply "A") as a synonym for "the wireless interface of device A to network N".

Let A and B be two devices in network N. Suppose that, when device A transmits an IP packet through its interface on network N, that packet is correctly and directly received by device B without requiring storage and/or forwarding by any other device. We will then say that B can "detect" A. Note that therefore, when B detects A, an IP packet transmitted by A will be rigorously identical to the corresponding IP packet received by B.

Let S be the set of devices that detect device A through its wireless interface on network N. The following section gathers common characteristics concerning packet transmission over such networks, which were observed through experience with MANET routing protocol development (for instance, OLSR[RFC3626], AODV[RFC3561], TBRPF[RFC3684], DSR[RFC4728], and OSPF-MPR[RFC5449]), as well as

deployment and operation (e.g., Freifunk[FREIFUNK], Funkfeuer[FUNKFEUER]).

3.1. Asymmetry, Time-Variation, and Non-Transitivity

First, even though a device C in set S can (by definition) detect device A, there is no guarantee that C can, conversely, send IP packets directly to A. In other words, even though C can detect A (since it is a member of set S), there is no guarantee that A can detect C. Thus, multi-hop ad hoc wireless communications may be "asymmetric". Such cases are common.

Second, there is no guarantee that, as a set, S is at all stable, i.e. the membership of set S may in fact change at any rate, at any time. Thus, multi-hop ad hoc wireless communications may be "time-variant". Time variation is often observed in multi-hop ad hoc wireless networks due to variability of the wireless medium, and to device mobility.

Now, conversely, let V be the set of devices which A detects. Suppose that A is communicating at time t0 through its interface on network N. As a consequence of time variation and asymmetry, we observe that A:

1. cannot assume that S = V, and

2. cannot assume that S and/or V are unchanged at time t1 later than t0.

Furthermore, transitivity is not guaranteed over multi-hop ad hoc wireless networks. Suppose that, through their respective interfaces within network N:

1. device B and device A can detect one another (i.e. B is a member of sets S and V), and,

2. device A and device C can also detect one another (i.e. C is a also a member of sets S and V).

These assumptions do not imply that B can detect C, nor that C can detect B (through their interface on network N). Such "non-transitivity" is common on multi-hop ad hoc wireless networks.

In summary: multi-hop ad hoc wireless communications can be asymmetric, non-transitive, and time-varying.

3.2. Radio Range and Wireless Irregularities

Section 3.1 presents an abstract description of some common characteristics concerning packet transmission over multi-hop ad hoc wireless networks. This section describes practical examples, which illustrate the characteristics listed in Section 3.1 as well as other common effects.

Wireless communications are particularly subject to limitations on the distance across which they may be established. The range-limitation factor creates specific problems on multi-hop ad hoc wireless networks. Due to the lack of isolation between the transmitters, the radio ranges of several devices often partially overlap, causing communication to be non-transitive and/or asymmetric as described in Section 3.1. Moreover, the range of each device may depend on location and environmental factors. This is in addition to possible time variations of range and signal strength.

For example it may happen that a device B detects a device A which transmits at high power, whereas B transmits at lower power. In such cases, as depicted in Figure 1, B can detect A, but A cannot detect B. This exemplifies asymmetry in wireless communications as defined in Section 3.1.

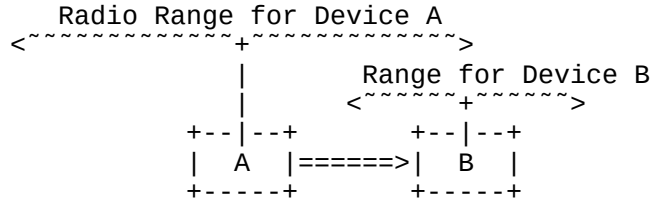


Figure 1: Asymmetric Wireless Communication

Another example, depicted in Figure 2, is known as the "Hidden Terminal" problem. Even though the devices all have equal power for their radio transmissions, they cannot all detect one another. In the figure, devices A and B can detect one another, and devices A and C can also detect one another. Nevertheless, B and C cannot detect one another. When B and C simultaneously try to communicate with A, their radio signals collide. Device A may then receive incoherent noise, and may even be unable to determine the source of the noise. The hidden terminal problem is a consequence of the property of non-transitivity in multi-hop ad hoc wireless communications as described in Section 3.1.

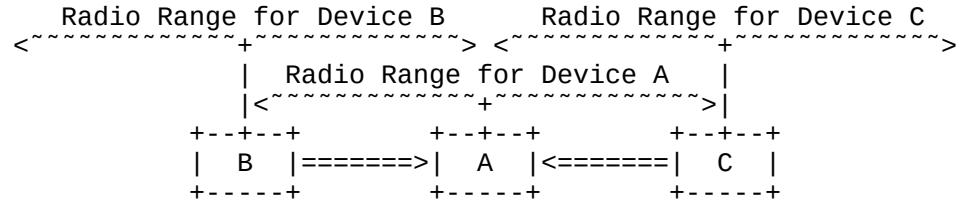


Figure 2: Hidden Terminal Problem

Another situation, shown in Figure 3, is known as the "Exposed Terminal" problem. In the figure, device A and device B can detect each other, and A is transmitting packets to B, thus A cannot detect device C -- but C can detect A. As shown in Figure 3, during the on-going transmission of A, device C cannot reliably communicate with device D because of interference within C's radio range due to A's transmissions. Device C is then said to be "exposed", because it is exposed to co-channel interference from A and is thereby prevented from reliably exchanging protocol messages with D -- even though these transmissions would not interfere with the reception of data sent from A destined to B.

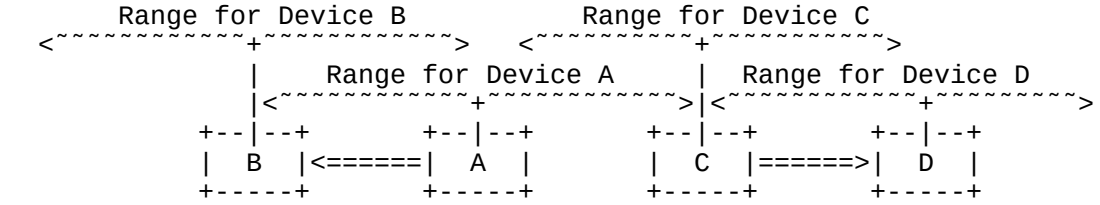


Figure 3: Exposed Terminal Problem

Hidden and exposed terminal situations are often observed in multi-hop ad hoc wireless networks. Asymmetry issues with wireless communication may also arise for reasons other than power inequality (e.g., multipath interference). Such problems are often resolved by specific mechanisms below the IP layer; CSMA/CA, for example, requires that the physical medium be unoccupied from the point of view of both devices before starting transmission. Nevertheless, depending on the link layer technology in use and the position of the devices, such problems may affect the IP layer due to range limitation and partial overlap.

Besides radio range limitations, wireless communications are affected by irregularities in the shape of the geographical area over which devices may effectively communicate (see for instance [MC03],

[MI03]). For example, even omnidirectional wireless transmission is typically non-isotropic (i.e. non-circular). Signal strength often suffers frequent and significant variations, which do not have a simple dependence on distance. Instead, the dependence is a complex function of the environment including obstacles, weather conditions, interference, and other factors that change over time. Because wireless communications often encounter different terrain, path, obstructions, atmospheric conditions and other phenomena, analytical formulation of signal strength is considered intractable [VTC99]. The radio engineering community has developed numerous radio propagation approximations, relying on median values observed in specific environments [SAR03].

These irregularities cause communications on multi-hop ad hoc wireless networks to be non-transitive, asymmetric, or time-varying, as described in Section 3.1, and may impact protocols at the IP layer and above. There may be no indication to the IP layer when a previously established communication channel becomes unusable; "link down" triggers are often absent in multi-hop ad hoc wireless networks, since the absence of detectable radio energy (e.g., in carrier waves) may simply indicate that neighboring devices are not currently transmitting.

4. Alternative Terminology

Many terms have been used in the past to describe the relationship of devices in a multi-hop ad hoc wireless network based on their ability to send or receive packets to/from each other. The terms used in previous sections of this document have been selected because the authors believe they are unambiguous, with respect to the goal of this document as formulated in Section 1.

In this section, we exhibit some other terms that describe the same relationship between devices in multi-hop ad hoc wireless networks. In the following, let network N be, again, a multi-hop ad hoc wireless network. Let the set S be, as before, the set of devices that can directly receive packets transmitted by device A through its interface on network N. In other words, any device B belonging to S can detect packets transmitted by A. Then, due to the asymmetric nature of wireless communications:

- We may say that device A "reaches" device B. In this terminology, there is no guarantee that B reaches A, even if A reaches B.

- We may say that device B "hears" device A. In this terminology, there is no guarantee that A hears B, even if B hears A.

- We may say that device A "has a link" to device B. In this terminology, there is no guarantee that B has a link to A, even if A has a link to B.

- We may say that device B "is adjacent to" device A. In this terminology, there is no guarantee that A is adjacent to B, even if B is adjacent to A.

- We may say that device B "is downstream from" device A. In this terminology, there is no guarantee that A is downstream from B, even if B is downstream from A.

- We may say that device B "is a neighbor of" device A. In this terminology, there is no guarantee that A is a neighbor of B, even if B a neighbor of A. Terminology based on "neighborhood" is quite confusing for multi-hop wireless communications. For example, when B can detect A, but A cannot detect B, it is not clear whether or not B should be considered a neighbor of A; A would not necessarily be aware that B was a neighbor, as it cannot detect B. It is thus best to avoid the "neighbor" terminology, except when bidirectionality has been established.

This list of alternative terminologies is given here for illustrative purposes only, and is not suggested to be complete or even representative of the breadth of terminologies that have been used in various ways to explain the properties mentioned in Section 3. Note that bidirectionality is not synonymous with symmetry. For example, the error statistics in either direction are often different for a link that is otherwise considered bidirectional.

5. Security Considerations

Section 18 of RFC 3819 [RFC3819] provides an excellent overview of security considerations at the subnetwork layer. Beyond the material there, multi-hop ad hoc wireless networking (i) is not limited to subnetwork layer operation, and (ii) makes use of wireless communications.

On one hand, a detailed description of security implications of wireless communications in general is outside of the scope of this document. It is true that eavesdropping on a wireless link is much easier than for wired media (although significant progress has been made in the field of wireless monitoring of wired transmissions). As a result, traffic analysis attacks can be even more subtle and difficult to defeat in this context. Furthermore, such communications over a shared media are particularly prone to theft of service and denial of service (DoS) attacks.

On the other hand, the potential multi-hop aspect of the networks we consider in this document goes beyond traditional scope of subnetwork design. In practice, unplanned relaying of network traffic (both user traffic and control traffic) happens routinely. Due to the physical nature of wireless media, Man in the Middle (MITM) attacks are facilitated, which may significantly alter network performance. This highlights the importance of the "end-to-end principle": L3 security, end-to-end, becomes a primary goal, independently of securing layer-2 and layer-1 protocols (though L2 and L1 security often help to reach this goal).

6. IANA Considerations

This document does not have any IANA actions.

7. Informative References

[RFC2501] Corson, S. and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, DOI 10.17487/RFC2501, January 1999, <http://www.rfc-editor.org/info/rfc2501>.

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <http://www.rfc-editor.org/info/rfc3561>.

[RFC3626] Clausen, T., Ed. and P. Jacquet, Ed., "Optimized Link State Routing Protocol (OLSR)", RFC 3626, DOI 10.17487/RFC3626, October 2003, <http://www.rfc-editor.org/info/rfc3626>.

[RFC3684] Ogier, R., Templin, F., and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", RFC 3684, DOI 10.17487/RFC3684, February 2004, <http://www.rfc-editor.org/info/rfc3684>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <http://www.rfc-editor.org/info/rfc3819>.

[RFC4728] Johnson, D., Hu, Y., and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4", RFC 4728, DOI 10.17487/RFC4728, February 2007, <http://www.rfc-editor.org/info/rfc4728>.

[RFC5449] Baccelli, E., Jacquet, P., Nguyen, D., and T. Clausen, "OSPF Multipoint Relay (MPR) Extension for Ad Hoc Networks", RFC 5449, DOI 10.17487/RFC5449, February 2009, <http://www.rfc-editor.org/info/rfc5449>.

[RFC5820] Roy, A., Ed. and M. Chandra, Ed., "Extensions to OSPF to Support Mobile Ad Hoc Networking", RFC 5820, DOI 10.17487/RFC5820, March 2010, <http://www.rfc-editor.org/info/rfc5820>.

[RFC6250] Thaler, D., "Evolution of the IP Model", RFC 6250, DOI 10.17487/RFC6250, May 2011, <http://www.rfc-editor.org/info/rfc6250>.

[RFC7137] Retana, A. and S. Ratliff, "Use of the OSPF-MANET Interface in Single-Hop Broadcast Networks", RFC 7137, DOI 10.17487/RFC7137, February 2014, <http://www.rfc-editor.org/info/rfc7137>.

[RFC7181] Clausen, T., Dearlove, C., Jacquet, P., and U. Herberg, "The Optimized Link State Routing Protocol Version 2", RFC 7181, DOI 10.17487/RFC7181, April 2014, <http://www.rfc-editor.org/info/rfc7181>.

[DoD01] Freebersyser, J. and B. Leiner, "A DoD perspective on mobile ad hoc networks", Addison Wesley C. E. Perkins, Ed., 2001, pp. 29--51, 2001.

[FUNKFEUER] "Austria Wireless Community Network, http://www.funkfeuer.at", 2013.

[MC03] Corson, S. and J. Macker, "Mobile Ad hoc Networking: Routing Technology for Dynamic, Wireless Networks", IEEE Press Mobile Ad hoc Networking, Chapter 9, 2003.

[SAR03] Sarkar, T., Ji, Z., Kim, K., Medour, A., and M. Salazar-Palma, "A Survey of Various Propagation Models for Mobile Communication", IEEE Press Antennas and Propagation Magazine, Vol. 45, No. 3, 2003.

[VTC99] Kim, D., Chang, Y., and J. Lee, "Pilot power control and service coverage support in CDMA mobile systems", IEEE Press Proceedings of the IEEE Vehicular Technology Conference (VTC), pp.1464-1468, 1999.

[MI03] Kotz, D., Newport, C., and C. Elliott, "The Mistaken Axioms of Wireless-Network Research", Dartmouth College Computer Science Technical Report TR2003-467, 2003.

[FREIFUNK] "Freifunk Wireless Community Networks, http://www.freifunk.net", 2013.

Appendix A. Acknowledgements

This document stems from discussions with the following people, in alphabetical order: Jari Arkko, Teco Boot, Brian Carpenter, Carlos Jesus Bernardos Cano, Zhen Cao, Ian Chakeres, Thomas Clausen, Robert Cragie, Christopher Dearlove, Ralph Droms, Brian Haberman, Ulrich Herberg, Paul Lambert, Kenichi Mase, Thomas Narten, Erik Nordmark, Alexandru Petrescu, Stan Ratliff, Zach Shelby, Shubhranshu Singh, Fred Templin, Dave Thaler, Mark Townsley, Ronald Velt in't, and Seung Yi.

Authors' Addresses

Emmanuel Baccelli
INRIA

EMail: Emmanuel.Baccelli@inria.fr
URI: http://www.emmanuelbaccelli.org/

Charles E. Perkins
Futurewei

Phone: +1-408-330-4586
EMail: charlie.perkins@huawei.com

Network Virtualization Overlays (nvo3)
Internet-Draft
Intended status: Standard track
Expires January 7, 2017

T. Herbert
Facebook
L. Yong
Huawei USA
O. Zia
Microsoft

July 6, 2016

Generic UDP Encapsulation
draft-ietf-nvo3-gue-04

Status of this Memo

Copyright Notice

Abstract

This specification describes Generic UDP Encapsulation (GUE), which is a scheme for using UDP to encapsulate packets of arbitrary IP protocols for transport across layer 3 networks. By encapsulating packets in UDP, specialized capabilities in networking hardware for efficient handling of UDP packets can be leveraged. GUE specifies basic encapsulation methods upon which higher level constructs, such tunnels and overlay networks for network virtualization, can be constructed. GUE is extensible by allowing optional data fields as part of the encapsulation, and is generic in that it can encapsulate packets of various IP protocols.

Table of Contents

1. Introduction

This specification describes Generic UDP Encapsulation (GUE) which is a general method for encapsulating packets of arbitrary IP protocols within User Datagram Protocol (UDP) [RFC0768] packets. Encapsulating packets in UDP facilitates efficient transport across networks. Networking devices widely provide protocol specific processing and optimizations for UDP (as well as TCP) packets. Packets for atypical IP protocols (those not usually parsed by networking hardware) can be encapsulated in UDP packets to maximize deliverability and to leverage flow specific mechanisms for routing and packet steering.

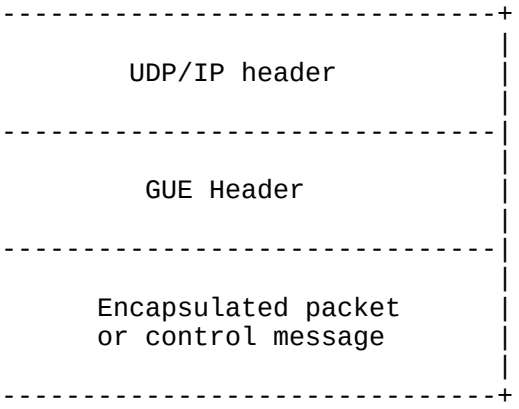
GUE provides an extensible header format for including optional data in the encapsulation header. This data potentially covers items such as virtual networking identifier, security data for validating or authenticating the GUE header, congestion control data, etc. GUE also allows private optional data in the encapsulation header. This feature can be used by a site or implementation to define local custom optional data, and allows experimentation of options that may eventually become standard.

This document does not define any specific GUE extensions. [GUEEXTENS] specifies a set of core extensions and [GUE4NVO3] defines an extension for using GUE with network virtualization.

The motivation for the GUE protocol is described in section 6.

2. Base packet format

A GUE packet is comprised of a UDP packet whose payload is a GUE header followed by a payload which is either an encapsulated packet of some IP protocol or a control message (like an OAM message). A GUE packet has the general format:



The GUE header is variable length as determined by the presence of optional fields.

2.1. GUE version

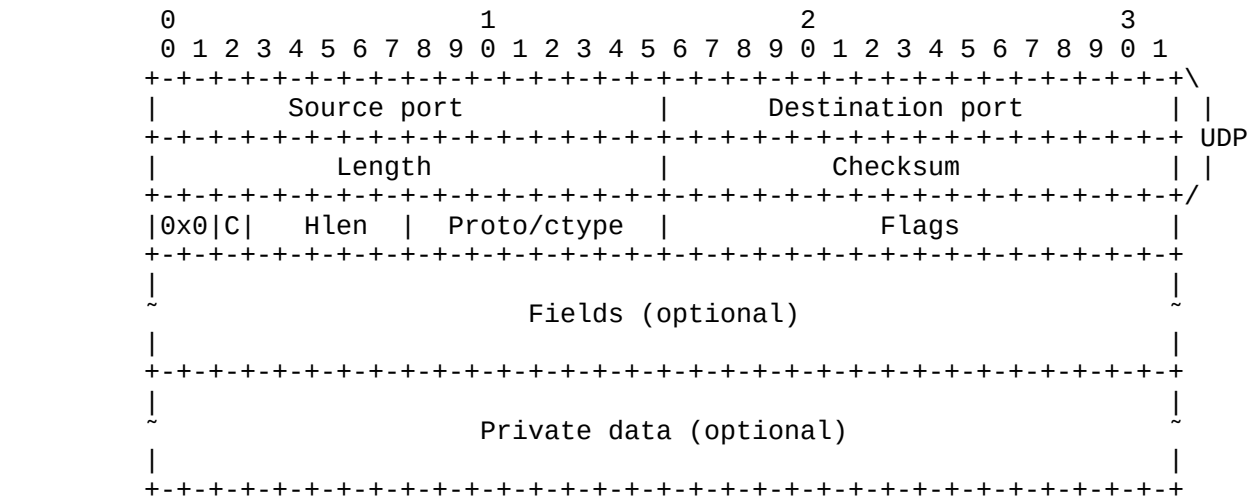
The first two bits of the GUE header contain the GUE protocol version number. The rest of the fields after the GUE version number are defined based on the version number. Versions 0x0 and 0x1 are described in this specification; versions 0x2 and 0x3 are reserved,

3. Version 0

Version 0 of GUE defines a gereric extensible format to encapsulate packets by Internet protocol number.

3.1. Header format

The header format for version 0x0 of GUE in UDP is:



The contents of the UDP header are:

- Source port: This should be set to a flow entropy value for use with ECMP. The properties of flow entropy are described in section 5.11.
- Destination port: The GUE assigned port number, 6080.
- Length: Canonical length of the UDP packet (length of UDP header and payload).
- Checksum: Standard UDP checksum (see section 5.7).

The GUE header consists of:

- Ver: GUE protocol version (0x0).
- C: Control flag. When set indicates a control message, not set indicates a data message.
- Hlen: Length in 32-bit words of the GUE header, including optional fields but not the first four bytes of the header. Computed as (header_len - 4) / 4. All GUE headers are a multiple of four bytes in length. Maximum header length is 128 bytes.
- Proto/ctype: When the C bit is set this field contains a control

message type for the payload (section 3.2.2). When C bit is not set, the field holds the Internet protocol number for the encapsulated packet in the payload (section 3.2.1). The control message or encapsulated packet begins at the offset provided by Hlen.

- o Flags. Header flags that may be allocated for various purposes and may indicate presence of optional fields. Undefined header flag bits MUST be set to zero on transmission.

- o Fields: Optional fields whose presence is indicated by corresponding flags.

- o Private data: Optional private data (see section 3.4). If private data is present it immediately follows that last field present in the header. The length of this data is determined by subtracting the starting offset from the header length.

3.2. Proto/ctype field

When the C bit is not set, the proto/ctype field must be set to a valid Internet protocol number. The protocol number serves as an indication of the type of next protocol header which is contained in the GUE payload at the offset indicated in Hlen. Intermediate devices may parse the GUE payload per the number in the proto/ctype field, and header flags cannot affect the interpretation of the proto/ctype field.

3.2.1 Proto field

When the C bit is not set the proto/ctype field contains an IANA Internet Protocol Number. The protocol number in interpreted relative to the IP protocol that encapsulates the UDP packet (i.e. protocol of the outer IP header).

For an IPv4 header the protocol may be set to any number except for those that refer to IPv6 extension headers or ICMPv6 options (number 58). An exception is that the destination options extension header using the PadN option may be used with IPv4 as described in section 3.6. The "no next header" protocol number (59) may be used with IPv4 as described below.

For an IPv6 header the protocol may be set to any defined protocol number except Hop-by-hop options (number 0). If a received GUE packet in IPv6 contains a protocol number that is an extension header (e.g. Destination Options) then the extension header is processed after the GUE header as though the GUE header itself were an extension header.

IP protocol number 59 ("No next header") may be set to indicate that the GUE payload does not begin with the header of an IP protocol. This would be the case, for instance, if the GUE payload were a fragment when performing GUE level fragmentation. The interpretation of the payload is performed though other means (such as flags and optional fields), and intermediate devices must not parse packets the packet based on the IP protocol number in this case.

3.2.2 Ctype field

When the C bit is set, the proto/ctype field must be set to a valid control message type. A value of zero indicates that the GUE payload requires further interpretation to deduce the control type. This might be the case when the payload is a fragment of a control message, where only the reassembled packet can be interpreted as a control message.

Control message types 1 through 127 may be defined in standards. Types 128 through 255 are reserved to be user defined for experimentation or private control messages.

This document does not specify any standard control message types, other than type 0, for GUE.

3.3. Flags and optional fields

Flags and associated optional fields are the primary mechanism of extensibility in GUE. There are sixteen flag bits in the GUE header.

A flag may indicate presence of optional fields. The size of an optional field indicated by a flag must be fixed.

Flags may be paired together to allow different lengths for an optional field. For example, if two flag bits are paired, a field may possibly be three different lengths. Regardless of how flag bits may be paired, the lengths and offsets of optional fields corresponding to a set of flags must be well defined.

Optional fields are placed in order of the flags. New flags are to be allocated from high to low order bit contiguously without holes. Flags allow random access, for instance to inspect the field corresponding to the Nth flag bit, an implementation only considers the previous N-1 flags to determine the offset. Flags after the Nth flag are not pertinent in calculating the offset of the Nth flag.

Flags (or paired flags) are idempotent such that new flags must not cause reinterpretation of old flags. Also, new flags should not alter interpretation of other elements in the GUE header nor how the

message is parsed (for instance, in a data message the proto/ctype field always holds an IP protocol number as an invariant).

The set of available flags may be extended in the future by defining a "flag extensions bit" that refers to a field containing a new set of flags.

3.4. Private data

An implementation may use private data for its own use. The private data immediately follows the last field in the GUE header and is not a fixed length. This data is considered part of the GUE header and must be accounted for in header length (Hlen). The length of the private data must be a multiple of four and is determined by subtracting the offset of private data in the GUE header from the header length. Specifically:

```
Private_length = (Hlen * 4) - Length(flags)
```

Where "Length(flags)" returns the sum of lengths of all the optional fields present in the GUE header. When there is no private data present, length of the private data is zero.

The semantics and interpretation of private data are implementation specific. The private data may be structured as necessary, for instance it might contain its own set of flags and optional fields.

An encapsulator and decapsulator MUST agree on the meaning of private data before using it. The mechanism to achieve this agreement is outside the scope of this document but could include implementation-defined behavior, coordinated configuration, in-band communication using GUE control messages, and out-of-band messages.

If a decapsulator receives a GUE packet with private data, it MUST validate the private data appropriately. If a decapsulator does not expect private data from an encapsulator the packet MUST be dropped. If a decapsulator cannot validate the contents of private data per the provided semantics the packet MUST also be dropped. An implementation may place security data in GUE private data which must be verified for packet acceptance.

3.5. Message types

3.5.1. Control messages

Control messages are indicated in the GUE header when the C bit is set. The payload is interpreted as a control message with type specified in the proto/ctype field. The format and contents of the

control message are indicated by the type and can be variable length.

Other than interpreting the proto/ctype field as a control message type, the meaning and semantics of the rest of the elements in the GUE header are the same as that of data messages. Forwarding and routing of control messages should be the same as that of a data message with the same outer IP and UDP header and GUE flags-- this ensures that control messages can be created that follow the same path as data messages.

Control messages can be defined for OAM type messages. For instance, an echo request and corresponding echo reply message may be defined to test for liveness.

3.5.2. Data messages

Data messages are indicated in GUE header with C bit not set. The payload of a data message is interpreted as an encapsulated packet of an Internet protocol indicated in the proto/ctype field. The packet immediately follows the GUE header.

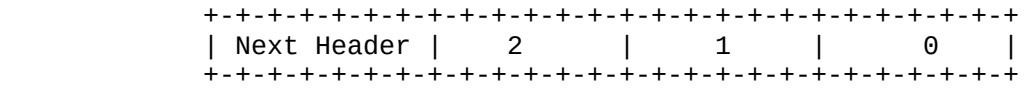
Data messages are a primary means of encapsulation and can be used to create tunnels for overlay networks.

3.6. Hiding the transport layer protocol number

The GUE header indicates the Internet protocol of the encapsulated packet. This is either contained in the Proto/ctype field of the primary GUE header, or is contained in the Payload Type field of a GUE Transform Field (used to encrypt the payload with DTLS). If the protocol number must be obfuscated, that is the transport protocol in use must be hidden from the network, then a trivial destination options can be used at the beginning of the payload.

The PadN destination option can be used to encode the transport protocol as a next header of an extension header (and maintain alignment of encapsulated transport headers). The Proto/ctype field or Payload Type field of the GUE Transform field is set to 60 to indicate that the first encapsulated header is a Destination Options extension header.

The format of the extension header is below:



For IPv4, it is permitted in GUE to used this precise destination

option to contain the obfuscated protocol number. In this case next header must refer to a valid IP protocol for IPv4. No other extension headers or destination options are permitted with IPv4.

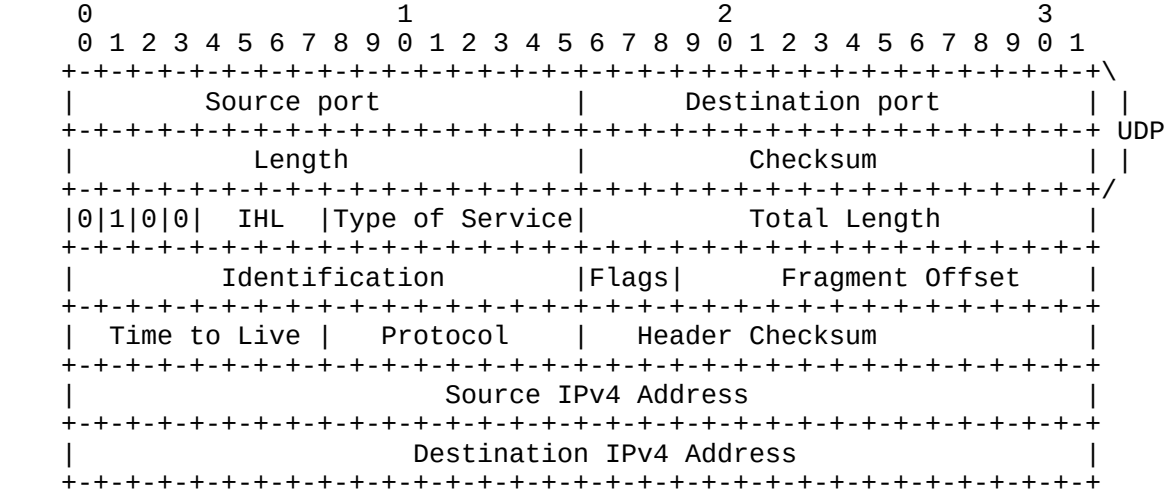
4. Version 1

Version 1 of GUE allows direct encapsulation of IPv4 and IPv6 in UDP. In this version there is no GUE header; a UDP packet carries an IP packet. The first two bits of the UDP payload for GUE are the GUE version and coincide with the first two bits of the version number in the IP header. The first two version bits of IPv4 and IPv6 are 01, so we use GUE version 1 for direct IP encapsulation which makes two bits of GUE version to also be 01.

This technique is effectively a means to compress out the GUE header when encapsulating IPv4 or IPv6 packets and there are no flags or optional fields present. This method is compatible to use on the same port number as packets with the the GUE header (GUE version 0 packets). This technique saves encapsulation overhead on costly links for the common use of IP encapsulation, and also obviates the need to allocate a separate port number for IP-over-UDP encapsulation.

4.1. Direct encapsulation of IPv4

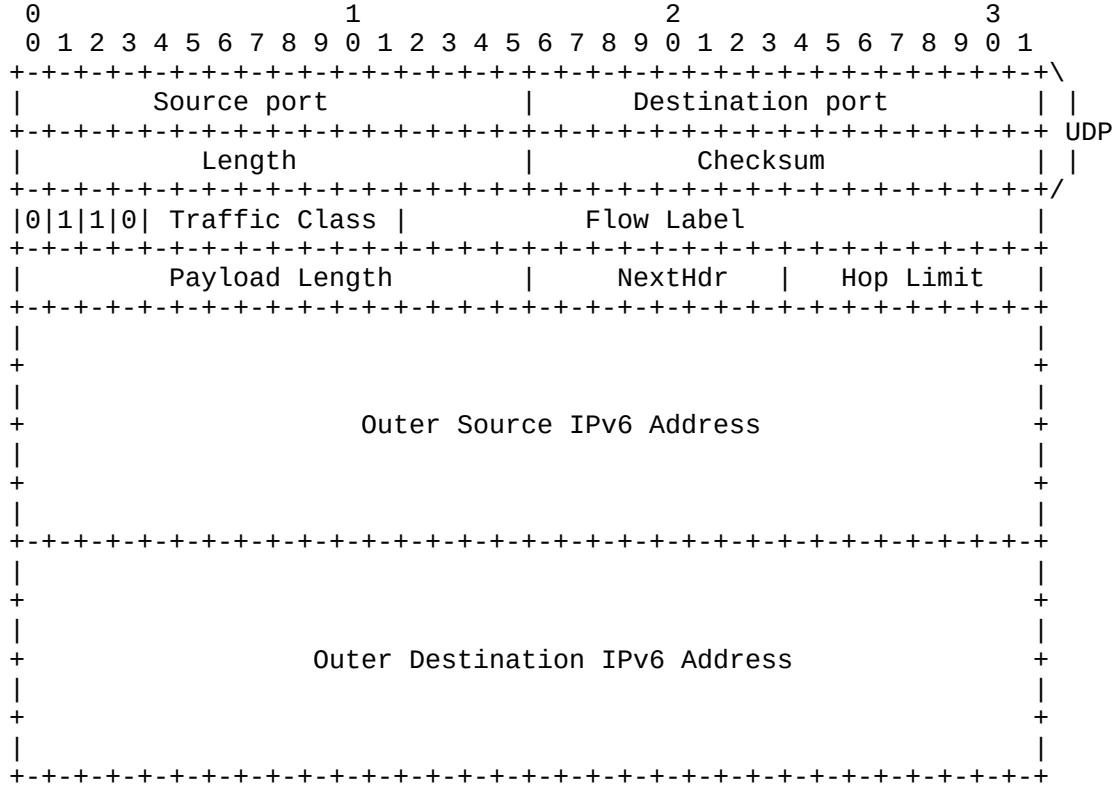
The format for encapsulating IPv4 directly in UDP is demonstrated below.



Note that 0100 value IP version field express the GUE version as 1 (bits 01) and IP version as 4 (bits 0100).

4.2. Direct encapsulation of IPv6

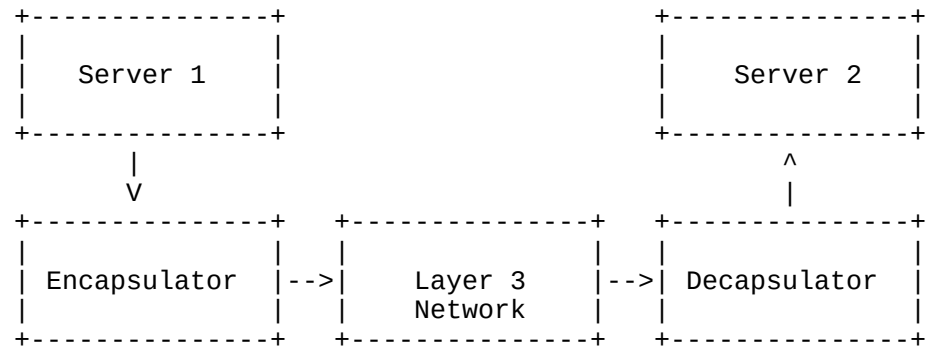
The format for encapsulating IPv4 directly in UDP is demonstrated below.



Note that 0110 value IP version field express the GUE version as 1 (bits 01) and IP version as 6 (bits 0110).

5. Operation

The figure below illustrates the use of GUE encapsulation between two servers. Sever 1 is sending packets to server 2. An encapsulator performs encapsulation of packets from server 1. These encapsulated packets traverse the network as UDP packets. At the decapsulator, packets are decapsulated and sent on to server 2. Packet flow in the reverse direction need not be symmetric; GUE encapsulation is not required in the reverse path.



The encapsulator and decapsulator may be co-resident with the corresponding servers, or may be on separate nodes in the network.

5.1. Network tunnel encapsulation

Network tunneling can be achieved by encapsulating layer 2 or layer 3 packets. In this case the encapsulator and decapsulator nodes are the tunnel endpoints. These could be routers that provide network tunnels on behalf of communicating servers.

5.2. Transport layer encapsulation

When encapsulating layer 4 packets, the encapsulator and decapsulator should be co-resident with the servers. In this case, the encapsulation headers are inserted between the IP header and the transport packet. The addresses in the IP header refer to both the endpoints of the encapsulation and the endpoints for terminating the the transport protocol.

5.3. Encapsulator operation

Encapsulators create GUE data messages, set the fields of the UDP header, set flags and optional fields in the GUE header, and forward packets to a decapsulator.

An encapsulator may be an end host originating the packets of a flow, or may be a network device performing encapsulation on behalf of hosts (routers implementing tunnels for instance). In either case, the intended target (decapsulator) is indicated by the outer destination IP address.

If an encapsulator is tunneling packets, that is encapsulating packets of layer 2 or layer 3 protocols (e.g. EtherIP, IPIP, ESP tunnel mode), it should follow standard conventions for tunneling of one IP protocol over another. Diffserv interaction with tunnels is

described in [RFC2983], ECN propagation for tunnels is described in [RFC6040].

5.4. Decapsulator operation

A decapsulator performs decapsulation of GUE packets. A decapsulator is addressed by the outer destination IP address of a GUE packet. The decapsulator validates packets, including fields of the GUE header. If a packet is acceptable, the UDP and GUE headers are removed and the packet is resubmitted for protocol processing or control message processing if it is a control message.

If a decapsulator receives a GUE packet with an unsupported version, unknown flag, bad header length (too small for included optional fields), unknown control message type, bad protocol number, an unsupported payload type, or an otherwise malformed header, it MUST drop the packet. Such events may be logged subject to configuration and rate limiting of logging messages. No error message is returned back to the encapsulator. Note that set flags in GUE that are unknown to a decapsulator MUST NOT be ignored. If a GUE packet is received by a decapsulator with unknown flags, the packet MUST be dropped.

5.5. Router and switch operation

Routers and switches should forward GUE packets as standard UDP/IP packets. The outer five-tuple should contain sufficient information to perform flow classification corresponding to the flow of the inner packet. A switch should not normally need to parse a GUE header, and none of the flags or optional fields in the GUE header should affect routing.

A router should not modify a GUE header when forwarding a packet. It may encapsulate a GUE packet in another GUE packet, for instance to implement a network tunnel (i.e. by encapsulating an IP packet with a GUE payload in another IP packet as a GUE payload). In this case the router takes the role of an encapsulator, and the corresponding decapsulator is the logical endpoint of the tunnel.

5.6. Middlebox interactions

A middle box may interpret some flags and optional fields of the GUE header for classification purposes, but is not required to understand any of the flags or fields in GUE packets. A middle box must not drop a GUE packet because there are flags unknown to it. The header length in the GUE header allows a middlebox to inspect the payload packet without needing to parse the flags or optional fields.

5.6.1. Inferring connection semantics

A middlebox may infer bidirectional connection semantics for a UDP flow. For instance a stateful firewall may create a five-tuple rule to match flows on egress, and a corresponding five-tuple rule for matching ingress packets where the roles of source and destination are reversed for the IP addresses and UDP port numbers. To operate in this environment, a GUE tunnel must assume connected semantics defined by the UDP five tuple and the use of GUE encapsulation must be symmetric between both endpoints. The source port set in the UDP header must be the destination port the peer would set for replies. In this case the UDP source port for a tunnel would be a fixed value and not set to be flow entropy as described in section 5.11.

The selection of whether to make the UDP source port fixed or set to a flow entropy value for each packet sent should be configurable for a tunnel.

5.6.2. NAT

IP address and port translation can be performed on the UDP/IP headers adhering to the requirements for NAT with UDP [RFC478]. In the case of stateful NAT, connection semantics must be applied to a GUE tunnel as described in section 5.6.1.

5.7. Checksum Handling

This section describes the requirements around the UDP checksum. Checksums are an important consideration in that that they can provide end to end validation and protect against packet mis-delivery. The latter is allowed by the inclusion of a pseudo header that covers the IP addresses and UDP ports of the encapsulating headers.

5.7.1. Checksum requirements

The potential for mis-delivery of packets due to corruption of IP, UDP, or GUE headers must be considered. One of the following requirements must be met:

- UDP checksums are enabled (for IPv4 or IPv6).
- The GUE header checksum is used (defined in [GUEEXTENS]).
- Zero UDP checksums are used in accordance with applicable requirements in [GREUDP], [RFC6935], and [RFC6936].

5.7.2. UDP Checksum with IPv4

For UDP in IPv4, the UDP checksum MUST be processed as specified in

[RFC768] and [RFC1122] for both transmit and receive. An encapsulator MAY set the UDP checksum to zero for performance or implementation considerations. The IPv4 header includes a checksum that protects against mis-delivery of the packet due to corruption of IP addresses. The UDP checksum potentially provides protection against corruption of the UDP header, GUE header, and GUE payload. Enabling or disabling the use of checksums is a deployment consideration that should take into account the risk and effects of packet corruption, and whether the packets in the network are already adequately protected by other, possibly stronger mechanisms such as the Ethernet CRC. If an encapsulator sets a zero UDP checksum for IPv4 it SHOULD use the GUE header checksum as described in [GUEEXTENS].

When a decapsulator receives a packet, the UDP checksum field MUST be processed. If the UDP checksum is non-zero, the decapsulator MUST verify the checksum before accepting the packet. By default a decapsulator SHOULD accept UDP packets with a zero checksum. A node MAY be configured to disallow zero checksums per [RFC1122]; this may be done selectively, for instance disallowing zero checksums from certain hosts that are known to be sending over paths subject to packet corruption. If verification of a non-zero checksum fails, a decapsulator lacks the capability to verify a non-zero checksum, or a packet with a zero-checksum was received and the decapsulator is configured to disallow, the packet MUST be dropped.

5.7.3. UDP Checksum with IPv6

For UDP in IPv6, the UDP checksum MUST be processed as specified in [RFC768] and [RFC2460] for both transmit and receive. Unlike IPv4, there is no header checksum in IPv6 that protects against mis-delivery due to address corruption. Therefore, when GUE is used over IPv6, either the UDP checksum must be enabled or the GUE header checksum must be used. An encapsulator MAY set a zero UDP checksum for performance or implementation reasons, in which case the GUE header checksum MUST be used or applicable requirements for using zero UDP checksums in [GREUDP] MUST be met. If the UDP checksum is enabled, then the GUE header checksum should not be used since it is mostly redundant.

When a decapsulator receives a packet, the UDP checksum field MUST be processed. If the UDP checksum is non-zero, the decapsulator MUST verify the checksum before accepting the packet. By default a decapsulator MUST only accept UDP packets with a zero checksum if the GUE header checksum is used and is verified. If verification of a non-zero checksum fails, a decapsulator lacks the capability to verify a non-zero checksum, or a packet with a zero-checksum and no GUE header checksum was received, the packet MUST be dropped.

5.8. MTU and fragmentation

Standard conventions for handling of MTU (Maximum Transmission Unit) and fragmentation in conjunction with networking tunnels (encapsulation of layer 2 or layer 3 packets) should be followed. Details are described in MTU and Fragmentation Issues with In-the-Network Tunneling [RFC4459]

If a packet is fragmented before encapsulation in GUE, all the related fragments must be encapsulated using the same UDP source port. An operator may set MTU to account for encapsulation overhead and reduce the likelihood of fragmentation.

Alternative to IP fragmentation, the GUE fragmentation extension can be used. GUE fragmentation is described in [GUEEXTENS].

5.9. Congestion control

Per requirements of [RFC5405], if the IP traffic encapsulated with GUE implements proper congestion control no additional mechanisms should be required.

In the case that the encapsulated traffic does not implement any or sufficient control, or it is not known rather a transmitter will consistently implement proper congestion control, then congestion control at the encapsulation layer must be provided. Note this case applies to a significant use case in network virtualization in which guests run third party networking stacks that cannot be implicitly trusted to implement conformant congestion control.

Out of band mechanisms such as rate limiting, Managed Circuit Breaker [CIRCBRK], or traffic isolation may used to provide rudimentary congestion control. For finer grained congestion control that allows alternate congestion control algorithms, reaction time within an RTT, and interaction with ECN, in-band mechanisms may warranted.

DCCP [RFC4340] may be used to provide congestion control for encapsulated flows. In this case, the protocol stack for an IP tunnel may be IP-GUE-DCCP-IP. Alternatively, GUE can be extended to include congestion control (related data carried in GUE optional fields). Congestion control mechanisms for GUE will be elaborated in other specifications.

5.10. Multicast

GUE packets may be multicast to decapsulators using a multicast destination address in the encapsulating IP headers. Each receiving

host will decapsulate the packet independently following normal decapsulator operations. The receiving decapsulators should agree on the same set of GUE parameters and properties; how such an agreement is reached is outside the scope of this document.

GUE allows encapsulation of unicast, broadcast, or multicast traffic. Flow entropy (the value in the UDP source port) may be generated from the header of encapsulated unicast or broadcast/multicast packets at an encapsulator. The mapping mechanism between the encapsulated multicast traffic and the multicast capability in the IP network is transparent and independent to the encapsulation and is otherwise outside the scope of this document.

5.11. Flow entropy for ECMP

5.11.1. Flow classification

A major objective of using GUE is that a network device can perform flow classification corresponding to the flow of the inner encapsulated packet based on the contents in the outer headers.

Hardware devices commonly perform hash computations on packet headers to classify packets into flows or flow buckets. Flow classification is done to support load balancing (statistical multiplexing) of flows across a set of networking resources. Examples of such load balancing techniques are Equal Cost Multipath routing (ECMP), port selection in Link Aggregation, and NIC device Receive Side Scaling (RSS). Hashes are usually either a three-tuple hash of IP protocol, source address, and destination address; or a five-tuple hash consisting of IP protocol, source address, destination address, source port, and destination port. Typically, networking hardware will compute five-tuple hashes for TCP and UDP, but only three-tuple hashes for other IP protocols. Since the five-tuple hash provides more granularity, load balancing can be finer grained with better distribution. When a packet is encapsulated with GUE, the source port in the outer UDP packet is set to a flow entropy value that corresponds the flow of the inner packet. When a device computes a five-tuple hash on the outer UDP/IP header of a GUE packet, the resultant value classifies the packet per its inner flow.

Examples of deriving an flow entropy for encapsulation are:

- If the encapsulated packet is a layer 4 packet, TCP/IPv4 for instance, the flow entropy could be based on the canonical five-tuple hash of the inner packet.

- If the encapsulated packet is an AH transport mode packet with TCP as next header, the flow entropy could be a hash over a three-tuple: TCP protocol and TCP ports of the encapsulated packet.

- If a node is encrypting a packet using ESP tunnel mode and GUE encapsulation, the flow entropy could be based on the contents of clear-text packet. For instance, a canonical five-tuple hash for a TCP/IP packet could be used.

5.11.2. Flow entropy properties

The flow entropy is the value set in the UDP source port of a GUE packet. Flow entropy in the UDP source port should adhere to the following properties:

- The value set in the source port should be within the ephemeral port range. IANA suggests this range to be 49152 to 65535, where the high order two bits of the port are set to one. This provides fourteen bits of entropy for the value.

- The flow entropy should have a uniform distribution across encapsulated flows.

- An encapsulator may occasionally change the flow entropy used for an inner flow per its discretion (for security, route selection, etc). To avoid thrashing or flapping the value, the flow entropy used for a flow should not change more than once every thirty seconds (or a configurable value).

- Decapsulators, or any networking devices, should not attempt to interpret flow entropy as anything more than an opaque value. Neither should they attempt to reproduce the hash calculation used by an encapasulator in creating a flow entropy value. They may use the value to match further receive packets for steering decisions, but cannot assume that the hash uniquely or permanently identifies a flow.

- Input to the flow entropy calculation is not restricted to ports and addresses; input could include flow label from an IPv6 packet, SPI from an ESP packet, or other flow related state in the encapsulator that is not necessarily conveyed in the packet.

- The assignment function for flow entropy should be randomly seeded to mitigate denial of service attacks. The seed may be changed periodically.

6. Motivation for GUE

This section presents the motivation for GUE with respect to other encapsulation methods.

A number of different encapsulation techniques have been proposed for the encapsulation of one protocol over another. EtherIP [RFC3378] provides layer 2 tunneling of Ethernet frames over IP. GRE [RFC2784], MPLS [RFC4023], and L2TP [RFC2661] provide methods for tunneling layer 2 and layer 3 packets over IP. NVGRE [RFC7637] and VXLAN [RFC7348] are proposals for encapsulation of layer 2 packets for network virtualization. IPIP [RFC2003] and Generic packet tunneling in IPv6 [RFC2473] provide methods for tunneling IP packets over IP.

Several proposals exist for encapsulating packets over UDP including ESP over UDP [RFC3948], TCP directly over UDP [TCPUDP], VXLAN [RFC7348], LISP [RFC6830] which encapsulates layer 3 packets, MPLS/UDP [7510], and Generic UDP Encapsulation for IP Tunneling (GRE over UDP)[GREUDP]. Generic UDP tunneling [GUT] is a proposal similar to GUE in that it aims to tunnel packets of IP protocols over UDP.

GUE has the following discriminating features:

- UDP encapsulation leverages specialized network device processing for efficient transport. The semantics for using the UDP source port for flow entropy as input to ECMP are defined in section 5.11.

- GUE permits encapsulation of arbitrary IP protocols, which includes layer 2 3, and 4 protocols.

- Multiple protocols can be multiplexed over a single UDP port number. This is in contrast to techniques to encapsulate protocols over UDP using a protocol specific port number (such as ESP/UDP, GRE/UDP, SCTP/UDP). GUE provides a uniform and extensible mechanism for encapsulating all IP protocols in UDP with minimal overhead (four bytes of additional header).

- GUE is extensible. New flags and optional fields can be defined.

- The GUE header includes a header length field. This allows a network node to inspect an encapsulated packet without needing to parse the full encapsulation header.

- Private data in the encapsulation header allows local customization and experimentation while being compatible with processing in network nodes (routers and middleboxes).

- GUE includes both data messages (encapsulation of packets) and control messages (such as OAM).

7. Security Considerations

Security for GUE can be provided by lower layers or through GUE security extensions.

IPsec in transport mode may be used to authenticate or encrypt GUE packets (GUE header and payload). Existing network security mechanisms, such as address spoofing detection, DDOS mitigation, and transparent encrypted tunnels can be applied to GUE packets.

Advanced security extensions for Generic UDP Encapsulation, including security for the GUE header and payload, are described in detail in [GUESEC].

Encapsulation of IP protocols within GUE should not increase security risk, nor provide additional security in itself. A hash function for computing flow entropy (section 5.11) should be randomly seeded to mitigate some possible denial service attacks.

8. IANA Consideration

8.1. UDP source port

A user UDP port number assignment for GUE has been assigned:

```
Service Name: gue
Transport Protocol(s): UDP
Assignee: Tom Herbert <therbert@google.com>
Contact: Tom Herbert <therbert@google.com>
Description: Generic UDP Encapsulation
Reference: draft-herbert-gue
Port Number: 6080
Service Code: N/A
Known Unauthorized Uses: N/A
Assignment Notes: N/A
```

8.2. GUE version number

IANA is requested to set up a registry for the GUE version number. The GUE version number is 2 bits containing four possible values. This document defines version 0 and 1. New values are assigned via Standards Action [RFC5226].

Version number	Description	Reference
0	Version 0	This document
1	Version 1	This document
2..3	Unassigned	

8.3. Control types

IANA is requested to set up a registry for the GUE control types. Control types are 8 bit values. New values are assigned via Standards Action [RFC5226].

Control type	Description	Reference
0	Need further interpretation	This document
1..127	Unassigned	
128..255	User defined	This document

8.4. Flag-fields

IANA is requested to create a "GUE flag-fields" registry to allocate flags and optional fields for the GUE header flags and extension fields. This shall be a registry of bit assignments for flags, length of optional fields for corresponding flags, and descriptive strings. There are sixteen bits for primary GUE header flags (bit number 0-15). New values are assigned via Standards Action [RFC5226].

Flags bits	Field size	Description	Reference
Bit 0	4 bytes	VNID	[GUE4NVO3]
Bit 1..2	01->8 bytes 10->16 bytes 11->32 bytes	Security	[GUEEXTENS]
Bit 3	4 bytes	Checksum	[GUEEXTENS]
Bit 4	8 bytes	Fragmen- tation	[GUEEXTENS]
Bit 5	4 bytes	Payload transform	[GUEEXTENS]
Bit 6	4 bytes	Remote checksum offload	[GUEEXTENS]
Bit 7..15		Unassigned	

New flags are to be allocated from high to low order bit contiguously without holes.

9. Acknowledgements

The authors would like to thank David Liu, Erik Nordmark, Fred Templin, and Adrian Farrel for valuable input on this draft.

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>.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, DOI 10.17487/RFC2434, October 1998, <http://www.rfc-editor.org/info/rfc2434>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000,

<http://www.rfc-editor.org/info/rfc2983>.

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

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, DOI 10.17487/RFC4459, April 2006, <http://www.rfc-editor.org/info/rfc4459>.

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

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, DOI 10.17487/RFC2434, October 1998, <http://www.rfc-editor.org/info/rfc2434>.

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

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

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, DOI 10.17487/RFC4459, April 2006, <http://www.rfc-editor.org/info/rfc4459>.

10.2. Informative References

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <http://www.rfc-editor.org/info/rfc2003>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <http://www.rfc-editor.org/info/rfc3948>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

[RFC3378] Housley, R. and S. Hollenbeck, "EtherIP: Tunneling Ethernet Frames in IP Datagrams", RFC 3378, DOI 10.17487/RFC3378, September 2002, <http://www.rfc-editor.org/info/rfc3378>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005, <http://www.rfc-editor.org/info/rfc4023>.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <http://www.rfc-editor.org/info/rfc2661>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004, <http://www.rfc-editor.org/info/rfc3828>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605,

August 2015, <http://www.rfc-editor.org/info/rfc7605>.

[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>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <http://www.rfc-editor.org/info/rfc4340>.

[GUEEXTENS] Herbert, T., Yong, L., and Templin, F., "Extensions for Generic UDP Encapsulation" draft-herbert-gue-extensions-00

[GUE4NVO3] Yong, L., Herbert, T., Zia, O., "Generic UDP Encapsulation (GUE) for Network Virtualization Overlay" draft-hy-nvo3-gue-4-nvo-03

[TCPUDP] Chesire, S., Graessley, J., and McGuire, R., "Encapsulation of TCP and other Transport Protocols over UDP" draft-cheshire-tcp-over-udp-00

[GREUDP] Crabbe, E., Yong, L., Xu, X., and Herbert, T.,Cpsest "Generic UDP Encapsulation for IP Tunneling" draft-ietf-tsvwg-gre-in-udp-encap-13

[GUT] Manner, J., Varia, N., and Briscoe, B., "Generic UDP Tunnelling (GUT) draft-manner-tsvwg-gut-02.txt"

[CIRCBRK] Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15

Appendix A: NIC processing for GUE

This appendix provides some guidelines for Network Interface Cards (NICs) to implement common offloads and accelerations to support GUE. Note that most of this discussion is generally applicable to other methods of UDP based encapsulation.

This appendix is informational and does not constitute a normative part of this document.

A.1. Receive multi-queue

Contemporary NICs support multiple receive descriptor queues (multi-queue). Multi-queue enables load balancing of network processing for a NIC across multiple CPUs. On packet reception, a NIC must select the appropriate queue for host processing. Receive Side Scaling is a common method which uses the flow hash for a packet to index an indirection table where each entry stores a queue number. Flow Director and Accelerated Receive Flow Steering (aRFS) allow a host to program the queue that is used for a given flow which is identified either by an explicit five-tuple or by the flow's hash.

GUE encapsulation should be compatible with multi-queue NICs that support five-tuple hash calculation for UDP/IP packets as input to RSS. The flow entropy in the UDP source port ensures classification of the encapsulated flow even in the case that the outer source and destination addresses are the same for all flows (e.g. all flows are going over a single tunnel).

By default, UDP RSS support is often disabled in NICs to avoid out of order reception that can occur when UDP packets are fragmented. As discussed above, fragmentation of GUE packets should be mitigated by fragmenting packets before entering a tunnel, path MTU discovery in higher layer protocols, or operator adjusting MTUs. Other UDP traffic may not implement such procedures to avoid fragmentation, so enabling UDP RSS support in the NIC should be a considered tradeoff during configuration.

A.2. Checksum offload

Many NICs provide capabilities to calculate standard ones complement payload checksum for packets in transmit or receive. When using GUE encapsulation there are at least two checksums that may be of interest: the encapsulated packet's transport checksum, and the UDP checksum in the outer header.

A.2.1. Transmit checksum offload

NICs may provide a protocol agnostic method to offload transmit checksum (NETIF_F_HW_CSUM in Linux parlance) that can be used with GUE. In this method the host provides checksum related parameters in a transmit descriptor for a packet. These parameters include the starting offset of data to checksum, the length of data to checksum, and the offset in the packet where the computed checksum is to be written. The host initializes the checksum field to pseudo header checksum.

In the case of GUE, the checksum for an encapsulated transport layer

packet, a TCP packet for instance, can be offloaded by setting the appropriate checksum parameters.

NICs typically can offload only one transmit checksum per packet, so simultaneously offloading both an inner transport packet's checksum and the outer UDP checksum is likely not possible. In this case setting UDP checksum to zero (per above discussion) and offloading the inner transport packet checksum might be acceptable.

If an encapsulator is co-resident with a host, then checksum offload may be performed using remote checksum offload (described in [GUEEXTENS]). Remote checksum offload relies on NIC offload of the simple UDP/IP checksum which is commonly supported even in legacy devices. In remote checksum offload the outer UDP checksum is set and the GUE header includes an option indicating the start and offset of the inner "offloaded" checksum. The inner checksum is initialized to the pseudo header checksum. When a decapsulator receives a GUE packet with the remote checksum offload option, it completes the offload operation by determining the packet checksum from the indicated start point to the end of the packet, and then adds this into the checksum field at the offset given in the option. Computing the checksum from the start to end of packet is efficient if checksum-complete is provided on the receiver.

A.2.2. Receive checksum offload

GUE is compatible with NICs that perform a protocol agnostic receive checksum (CHECKSUM_COMPLETE in Linux parlance). In this technique, a NIC computes a ones complement checksum over all (or some predefined portion) of a packet. The computed value is provided to the host stack in the packet's receive descriptor. The host driver can use this checksum to "patch up" and validate any inner packet transport checksum, as well as the outer UDP checksum if it is non-zero.

Many legacy NICs don't provide checksum-complete but instead provide an indication that a checksum has been verified (CHECKSUM_UNNECESSARY in Linux). Usually, such validation is only done for simple TCP/IP or UDP/IP packets. If a NIC indicates that a UDP checksum is valid, the checksum-complete value for the UDP packet is the "not" of the pseudo header checksum. In this way, checksum-unnecessary can be converted to checksum-complete. So if the NIC provides checksum-unnecessary for the outer UDP header in an encapsulation, checksum conversion can be done so that the checksum-complete value is derived and can be used by the stack to validate an checksums in the encapsulated packet.

A.3. Transmit Segmentation Offload

Transmit Segmentation Offload (TSO) is a NIC feature where a host

provides a large (>MTU size) TCP packet to the NIC, which in turn splits the packet into separate segments and transmits each one. This is useful to reduce CPU load on the host.

The process of TSO can be generalized as:

- Split the TCP payload into segments which allow packets with size less than or equal to MTU.

- For each created segment:

  1. Replicate the TCP header and all preceding headers of the original packet.

  2. Set payload length fields in any headers to reflect the length of the segment.

  3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.

  4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation in GUE. For each segment the Ethernet, outer IP, UDP header, GUE header, inner IP header if tunneling, and TCP headers are replicated. Any packet length header fields need to be set properly (including the length in the outer UDP header), and checksums need to be set correctly (including the outer UDP checksum if being used).

To facilitate TSO with GUE it is recommended that optional fields should not contain values that must be updated on a per segment basis-- for example the GUE fields should not include checksums, lengths, or sequence numbers that refer to the payload. If the GUE header does not contain such fields then the TSO engine only needs to copy the bits in the GUE header when creating each segment and does not need to parse the GUE header.

A.4. Large Receive Offload

Large Receive Offload (LRO) is a NIC feature where packets of a TCP connection are reassembled, or coalesced, in the NIC and delivered to the host as one large packet. This feature can reduce CPU utilization in the host.

LRO requires significant protocol awareness to be implemented correctly and is difficult to generalize. Packets in the same flow

need to be unambiguously identified. In the presence of tunnels or network virtualization, this may require more than a five-tuple match (for instance packets for flows in two different virtual networks may have identical five-tuples). Additionally, a NIC needs to perform validation over packets that are being coalesced, and needs to fabricate a single meaningful header from all the coalesced packets.

The conservative approach to supporting LRO for GUE would be to assign packets to the same flow only if they have identical five-tuple and were encapsulated the same way. That is the outer IP addresses, the outer UDP ports, GUE protocol, GUE flags and fields, and inner five tuple are all identical.

Appendix B: Implementation considerations

This appendix is informational and does not constitute a normative part of this document.

B.1. Priveleged ports

Using the source port to contain a flow entropy value disallows the security method of a receiver enforcing that the source port be a privileged port. Privileged ports are defined by some operating systems to restrict source port binding. Unix, for instance, considered port number less than 1024 to be privileged.

Enforcing that packets are sent from a privileged port is widely considered an inadequate security mechanism and has been mostly deprecated. To approximate this behavior, an implementation could restrict a user from sending a packet destined to the GUE port without proper credentials.

B.2. Setting flow entropy as a route selector

An encapsulator generating flow entropy in the UDP source port may modulate the value to perform a type of multipath source routing. Assuming that networking switches perform ECMP based on the flow hash, a sender can affect the path by altering the flow entropy. For instance, a host may store a flow hash in its PCB for an inner flow, and may alter the value upon detecting that packets are traversing a lossy path. Changing the flow entropy for a flow should be subject to hysteresis (at most once every thirty seconds) to limit the number of out of order packets.

B.3. Hardware protocol implementation considerations

A low level protocol, such is GUE, is likely interesting to being supported by high speed network devices. Variable length header (VLH)

protocols like GUE are often considered difficult to efficiently implement in hardware. In order to retain the important characteristics of an extensible and robust protocol, hardware vendors may practice "constrained flexibility". In this model, only certain combinations or protocol header parameterizations are implemented in hardware fast path. Each such parameterization is fixed length so that the particular instance can be optimized as a fixed length protocol. In the case of GUE this constitutes specific combinations of GUE flags, fields, and next protocol. The selected combinations would naturally be the most common cases which form the "fast path", and other combinations are assumed to take the "slow path".

In time, needs and requirements of the protocol may change which may manifest themselves as new parameterizations to be supported in the fast path. To allow allow this extensibility, a device practicing constrained flexibility should allow the fast path parameterizations to be programmable.

Authors' Addresses

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA 94052
US

Email: tom@herbertland.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75024
US

Email: lucy.yong@huawei.com

Osama Zia
Microsoft
1 Microsoft Way
Redmond, WA 98029
US

Email: osamaz@microsoft.com

Transport Area Working Group
Internet-Draft
Updates: 3819 (if approved)
Intended status: Best Current Practice
Expires: January 9, 2017

B. Briscoe
Simula Research Laboratory
J. Kaippallimalil
Huawei
P. Thaler
Broadcom Corporation

July 8, 2016

Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP
draft-ietf-tsvwg-ecn-encap-guidelines-07

Abstract

The purpose of this document is to guide the design of congestion notification in any lower layer or tunnelling protocol that encapsulates IP. The aim is for explicit congestion signals to propagate consistently from lower layer protocols into IP. Then the IP internetwork layer can act as a portability layer to carry congestion notification from non-IP-aware congested nodes up to the transport layer (L4). Following these guidelines should assure interworking between new lower layer congestion notification mechanisms, whether specified by the IETF or other standards bodies.

Table of Contents

1. Introduction

The benefits of Explicit Congestion Notification (ECN) described below can only be fully realised if support for ECN is added to the relevant subnetwork technology, as well as to IP. When a lower layer buffer drops a packet obviously it does not just drop at that layer; the packet disappears from all layers. In contrast, when a lower layer marks a packet with ECN, the marking needs to be explicitly propagated up the layers. The same is true if a buffer marks the outer header of a packet that encapsulates inner tunnelled headers. Forwarding ECN is not as straightforward as other headers because it has to be assumed ECN may be only partially deployed. If an egress at any layer is not ECN-aware, or if the ultimate receiver or sender is not ECN-aware, congestion needs to be indicated by dropping a packet, not marking it.

The purpose of this document is to guide the addition of congestion notification to any subnet technology or tunnelling protocol, so that lower layer equipment can signal congestion explicitly and it will propagate consistently into encapsulated (higher layer) headers, otherwise the signals will not reach their ultimate destination.

ECN is defined in the IP header (v4 and v6) [RFC3168] to allow a resource to notify the onset of queue build-up without having to drop packets, by explicitly marking a proportion of packets with the congestion experienced (CE) codepoint.

Given a suitable marking scheme, ECN removes nearly all congestion loss and it cuts delays for two main reasons:

- It avoids the delay when recovering from congestion losses, which particularly benefits small flows or real-time flows, making their delivery time predictably short [RFC2884];

- As ECN is used more widely by end-systems, it will gradually remove the need to configure a degree of delay into buffers before they start to notify congestion (the cause of bufferbloat). This is because drop involves a trade-off between sending a timely signal and trying to avoid impairment, whereas ECN is solely a signal not an impairment, so there is no harm triggering it earlier.

Some lower layer technologies (e.g. MPLS, Ethernet) are used to form subnetworks with IP-aware nodes only at the edges. These networks are often sized so that it is rare for interior queues to overflow. However, until recently this was more due to the inability of TCP to saturate the links. For many years, fixes such as window scaling [RFC1323] proved hard to deploy. And the New Reno variant of TCP has

remained in widespread use despite its inability to scale to high flow rates. However, now that modern operating systems are finally capable of saturating interior links, even the buffers of well-provisioned interior switches will need to signal episodes of queuing.

Propagation of ECN is defined for MPLS [RFC5129], and is being defined for TRILL [RFC7780], [I-D.eastlake-trill-ecn-support], but it remains to be defined for a number of other subnetwork technologies.

Similarly, ECN propagation is yet to be defined for many tunnelling protocols. [RFC6040] defines how ECN should be propagated for IP-in-IP [RFC2003] and IPsec [RFC4301] tunnels, and it is cited by more recent tunnelling protocols, e.g. Generic UDP Encapsulation (GUE) [I-D.ietf-nvo3-gue] and Geneve [I-D.ietf-nvo3-geneve]. However, as Section 9.3 of RFC3168 pointed out, ECN support will need to be defined for other tunnelling protocols, e.g. L2TP [RFC2661], GRE [RFC1701], [RFC2784], PPTP [RFC2637] and GTP [GTPv1], [GTPv1-U], [GTPv2-C], VXLAN [RFC7348].

Incremental deployment is the most delicate aspect when adding support for ECN. The original ECN protocol in IP [RFC3168] was carefully designed so that a congested buffer would not mark a packet (rather than drop it) unless both source and destination hosts were ECN-capable. Otherwise its congestion markings would never be detected and congestion would just build up further. However, to support congestion marking below the IP layer, it is not sufficient to only check that the two end-points support ECN; correct operation also depends on the decapsulator at each subnet egress faithfully propagating congestion notifications to the higher layer. Otherwise, a legacy decapsulator might silently fail to propagate any ECN signals from the outer to the forwarded header. Then the lost signals would never be detected and again congestion would build up further. The guidelines given later require protocol designers to carefully consider incremental deployment, and suggest various safe approaches for different circumstances.

Of course, the IETF does not have standards authority over every link layer protocol. So this document gives guidelines for designing propagation of congestion notification across the interface between IP and protocols that may encapsulate IP (i.e. that can be layered beneath IP). Each lower layer technology will exhibit different issues and compromises, so the IETF or the relevant standards body must be free to define the specifics of each lower layer congestion notification scheme. Nonetheless, if the guidelines are followed, congestion notification should interwork between different technologies, using IP in its role as a ’portability layer’.

Therefore, the capitalised term 'SHOULD' or 'SHOULD NOT' are often used in preference to 'MUST' or 'MUST NOT', because it is difficult to know the compromises that will be necessary in each protocol design. If a particular protocol design chooses to contradict a 'SHOULD (NOT)' given in the advice below, it MUST include a sound justification.

It has not been possible to give common guidelines for all lower layer technologies, because they do not all fit a common pattern. Instead they have been divided into a few distinct modes of operation: feed-forward-and-upward; feed-upward-and-forward; feed-backward; and null mode. These modes are described in Section 4, then in the following sections separate guidelines are given for each mode.

This document updates the advice to subnetwork designers about ECN in Section 13 of [RFC3819].

1.1. Scope

This document only concerns wire protocol processing of explicit notification of congestion and makes no changes or recommendations concerning algorithms for congestion marking or for congestion response (algorithm issues should be independent of the layer the algorithm operates in).

The question of congestion notification signals with different semantics to those of ECN in IP is touched on in a couple of specific cases (e.g. QCN [IEEE802.1Qau]) and with schemes with multiple severity levels such as PCN [RFC6660]). However, no attempt is made to give guidelines about schemes with different semantics that are yet to be invented.

The semantics of congestion signals can be relative to the traffic class. Therefore correct propagation of congestion signals could depend on correct propagation of any traffic class field between the layers. In this document, correct propagation of traffic class information is assumed, while what 'correct' means and how it is achieved is covered elsewhere (e.g. [RFC2983]) and is outside the scope of the present document.

Note that these guidelines do not require the subnet wire protocol to be changed to accommodate congestion notification. Another way to add congestion notification without consuming header space in the subnet protocol might be to use a parallel control plane protocol.

This document focuses on the congestion notification interface between IP and lower layer protocols that can encapsulate IP, where

the term 'IP' includes v4 or v6, unicast, multicast or anycast. However, it is likely that the guidelines will also be useful when a lower layer protocol or tunnel encapsulates itself (e.g. Ethernet MAC in MAC [IEEE802.1Qah]) or when it encapsulates other protocols. In the feed-backward mode, propagation of congestion signals for multicast and anycast packets is out-of-scope (because it would be so complicated that it is hoped no-one would attempt such an abomination).

2. Terminology

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

Further terminology used within this document:

Protocol data unit (PDU): Information that is delivered as a unit among peer entities of a layered network consisting of protocol control information (typically a header) and possibly user data (payload) of that layer. The scope of this document includes layer 2 and layer 3 networks, where the PDU is respectively termed a frame or a packet (or a cell in ATM). PDU is a general term for any of these. This definition also includes a payload with a shim header lying somewhere between layer 2 and 3.

Transport: The end-to-end transmission control function, conventionally considered at layer-4 in the OSI reference model. Given the audience for this document will often use the word transport to mean low level bit carriage, whenever the term is used it will be qualified, e.g. 'L4 transport'.

Encapsulator: The link or tunnel endpoint function that adds an outer header to a PDU (also termed the 'link ingress', the 'subnet ingress', the 'ingress tunnel endpoint' or just the 'ingress' where the context is clear).

Decapsulator: The link or tunnel endpoint function that removes an outer header from a PDU (also termed the 'link egress', the 'subnet egress', the 'egress tunnel endpoint' or just the 'egress' where the context is clear).

Incoming header: The header of an arriving PDU before encapsulation.

Outer header: The header added to encapsulate a PDU.

Inner header: The header encapsulated by the outer header.

Outgoing header: The header forwarded by the decapsulator.

CE: Congestion Experienced [RFC3168]

ECT: ECN-Capable Transport [RFC3168]

Not-ECT: Not ECN-Capable Transport [RFC3168]

Load Regulator: For each flow of PDUs, the transport function that is capable of controlling the data rate. Typically located at the data source, but in-path nodes can regulate load in some congestion control arrangements (e.g. admission control, policing nodes or transport circuit-breakers [I-D.ietf-tsvwg-circuit-breaker]). Note the term "a function capable of controlling the load" deliberately includes a transport that doesn't actually control the load responsively but ideally it ought to (e.g. a sending application without congestion control that uses UDP).

ECN-PDU: A PDU that is part of a feedback loop within which all the nodes that need to propagate explicit congestion notifications back to the Load Regulator are ECN-capable. An IP packet with a non-zero ECN field implies that the endpoints are ECN-capable, so this would be an ECN-PDU. However, ECN-PDU is intended to be a general term for a PDU at any layer, not just IP.

Not-ECN-PDU: A PDU that is part of a feedback-loop within which some nodes necessary to propagate explicit congestion notifications back to the load regulator are not ECN-capable.

Congestion Baseline: The location of the function on the path that initialised the values of all congestion notification fields in a sequence of packets, before any are set to the congestion experienced (CE) codepoint if they experience congestion further downstream. Typically the original data source at layer-4.

3. Guidelines in All Cases

RFC 3168 specifies that the ECN field in the IP header is intended to be marked by active queue management algorithms. Any congestion notification from an algorithm that does not conform to the recommendations in [RFC7567] MUST NOT be propagated from a lower layer into the ECN field in IP (see also [RFC4774] on alternate uses of the ECN field).

4. Modes of Operation

This section sets down the different modes by which congestion information is passed between the lower layer and the higher one. It acts as a reference framework for the following sections, which give normative guidelines for designers of explicit congestion notification protocols, taking each mode in turn:

Feed-Forward-and-Up: Nodes feed forward congestion notification towards the egress within the lower layer then up and along the layers towards the end-to-end destination at the transport layer. The following local optimisation is possible:

Feed-Up-and-Forward: A lower layer switch feeds-up congestion notification directly into the ECN field in the higher layer (e.g. IP) header, irrespective of whether the node is at the egress of a subnet.

Feed-Backward: Nodes feed back congestion signals towards the ingress of the lower layer and (optionally) attempt to control congestion within their own layer.

Null: Nodes cannot experience congestion at the lower layer except at ingress nodes (which are IP-aware or equivalently higher-layer-aware).

4.1. Feed-Forward-and-Up Mode

Like IP and MPLS, many subnet technologies are based on self-contained protocol data units (PDUs) or frames sent unreliably. They provide no feedback channel at the subnetwork layer, instead relying on higher layers (e.g. TCP) to feed back loss signals.

In these cases, ECN may best be supported by standardising explicit notification of congestion into the lower layer protocol that carries the data forwards. It will then also be necessary to define how the egress of the lower layer subnet propagates this explicit signal into the forwarded upper layer (IP) header. It can then continue forwards until it finally reaches the destination transport (at L4). Then typically the destination will feed this congestion notification back to the source transport using an end-to-end protocol (e.g. TCP). This is the arrangement that has already been used to add ECN to IP-in-IP tunnels [RFC6040], IP-in-MPLS and MPLS-in-MPLS [RFC5129].

This mode is illustrated in Figure 1. Along the middle of the figure, layers 2, 3 and 4 of the protocol stack are shown, and one packet is shown along the bottom as it progresses across the network from source to destination, crossing two subnets connected by a

router, and crossing two switches on the path across each subnet. Congestion at the output of the first switch (shown as *) leads to a congestion marking in the L2 header (shown as C in the illustration of the packet). The chevrons show the progress of the resulting congestion indication. It is propagated from link to link across the subnet in the L2 header, then when the router removes the marked L2 header, it propagates the marking up into the L3 (IP) header. The router forwards the marked L3 header into subnet 2, and when it adds a new L2 header it copies the L3 marking into the L2 header as well, as shown by the 'C's in both layers (assuming the technology of subnet 2 also supports explicit congestion marking).

Note that there is no implication that each 'C' marking is encoded the same; a different encoding might be used for the 'C' marking in each protocol.

Finally, for completeness, we show the L3 marking arriving at the destination, where the host transport protocol (e.g. TCP) feeds it back to the source in the L4 acknowledgement (the 'C' at L4 in the packet at the top of the diagram).

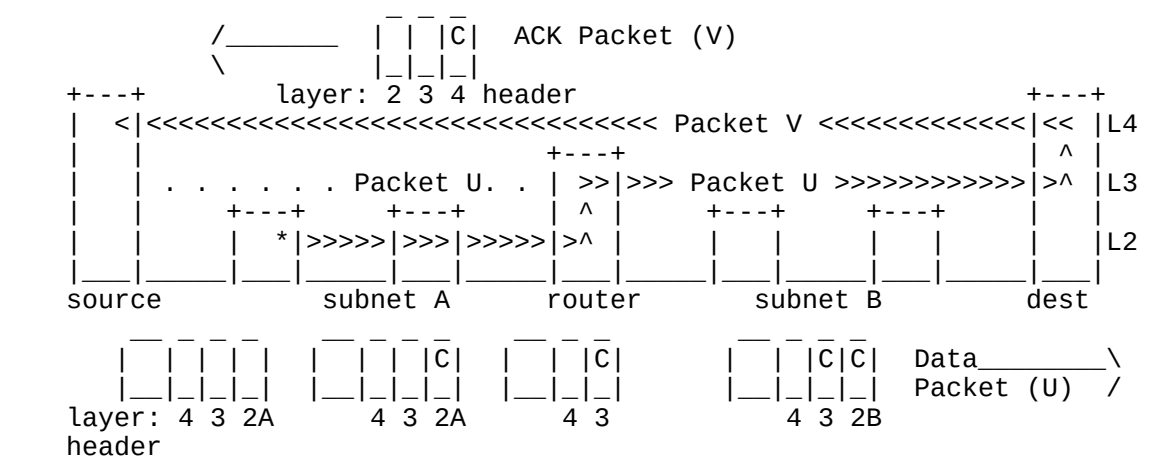


Figure 1: Feed-Forward-and-Up Mode

Of course, modern networks are rarely as simple as this text-book example, often involving multiple nested layers. For example, a 3GPP mobile network may have two IP-in-IP (GTP) tunnels in series and an MPLS backhaul between the base station and the first router. Nonetheless, the example illustrates the general idea of feeding congestion notification forward then upward whenever a header is removed at the egress of a subnet.

Note that the FECN (forward ECN) bit in Frame Relay and the explicit forward congestion indication (EFCI [ITU-T.I.371]) bit in ATM user data cells follow a feed-forward pattern. However, in ATM, this arrangement is only part of a feed-forward-and-backward pattern at the lower layer, not feed-forward-and-up out of the lower layer--the intention was never to interface to IP ECN at the subnet egress. To our knowledge, Frame Relay FECN is solely used to detect where more capacity should be provisioned [Buck00].

4.2. Feed-Up-and-Forward Mode

Ethernet is particularly difficult to extend incrementally to support explicit congestion notification. One way to support ECN in such cases has been to use so called 'layer-3 switches'. These are Ethernet switches that bury into the Ethernet payload to find an IP header and manipulate or act on certain IP fields (specifically Diffserv & ECN). For instance, in Data Center TCP [DCTCP], layer-3 switches are configured to mark the ECN field of the IP header within the Ethernet payload when their output buffer becomes congested. With respect to switching, a layer-3 switch acts solely on the addresses in the Ethernet header; it doesn't use IP addresses, and it doesn't decrement the TTL field in the IP header.

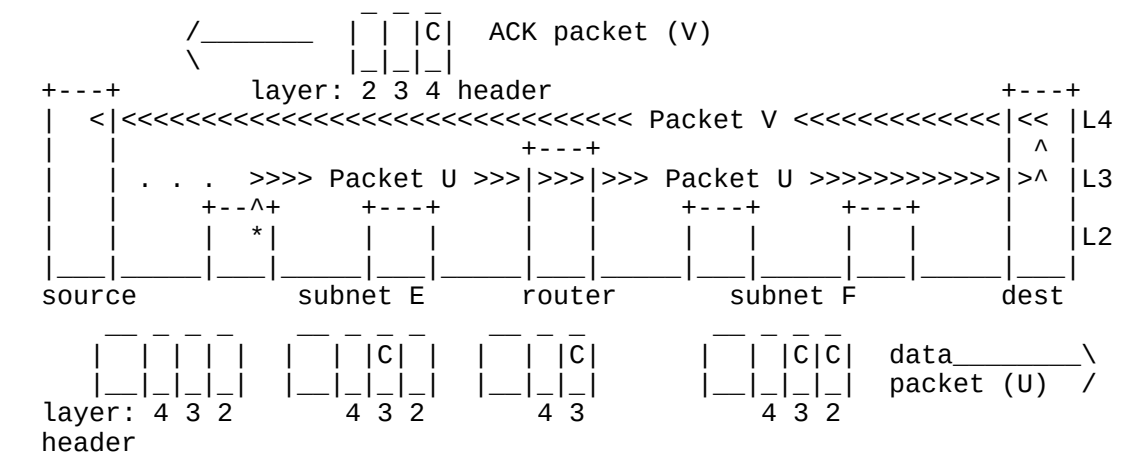


Figure 2: Feed-Up-and-Forward Mode

By comparing Figure 2 with Figure 1, it can be seen that subnet E (perhaps a subnet of layer-3 Ethernet switches) works in feed-up-and-forward mode by notifying congestion directly into L3 at the point of congestion, even though the congested switch does not otherwise act at L3. In this example, the technology in subnet F (e.g. MPLS) does

support ECN natively, so when the router adds the layer-2 header it copies the ECN marking from L3 to L2 as well.

4.3. Feed-Backward Mode

In some layer 2 technologies, explicit congestion notification has been defined for use internally within the subnet with its own feedback and load regulation, but typically the interface with IP for ECN has not been defined.

For instance, for the available bit-rate (ABR) service in ATM, the relative rate mechanism was one of the more popular mechanisms for managing traffic, tending to supersede earlier designs. In this approach ATM switches send special resource management (RM) cells in both the forward and backward directions to control the ingress rate of user data into a virtual circuit. If a switch buffer is approaching congestion or is congested it sends an RM cell back towards the ingress with respectively the No Increase (NI) or Congestion Indication (CI) bit set in its message type field [ATM-TM-ABR]. The ingress then holds or decreases its sending bit-rate accordingly.

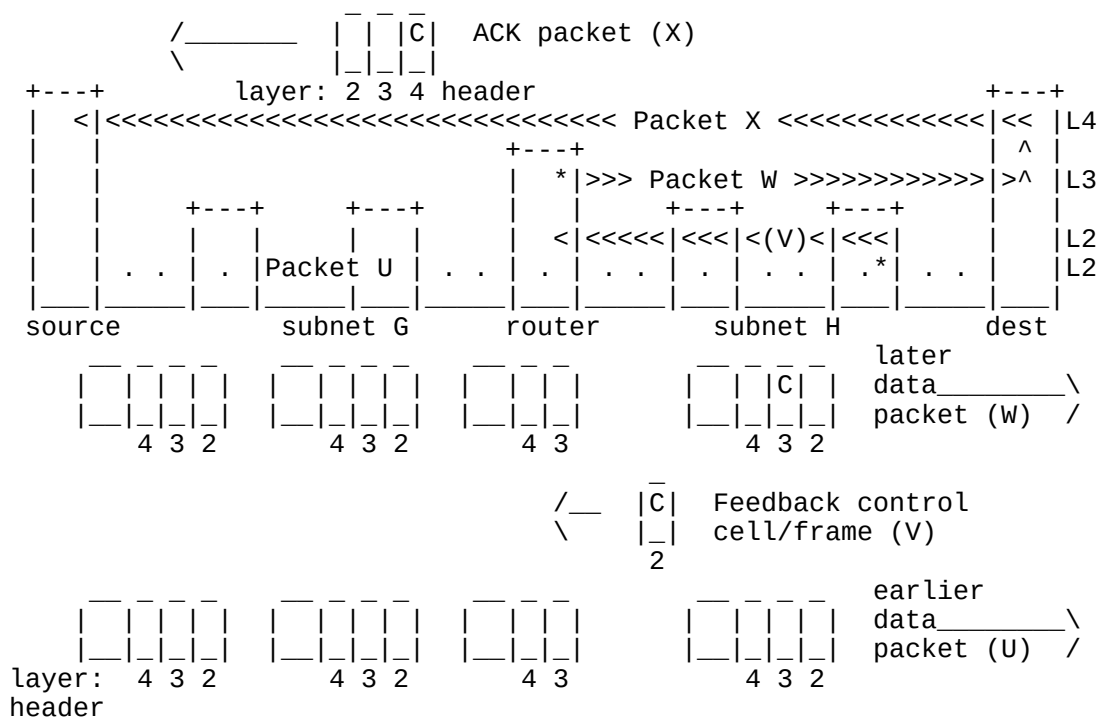


Figure 3: Feed-Backward Mode

ATM's feed-backward approach doesn't fit well when layered beneath IP's feed-forward approach--unless the initial data source is the same node as the ATM ingress. Figure 3 shows the feed-backward approach being used in subnet H. If the final switch on the path is congested (*), it doesn't feed-forward any congestion indications on packet (U). Instead it sends a control cell (V) back to the router at the ATM ingress.

However, the backward feedback doesn't reach the original data source directly because IP doesn't support backward feedback (and subnet G is independent of subnet H). Instead, the router in the middle throttles down its sending rate but the original data sources don't reduce their rates. The resulting rate mismatch causes the middle router's buffer at layer 3 to back up until it becomes congested, which it signals forwards on later data packets at layer 3 (e.g. packet W). Note that the forward signal from the middle router is not triggered directly by the backward signal. Rather, it is triggered by congestion resulting from the middle router's mismatched rate response to the backward signal.

In response to this later forward signalling, end-to-end feedback at layer-4 finally completes the tortuous path of congestion indications back to the origin data source, as before.

4.4. Null Mode

Often link and physical layer resources are 'non-blocking' by design. In these cases congestion notification may be implemented but it does not need to be deployed at the lower layer; ECN in IP would be sufficient.

A degenerate example is a point-to-point Ethernet link. Excess loading of the link merely causes the queue from the higher layer to back up, while the lower layer remains immune to congestion. Even a whole meshed subnetwork can be made immune to interior congestion by limiting ingress capacity and sufficient sizing of interior links, e.g. a non-blocking fat-tree network. An alternative to fat links near the root is numerous thin links with multi-path routing to ensure even worst-case patterns of load cannot congest any link, e.g. a Clos network.

5. Feed-Forward-and-Up Mode: Guidelines for Adding Congestion Notification

Feed-forward-and-up is the mode already used for signalling ECN up the layers through MPLS into IP [RFC5129] and through IP-in-IP tunnels [RFC6040]. These RFCs take a consistent approach and the following guidelines are designed to ensure this consistency continues as ECN support is added to other protocols that encapsulate IP. The guidelines are also designed to ensure compliance with the more general best current practice for the design of alternate ECN schemes given in [RFC4774].

The rest of this section is structured as follows:

- Section 5.1 addresses the most straightforward cases, where [RFC6040] can be applied directly to add ECN to tunnels that are effectively the same as IP-in-IP tunnels.

- The subsequent sections give guidelines for adding ECN to a subnet technology that uses feed-forward-and-up mode like IP, but it is not so similar to IP that [RFC6040] rules can be applied directly. Specifically:

  * Sections 5.2, 5.3 and 5.4 respectively address how to add ECN support to the wire protocol and to the encapsulators and decapsulators at the ingress and egress of the subnet.

* Section 5.5 deals with the special, but common, case of sequences of tunnels or subnets that all use the same technology

* Section 5.6 deals with the question of reframing when IP packets do not map 1:1 into lower layer frames.

5.1. IP-in-IP Tunnels with Tightly Coupled Shim Headers

A common pattern for many tunnelling protocols is to encapsulate an inner IP header with shim header(s) then an outer IP header. In many cases the shim header(s) and the outer IP header are always added (or removed) as part of the same process. We call this a tightly coupled shim header. Processing the shim and outer together is often necessary because the shim(s) are not sufficient for packet forwarding in their own right; not unless complemented by an outer header.

For all such tightly coupled shim headers (such as those listed in the Introduction), the rules in [RFC6040] for propagating the ECN field can be applied directly between the inner and outer IP headers. [I-D.briscoe-tsvwg-rfc6040bis] clarifies that RFC 6040 is just as applicable when there is a tightly-coupled shim between two IP headers as wehen there is not.

5.2. Wire Protocol Design: Indication of ECN Support

This section is intended to guide the redesign of any lower layer protocol that encapsulate IP to add native ECN support at the lower layer. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A lower layer (or subnet) congestion notification system:

1. SHOULD NOT apply explicit congestion notifications to PDUs that are destined for legacy layer-4 transport implementations that will not understand ECN, and

2. SHOULD NOT apply explicit congestion notifications to PDUs if the egress of the subnet might not propagate congestion notifications onward into the higher layer.

   We use the term ECN-PDUs for a PDU on a feedback loop that will propagate congestion notification properly because it meets both the above criteria. And a Not-ECN-PDU is a PDU on a feedback loop that does not meet both criteria, and will therefore not

propagate congestion notification properly. A corollary of the above is that a lower layer congestion notification protocol:

3. SHOULD be able to distinguish ECN-PDUs from Not-ECN-PDUs.

Note that there is no need for all interior nodes within a subnet to be able to mark congestion explicitly. A mix of ECN and drop signals from different nodes is fine. However, if _any_ interior nodes might generate ECN markings, guideline 2 above says that all relevant egress node(s) SHOULD be able to propagate those markings up to the higher layer.

In IP, if the ECN field in each PDU is cleared to the Not-ECT (not ECN-capable transport) codepoint, it indicates that the L4 transport will not understand congestion markings. A congested buffer must not mark these Not-ECT PDUs, and therefore drops them instead.

The mechanism a lower layer uses to distinguish the ECN-capability of PDUs need not mimic that of IP. The above guidelines merely say that the lower layer system, as a whole, should achieve the same outcome. For instance, ECN-capable feedback loops might use PDUs that are identified by a particular set of labels or tags. Alternatively, logical link protocols that use flow state might determine whether a PDU can be congestion marked by checking for ECN-support in the flow state. Other protocols might depend on out-of-band control signals.

The per-domain checking of ECN support in MPLS [RFC5129] is a good example of a way to avoid sending congestion markings to transports that will not understand them, without using any header space in the subnet protocol.

In MPLS, header space is extremely limited, therefore RFC5129 does not provide a field in the MPLS header to indicate whether the PDU is an ECN-PDU or a Not-ECN-PDU. Instead, interior nodes in a domain are allowed to set explicit congestion indications without checking whether the PDU is destined for a transport that will understand them. Nonetheless, this is made safe by requiring that the network operator upgrades all decapsulating edges of a whole domain at once, as soon as even one switch within the domain is configured to mark rather than drop during congestion. Therefore, any edge node that might decapsulate a packet will be capable of checking whether the higher layer transport is ECN-capable. When decapsulating a CE-marked packet, if the decapsulator discovers that the higher layer (inner header) indicates the transport is not ECN-capable, it drops the packet--effectively on behalf of the earlier congested node (see Decapsulation Guideline 1 in Section 5.4).

It was only appropriate to define such an incremental deployment strategy because MPLS is targeted solely at professional operators, who can be expected to ensure that a whole subnetwork is consistently configured. This strategy might not be appropriate for other link technologies targeted at zero-configuration deployment or deployment by the general public (e.g. Ethernet). For such 'plug-and-play' environments it will be necessary to invent a failsafe approach that ensures congestion markings will never fall into black holes, no matter how inconsistently a system is put together. Alternatively, congestion notification relying on correct system configuration could be confined to flavours of Ethernet intended only for professional network operators, such as IEEE 802.1ah Provider Backbone Bridges (PBB).

ECN support in TRILL [I-D.eastlake-trill-ecn-support] provides a good example of how to add ECN to a lower layer protocol without relying on careful and consistent operator configuration. TRILL provides an extension header word with space for flags of different categories depending on whether logic to understand the extension is critical. The congestion experienced marking has been defined as a 'critical ingress-to-egress' flag. So if a transit RBridge sets this flag and an egress RBridge does not have any logic to process it, it will drop it; which is the desired default action anyway. Therefore TRILL RBridges can be updated with support for ECN in no particular order and, at the egress of the TRILL campus, congestion notification will be propagated to IP as ECN whenever ECN logic has been implemented, and as drop otherwise.

QCN [IEEE802.1Qau] provides another example of how to indicate to lower layer devices that the end-points will not understand ECN. An operator can define certain 802.1p classes of service to indicate non-QCN frames and an ingress bridge is required to map arriving not-QCN-capable IP packets to one of these non-QCN 802.1p classes.

5.3. Encapsulation Guidelines

This section is intended to guide the redesign of any node that encapsulates IP with a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

1. Egress Capability Check: A subnet ingress needs to be sure that the corresponding egress of a subnet will propagate any congestion notification added to the outer header across the subnet. This is necessary in addition to checking that an

incoming PDU indicates an ECN-capable (L4) transport. Examples of how this guarantee might be provided include:

* by configuration (e.g. if any label switches in a domain support ECN marking, [RFC5129] requires all egress nodes to have been configured to propagate ECN)

* by the ingress explicitly checking that the egress propagates ECN (e.g. TRILL uses IS-IS to check path capabilities before using critical options [RFC7780])

* by inherent design of the protocol (e.g. by encoding ECN marking on the outer header in such a way that a legacy egress that does not understand ECN will consider the PDU corrupt and discard it, thus at least propagating a form of congestion signal).

2. Egress Fails Capability Check: If the ingress cannot guarantee that the egress will propagate congestion notification, the ingress SHOULD disable ECN when it forwards the PDU at the lower layer. An example of how the ingress might disable ECN at the lower layer would be by setting the outer header of the PDU to identify it as a Not-ECN-PDU, assuming the subnet technology supports such a concept.

3. Standard Congestion Monitoring Baseline: Once the ingress to a subnet has established that the egress will correctly propagate ECN, on encapsulation it SHOULD encode the same level of congestion in outer headers as is arriving in incoming headers. For example it might copy any incoming congestion notification into the outer header of the lower layer protocol.

   This ensures that all outer headers reflect congestion accumulated along the whole upstream path since the Load Regulator, not just since the ingress of the subnet. A node that is not the Load Regulator SHOULD NOT re-initialise the level of CE markings in the outer to zero.

   This guideline is intended to ensure that any bulk congestion monitoring of outer headers (e.g. by a network management node monitoring ECN in passing frames) is most meaningful. For instance, if an operator measures CE in 0.4% of passing outer headers, this information is only useful if the operator knows where the proportion of CE markings was last initialised to 0% (the Congestion Baseline). Such monitoring information will not be useful if some subnet ingress nodes reset all outer CE markings while others copy incoming CE markings into the outer.

Most information can be extracted if the Congestion Baseline is standardised at the node that is regulating the load (the Load Regulator--typically the data source). Then the operator can measure both congestion since the Load Regulator, and congestion since the subnet ingress. The latter might be measurable by subtracting the level of CE markings on inner headers from that on outer headers (see Appendix C of [RFC6040]).

5.4. Decapsulation Guidelines

This section is intended to guide the redesign of any node that decapsulates IP from within a lower layer header when adding native ECN support to the lower layer protocol. It reflects the approaches used in [RFC6040] and in [RFC5129]. Therefore IP-in-IP tunnels or IP-in-MPLS or MPLS-in-MPLS encapsulations that already comply with [RFC6040] or [RFC5129] will already satisfy this guidance.

A subnet egress SHOULD NOT simply copy congestion notification from outer headers to the forwarded header. It SHOULD calculate the outgoing congestion notification field from the inner and outer headers using the following guidelines. If there is any conflict, rules earlier in the list take precedence over rules later in the list:

1. If the arriving inner header is a Not-ECN-PDU it implies the L4 transport will not understand explicit congestion markings. Then:

   * If the outer header carries an explicit congestion marking, drop is the only indication of congestion that the L4 transport will understand. If the congestion marking is the most severe possible, the packet MUST be dropped. However, if congestion can be marked with multiple levels severity and the packet's marking is not the most severe, the packet MAY be forwarded, but it SHOULD be dropped.

   * If the outer is an ECN-PDU that carries no indication of congestion or a Not-ECN-PDU the PDU SHOULD be forwarded, but still as a Not-ECN-PDU.

2. If the outer header does not support explicit congestion notification (a Not-ECN-PDU), but the inner header does (an ECN-PDU), the inner header SHOULD be forwarded unchanged.

3. In some lower layer protocols congestion may be signalled as a numerical level, such as in the control frames of quantised congestion notification [IEEE802.1Qau]. If such a multi-bit encoding encapsulates an ECN-capable IP data packet, a function

will be needed to convert the quantised congestion level into the frequency of congestion markings in outgoing IP packets.

4. Congestion indications may be encoded by a severity level. For instance increasing levels of congestion might be encoded by numerically increasing indications, e.g. pre-congestion notification (PCN) can be encoded in each PDU at three severity levels in IP or MPLS [RFC6660].

   If the arriving inner header is an ECN-PDU, where the inner and outer headers carry indications of congestion of different severity, the more severe indication SHOULD be forwarded in preference to the less severe.

5. The inner and outer headers might carry a combination of congestion notification fields that should not be possible given any currently used protocol transitions. For instance, if Encapsulation Guideline 3 in Section 5.3 had been followed, it should not be possible to have a less severe indication of congestion in the outer than in the inner. It MAY be appropriate to log unexpected combinations of headers and possibly raise an alarm.

   If a safe outgoing codepoint can be defined for such a PDU, the PDU SHOULD be forwarded rather than dropped. Some implementers discard PDUs with currently unused combinations of headers just in case they represent an attack. However, an approach using alarms and policy-mediated drop is preferable to hard-coded drop, so that operators can keep track of possible attacks but currently unused combinations are not precluded from future use through new standards actions.

5.5. Sequences of Similar Tunnels or Subnets

In some deployments, particularly in 3GPP networks, an IP packet may traverse two or more IP-in-IP tunnels in sequence that all use identical technology (e.g. GTP).

In such cases, it would be sufficient for every encapsulation and decapsulation in the chain to comply with RFC 6040. Alternatively, as an optimisation, a node that decapsulates a packet and immediately re-encapsulates it for the next tunnel MAY copy the incoming outer ECN field directly to the outgoing outer and the incoming inner ECN field directly to the outgoing inner. Then the overall behavior across the sequence of tunnel segments would still be consistent with RFC 6040.

Appendix C of RFC6040 describes how a tunnel egress can monitor how much congestion has been introduced within a tunnel. A network operator might want to monitor how much congestion had been introduced within a whole sequence of tunnels. Using the technique in Appendix C of RFC6040 at the final egress, the operator could monitor the whole sequence of tunnels, but only if the above optimisation were used consistently along the sequence of tunnels, in order to make it appear as a single tunnel. Therefore, tunnel endpoint implementations SHOULD allow the operator to configure whether this optimisation is enabled.

When ECN support is added to a subnet technology, consideration SHOULD be given to a similar optimisation between subnets in sequence if they all use the same technology.

5.6. Reframing and Congestion Markings

The guidance in this section is worded in terms of framing boundaries, but it applies equally whether the protocol data units are frames, cells or packets.

Where framing boundaries are different between two layers, congestion indications SHOULD be propagated on the basis that a congestion indication on a PDU applies to all the octets in the PDU. On average, an encapsulator or decapsulator SHOULD approximately preserve the number of marked octets arriving and leaving (counting the size of inner headers, but not added encapsulating headers).

The next departing frame SHOULD be immediately marked even if only enough incoming marked octets have arrived for part of the departing frame. This ensures that any outstanding congestion marked octets are propagated immediately, rather than held back waiting for a frame no bigger than the outstanding marked octets--which might involve a long wait.

For instance, an algorithm for marking departing frames could maintain a counter representing the balance of arriving marked octets minus departing marked octets. It adds the size of every marked frame that arrives and if the counter is positive it marks the next frame to depart and subtracts its size from the counter. This will often leave a negative remainder in the counter, which is deliberate.

6. Feed-Up-and-Forward Mode: Guidelines for Adding Congestion Notification

The guidance in this section is applicable, for example, when IP packets:

- are encapsulated in Ethernet headers, which have no support for ECN;

- are forwarded by the eNode-B (base station) of a 3GPP radio access network, which is required to apply ECN marking during congestion, [LTE-RA], [UTRAN], but the Packet Data Convergence Protocol (PDCP) that encapsulates the IP header over the radio access has no support for ECN.

This guidance also generalises to encapsulation by other subnet technologies with no native support for explicit congestion notification at the lower layer, but with support for finding and processing an IP header. It is unlikely to be applicable or necessary for IP-in-IP encapsulation, where feed-forward-and-up mode based on [RFC6040] would be more appropriate.

Marking the IP header while switching at layer-2 (by using a layer-3 switch) or while forwarding in a radio access network seems to represent a layering violation. However, it can be considered as a benign optimisation if the guidelines below are followed. Feed-up-and-forward is certainly not a general alternative to implementing feed-forward congestion notification in the lower layer, because:

- IPv4 and IPv6 are not the only layer-3 protocols that might be encapsulated by lower layer protocols

- Link-layer encryption might be in use, making the layer-2 payload inaccessible

- Many Ethernet switches do not have ’layer-3 switch’ capabilities so they cannot read or modify an IP payload

- It might be costly to find an IP header (v4 or v6) when it may be encapsulated by more than one lower layer header, e.g. Ethernet MAC in MAC [IEEE802.1Qah].

Nonetheless, configuring lower layer equipment to look for an ECN field in an encapsulated IP header is a useful optimisation. If the implementation follows the guidelines below, this optimisation does not have to be confined to a controlled environment such as within a data centre; it could usefully be applied on any network--even if the operator is not sure whether the above issues will never apply:

1. If a native lower-layer congestion notification mechanism exists for a subnet technology, it is safe to mix feed-up-and-forward with feed-forward-and-up on other switches in the same subnet. However, it will generally be more efficient to use the native mechanism.

2. The depth of the search for an IP header SHOULD be limited. If an IP header is not found soon enough, or an unrecognised or unreadable header is encountered, the switch SHOULD resort to an alternative means of signalling congestion (e.g. drop, or the native lower layer mechanism if available).

3. It is sufficient to use the first IP header found in the stack; the egress of the relevant tunnel can propagate congestion notification upwards to any more deeply encapsulated IP headers later.

7. Feed-Backward Mode: Guidelines for Adding Congestion Notification

It can be seen from Section 4.3 that congestion notification in a subnet using feed-backward mode has generally not been designed to be directly coupled with IP layer congestion notification. The subnet attempts to minimise congestion internally, and if the incoming load at the ingress exceeds the capacity somewhere through the subnet, the layer 3 buffer into the ingress backs up. Thus, a feed-backward mode subnet is in some sense similar to a null mode subnet, in that there is no need for any direct interaction between the subnet and higher layer congestion notification. Therefore no detailed protocol design guidelines are appropriate. Nonetheless, a more general guideline is appropriate:

> A subnetwork technology intended to eventually interface to IP SHOULD NOT be designed using only the feed-backward mode, which is certainly best for a stand-alone subnet, but would need to be modified to work efficiently as part of the wider Internet, because IP uses feed-forward-and-up mode.

The feed-backward approach at least works beneath IP, where the term 'works' is used only in a narrow functional sense because feed-backward can result in very inefficient and sluggish congestion control--except if it is confined to the subnet directly connected to the original data source, when it is faster than feed-forward. It would be valid to design a protocol that could work in feed-backward mode for paths that only cross one subnet, and in feed-forward-and-up mode for paths that cross subnets.

In the early days of TCP/IP, a similar feed-backward approach was tried for explicit congestion signalling, using source-quench (SQ) ICMP control packets. However, SQ fell out of favour and is now formally deprecated [RFC6633]. The main problem was that it is hard for a data source to tell the difference between a spoofed SQ message and a quench request from a genuine buffer on the path. It is also hard for a lower layer buffer to address an SQ message to the

original source port number, which may be buried within many layers of headers, and possibly encrypted.

Quantised congestion notification (QCN--also known as backward congestion notification or BCN) [IEEE802.1Qau] uses a feed-backward mode structurally similar to ATM's relative rate mechanism. However, QCN confines its applicability to scenarios such as some data centres where all endpoints are directly attached by the same Ethernet technology. If a QCN subnet were later connected into a wider IP-based internetwork (e.g. when attempting to interconnect multiple data centres) it would suffer the inefficiency shown Figure 3.

8. IANA Considerations (to be removed by RFC Editor)

This memo includes no request to IANA.

9. Security Considerations

If a lower layer wire protocol is redesigned to include explicit congestion signalling in-band in the protocol header, care SHOULD be take to ensure that the field used is specified as mutable during transit. Otherwise interior nodes signalling congestion would invalidate any authentication protocol applied to the lower layer header--by altering a header field that had been assumed as immutable.

The redesign of protocols that encapsulate IP in order to propagate congestion signals between layers raises potential signal integrity concerns. Experimental or proposed approaches exist for assuring the end-to-end integrity of in-band congestion signals, e.g.:

- Congestion exposure (ConEx) for networks to audit that their congestion signals are not being suppressed by other networks or by receivers, and for networks to police that senders are responding sufficiently to the signals, irrespective of the transport protocol used [RFC7713].

- The ECN nonce [RFC3540] for a TCP sender to detect whether a network or the receiver is suppressing congestion signals.

- A test with the same goals as the ECN nonce, but without the need for the receiver to co-operate with the protocol [I-D.moncaster-tcpm-rcv-cheat].

Given these end-to-end approaches are already being specified, it would make little sense to attempt to design hop-by-hop congestion signal integrity into a new lower layer protocol, because end-to-end integrity inherently achieves hop-by-hop integrity.

10. Conclusions

Following the guidance in the document enables ECN support to be extended to numerous protocols that encapsulate IP (v4 & v6) in a consistent way, so that IP continues to fulfil its role as an end-to-end interoperability layer. This includes:

- A wide range of tunnelling protocols with various forms of shim header between two IP headers;
- A wide range of subnet technologies, particularly those that work in the same 'feed-forward-and-up' mode that is used to support ECN in IP and MPLS.

Guidelines have been defined for supporting propagation of ECN between Ethernet and IP on so-called Layer-3 Ethernet switches, using a 'feed-up-an-forward' mode. This approach could enable other subnet technologies to pass ECN signals into the IP layer, even if they do not support ECN natively.

Finally, attempting to add ECN to a subnet technology in feed-backward mode is deprecated except in special cases, due to its likely sluggish response to congestion.

11. Acknowledgements

Thanks to Gorry Fairhurst for extensive reviews. Thanks also to the following reviewers: Richard Scheffenegger, Ingemar Johansson, Piers O'Hanlon and Michael Welzl, who pointed out that lower layer congestion notification signals may have different semantics to those in IP. Thanks are also due to the tsvwg chairs, TSV ADs and IETF liaison people such as Eric Gray, Dan Romascanu and Gonzalo Camarillo for helping with the liaisons with the IEEE and 3GPP. And thanks to Georg Mayer and particularly to Erik Guttman for the extensive search and categorisation of any 3GPP specifications that cite ECN specifications.

Bob Briscoe was part-funded by the European Community under its Seventh Framework Programme through the Trilogy project (ICT-216372) for initial drafts and through the Reducing Internet Transport Latency (RITE) project (ICT-317700) subsequently. The views expressed here are solely those of the authors.

12. Comments Solicited

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group mailing list <tsvwg@ietf.org>, and/or to the authors.

13. References

13.1. Normative References

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

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <http://www.rfc-editor.org/info/rfc3819>.

[RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <http://www.rfc-editor.org/info/rfc4774>.

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <http://www.rfc-editor.org/info/rfc5129>.

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

13.2. Informative References

[ATM-TM-ABR] Cisco, "Understanding the Available Bit Rate (ABR) Service Category for ATM VCs", Design Technote 10415, June 2005.

[Buck00] Buckwalter, J., "Frame Relay: Technology and Practice", Pub. Addison Wesley ISBN-13: 978-0201485240, 2000.

[DCTCP] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", ACM SIGCOMM CCR 40(4)63--74, October 2010, <http://portal.acm.org/citation.cfm?id=1851192>.

[GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface", Technical Specification TS 29.060.

[GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.

[GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.

[I-D.briscoe-tsvwg-rfc6040bis]
Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", draft-briscoe-tsvwg-rfc6040bis-00 (work in progress), July 2016.

[I-D.eastlake-trill-ecn-support]
3rd, D. and B. Briscoe, "TRILL: ECN (Explicit Congestion Notification) Support", draft-eastlake-trill-ecn-support-00 (work in progress), March 2016.

[I-D.ietf-nvo3-geneve]
Gross, J. and I. Ganga, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-01 (work in progress), January 2016.

[I-D.ietf-nvo3-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-04 (work in progress), July 2016.

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

[I-D.moncaster-tcpm-rcv-cheat]
Moncaster, T., Briscoe, B., and A. Jacquet, "A TCP Test to Allow Senders to Identify Receiver Non-Compliance", draft-moncaster-tcpm-rcv-cheat-03 (work in progress), July 2014.

[IEEE802.1Qah]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks--Amendment 6: Provider Backbone Bridges", IEEE Std 802.1Qah-2008, August 2008, <http://www.ieee802.org/1/pages/802.1ah.html>.

(Access Controlled link within page)

[IEEE802.1Qau]
Finn, N., Ed., "IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks - Amendment 13: Congestion Notification", IEEE Std 802.1Qau-2010, March 2010, <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5454061>.

(Access Controlled link within page)

[ITU-T.I.371]
ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU-T Rec. I.371 (03/04), March 2004, <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5454061>.

[LTE-RA] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", Technical Specification TS 36.300.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, DOI 10.17487/RFC1323, May 1992, <http://www.rfc-editor.org/info/rfc1323>.

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <http://www.rfc-editor.org/info/rfc1701>.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <http://www.rfc-editor.org/info/rfc2003>.

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <http://www.rfc-editor.org/info/rfc2637>.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999, <http://www.rfc-editor.org/info/rfc2661>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <http://www.rfc-editor.org/info/rfc2884>.

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

[RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, DOI 10.17487/RFC3540, June 2003, <http://www.rfc-editor.org/info/rfc3540>.

[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>.

[RFC6633] Gont, F., "Deprecation of ICMP Source Quench Messages", RFC 6633, DOI 10.17487/RFC6633, May 2012, <http://www.rfc-editor.org/info/rfc6633>.

[RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <http://www.rfc-editor.org/info/rfc6660>.

[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>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <http://www.rfc-editor.org/info/rfc7713>.

[RFC7780] Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A., Ghanwani, A., and S. Gupta, "Transparent Interconnection of Lots of Links (TRILL): Clarifications, Corrections, and Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016, <http://www.rfc-editor.org/info/rfc7780>.

[UTRAN] 3GPP, "UTRAN Overall Description", Technical Specification TS 25.401.

Appendix A. Outstanding Document Issues

1. [GF] Concern that certain guidelines warrant a MUST (NOT) rather than a SHOULD (NOT). Given the guidelines say that if any SHOULD (NOT)s are not followed, a strong justification will be needed, they have been left as SHOULD (NOT) pending further list discussion. In particular:

   * If inner is a Not-ECN-PDU and Outer is CE (or highest severity congestion level), MUST (not SHOULD) drop?

     This issue has been addressed by explaining when SHOULD or MUST is appropriate.

2. Consider whether an IETF Standard Track doc will be needed to Update the IP-in-IP protocols listed in Section 5.1--at least those that the IETF controls--and which Area it should sit under.

   This issue has been addressed by the production of [I-D.briscoe-tsvwg-rfc6040bis], but this text is left outstanding until that draft is adopted.

Appendix B. Changes in This Version (to be removed by RFC Editor)

From ietf-05 to ietf-06:

* Introduction: Added GUE and Geneve as examples of tightly coupled shims between IP headers that cite RFC 6040. And added VXLAN to list of those that do not.

* Replaced normative text about tightly coupled shims between IP headers, with reference to new draft-briscoe-tsvwg-rfc6040bis

* Wire Protocol Design: Indication of ECN Support: Added TRILL as an example of a well-design protocol that does not need an indication of ECN support in the wire protocol.

* Encapsulation Guidelines: In the case of a Not-ECN-PDU with a CE outer, replaced SHOULD be dropped, with explanations of when SHOULD or MUST are appropriate.

* Feed-Up-and-Forward Mode: Explained examples more carefully, referred to PDCP and cited UTRAN spec as well as E-UTRAN.

* Added the people involved in liaisons to the acknowledgements.

* Updated references.

* Marked open issues as resolved, but did not delete Open Issues Appendix (yet).

From ietf-04 to ietf-05:

* Explained why tightly coupled shim headers only "SHOULD" comply with RFC 6040, not "MUST".

* Updated references

From ietf-03 to ietf-04:

* Addressed Richard Scheffenegger’s review comments: primarily editorial corrections, and addition of examples for clarity.

From ietf-02 to ietf-03:

* Updated references, ad cited RFC4774.

From ietf-01 to ietf-02:

* Added Section for guidelines that are applicable in all cases.

* Updated references.

From ietf-00 to ietf-01: Updated references.

From briscoe-04 to ietf-00: Changed filename following tsvwg adoption.

From briscoe-03 to 04:

* Re-arranged the introduction to describe the purpose of the document first before introducing ECN in more depth. And clarified the introduction throughout.

* Added applicability to 3GPP TS 36.300.

From briscoe-02 to 03:

* Scope section:

  + Added dependence on correct propagation of traffic class information

  + For the feed-backward mode, deemed multicast and anycast out of scope

* Ensured all guidelines referring to subnet technologies also refer to tunnels and vice versa by adding applicability sentences at the start of sections 4.1, 4.2, 4.3, 4.4, 4.6 and 5.

* Added Security Considerations on ensuring congestion signal fields are classed as immutable and on using end-to-end congestion signal integrity technologies rather than hop-by-hop.

From briscoe-01 to 02:

* Added authors: JK & PT

* Added

  + Section 4.1 "IP-in-IP Tunnels with Tightly Coupled Shim Headers"

  + Section 4.5 "Sequences of Similar Tunnels or Subnets"

  + roadmap at the start of Section 4, given the subsections have become quite fragmented.

  + Section 9 "Conclusions"

* Clarified why transports are starting to be able to saturate interior links

* Under Section 1.1, addressed the question of alternative signal semantics and included multicast & anycast.

* Under Section 3.1, included a 3GPP example.

* Section 4.2. "Wire Protocol Design":

  + Altered guideline 2. to make it clear that it only applies to the immediate subnet egress, not later ones

  + Added a reminder that it is only necessary to check that ECN propagates at the egress, not whether interior nodes mark ECN

  + Added example of how QCN uses 802.1p to indicate support for QCN.

* Added references to Appendix C of RFC6040, about monitoring the amount of congestion signals introduced within a tunnel

* Appendix A: Added more issues to be addressed, including plan to produce a standards track update to IP-in-IP tunnel protocols.

* Updated acks and references

From briscoe-00 to 01:

* Intended status: BCP (was Informational) & updates 3819 added.

* Briefer Introduction: Introductory para justifying benefits of ECN. Moved all but a brief enumeration of modes of operation to their own new section (from both Intro & Scope). Introduced incr. deployment as most tricky part.

* Tightened & added to terminology section

* Structured with Modes of Operation, then Guidelines section for each mode.

* Tightened up guideline text to remove vagueness / passive voice / ambiguity and highlight main guidelines as numbered items.

* Added Outstanding Document Issues Appendix

* Updated references

Authors' Addresses

Bob Briscoe
Simula Research Laboratory
UK

EMail: ietf@bobbriscoe.net
URI: http://bobbriscoe.net/

John Kaippallimalil
Huawei
5340 Legacy Drive, Suite 175
Plano, Texas 75024
USA

EMail: john.kaippallimalil@huawei.com

Pat Thaler
Broadcom Corporation
5025 Keane Drive
Carmichael, CA 95608
USA

EMail: pthaler@broadcom.com

INTAREA
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2017

S. Kanugovi
S. Vasudevan
Nokia
F. Baboescu
Broadcom

July 17, 2016

Multiple Access Management Protocol
draft-kanugovi-intarea-mams-protocol-00

Abstract

A communication network includes an access network element that delivers data to/from the user and an associated core network element that typically is the IP gateway having connectivity with the application servers. Multiconnectivity scenarios are common where a device can be simultaneously connected to multiple communication networks based on different technology implementations and network architectures like WiFi, LTE, DSL. A smart combination and selection of access and core network paths can improve quality of experience that a user in a multiconnectivity scenario. This document presents the problem statement and proposes solution principles. It specifies the requirements and reference architecture for a multi access management services framework that can be used to flexibly select the best combination of uplink and downlink access and core network paths, ensuring better network efficiency and enhanced application performance.

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 January 18, 2017.

Table of Contents

1. Conventions used in this document

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].

2. Terminology

"Client": The end-user device supporting connections with multiple access nodes, possibly over different access technologies.

"Multiconnectivity Client": A client with multiple network connections.

"Access network element": The functional element in the network that delivers user data packets to the client via a point-to-point access link like WiFi airlink, LTE airlink, DSL.

"Core": The functional element that anchors the client’s IP address used for communication with applications via the network.

"User Plane Gateway": The functional element that can intercept and route user data packets.

"Network Connection manager"(NCM): A functional entity in the network that oversees distribution of data packets over the multiple available access and core network paths.

"Client Connection Manager" (CCM): A functional entity in the client that exchanges MAMS Signaling with the Network Connection Manager and configures the multiple network paths for transport of user data.

"Anchor network element": The functional element in the network with connectivity via multiple access paths to the client.

"Multi Access Data Proxy" (MADP): This entity handles the user data traffic forwarding across multiple network paths.

3. Problem Statement

Typically, a device has access to multiple communication networks based on different technologies, say LTE, WiFi, DSL, MuLTEfire, for accessing application services. Different technologies exhibit benefits and limitations in different scenarios. For example, WiFi leverages the large spectrum available in unlicensed spectrum to deliver high capacities at low cost in uncongested scenarios with small user population, but can show significant degradation in application performance in congested scenarios with large user population. Another example is LTE network, the capacity of which is often constrained by high cost and limited availability of the licensed spectrum, but offers predictable service even in multi-user scenarios due to controlled scheduling and licensed spectrum usage.

Additionally, the use of a particular access network path is often coupled with the use of the associated core network. For example, in an enterprise that has deployed WiFi and LTE communications network, enterprise applications, like printers, Corporate Audio and Video conferencing, are accessible only via WiFi access connected to the enterprise hosted WiFi core, whereas the LTE access can be used to get LTE operator core anchored services including access to public internet.

Application performance in different scenarios, therefore becomes dependent on the choice of the communication networks due to the tight coupling of the access and the core network paths. Therefore to leverage the best possible application performance in the widest possible scenarios, a framework is needed that allows flexible selection of the combination of access and core network paths for application data delivery.

For example, in uncongested scenarios, it would be beneficial to use WiFi access in the uplink and downlink for connecting to enterprise applications. Whereas in congested scenarios, where use of WiFi in uplink by multiple users can lead to degraded capacity and increased delays due to contention, it would be beneficial to use scheduled LTE uplink access combined with WiFi downlink.

4. Solution Principles

This document proposes a Multiple Access Management Services(MAMS) framework for dynamic selection of uplink and downlink access and core network paths for a device connected to multiple communication networks. The selection of paths is based on negotiation of capabilities and network link quality between the device and a functional element in the network, namely the network connection manager. NCM has the intelligence to setup and offer the best network path based on device and network capabilities, application needs and knowledge of the network state.

5. Requirements

The requirements set out in this section are for the behavior of the MAMS mechanism and the related functional elements.

5.1. Access technology agnostic interworking

The access nodes can be of different technology types like LTE, WiFi etc. Since MAMS routes user plane data packets at the IP layer, which makes it agnostic to the type of underlying technology used at the access node for delivery of data to the client.

5.2. Support common transport deployments

The network path selection and user plane distribution should work transparently across transport deployments that include e2e IPsec, VPNs, and middleboxes like NATs and proxies.

5.3. Independent Access path selection for Uplink and Downlink

IP layer routing enables the client to transmit on uplink using one access and receive data on downlink using another access, allowing client and network connection manager to select the access paths for uplink and downlink independent of each other.

5.4. IP anchor selection independent of uplink and downlink access

A client is able to flexibly negotiate the IP anchor, core network, independent of the access paths used to reach the IP anchor depending on the application needs.

5.5. Adaptive network path selection

The network connection manager node has the ability to determine the quality of each of the network paths, e.g. access link delay and capacity. The network path quality information is fed into the logic for selection of combination of network paths to be used for transporting user data. The path selection algorithm can use network path quality information, in addition to other considerations like network policies, for optimizing network usage and enhancing QoE delivered to the user.

5.6. Multipath support and Aggregation of access link capacities

MAMS supports distribution and aggregation of user data across multiple network paths. MAMS allows the client to leverage the combined capacity of the multiple network connections by enabling simultaneous transport of user data over multiple network paths. If required, packet re-ordering is done at the receiver, client and/or the Anchor network element, when user data packets are received out of order. MAMS allows flexibility to choose the flow steering and aggregation protocol based on capabilities supported by the client and the Anchor network element.

5.7. Scalable mechanism based on IP interworking

The mechanism is based on IP interworking, requiring only the IP connectivity between the access nodes and the interworking functionality is based on generically available IP routing and

encapsulation capabilities. This makes solution easy to deploy and scale easily when different networks are added and removed.

5.8. Separate Control and Data plane functions

The client negotiates with a network connection manager the choice of access for both uplink and downlink accesses and the IP anchor(core). The network connection manager configures the actual user data distribution function residing in the Anchor element, thus maintaining a clear separation between the control and data plane functions. This makes the MAMS framework amenable to SDN based architecture and implementations.

6. MAMS Reference Architecture

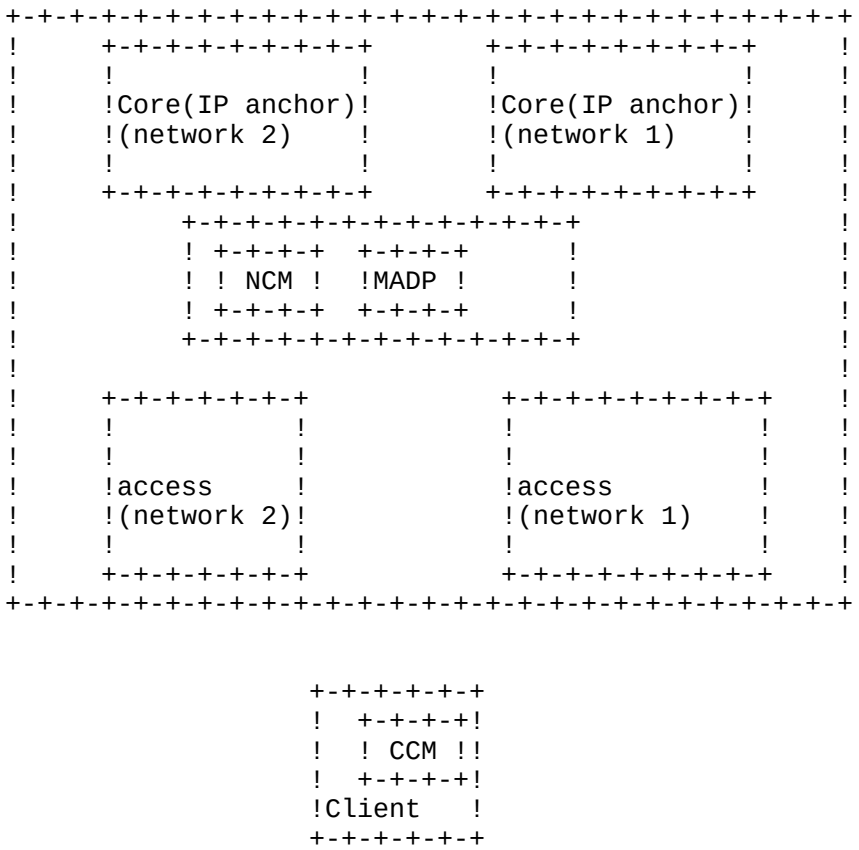


Figure 1: MAMS Reference Architecture

Figure 1 illustrates MAMS architecture for the scenario of a client served by 2 networks. The NCM and MADP, functional elements, are introduced for supporting MAMS mechanisms. The architecture is extendable to combine more than 2 networks, as well as any choice of participating network types (e.g. LTE, WLAN, MuLTEfire, DSL) and deployment architectures (e.g. with user plane gateway function at the access edge).

The MADP entity handles the user data traffic forwarding across multiple network paths. MADP is the distribution node for uplink and downlink data with visibility of packets at the IP layer. Identification and distribution rules for different user data traffic type at the MADP are configured by the NCM. The NCM configures the routing in the MADP based on signaling exchanged with the CCM in the client. In the UL, NCM specifies the core network path to be used by MADP to route uplink user data. In the DL, NCM specifies the access links to be used for delivery of data to the client.

The distribution algorithm at the MADP is configured by the NCM, based on static and/or dynamic network policies like assigning access and core paths for specific user data traffic type, data volume based percentage distribution, and link availability and feedback information from exchange of MAMS signaling with the CCM at the Client.

At the client, the Client Connection Manager (CCM) manages the multiple network connections. CCM is responsible for exchange of MAMS signaling messages with the NCM for supporting functions like configuring UL and DL user network path configuration for transporting user data packets, link sounding and reporting to support adaptive network path selection by NCM. In the downlink, for the user data received by the client, it configures IP layer forwarding for application data packet received over any of the accesses to reach the appropriate application module on the client. In the uplink, for the data transmitted by the client, it configures the routing table to determine the best access to be used for uplink data based on a combination of local policy and network policy delivered by the NCM.

A user plane tunnel, e.g. IPsec, may be needed for transporting user data packets between the MADP and the client. The user plane tunnel is needed to ensure security and routability of the user plane packets between the MADP and the client. The most common implementation of the user plane tunnel is the IPsec. In deployments where the access node belonging to the two networks are connected via a secure and direct IP path, user plane tunnel may not be needed.

7. MAMS call flow

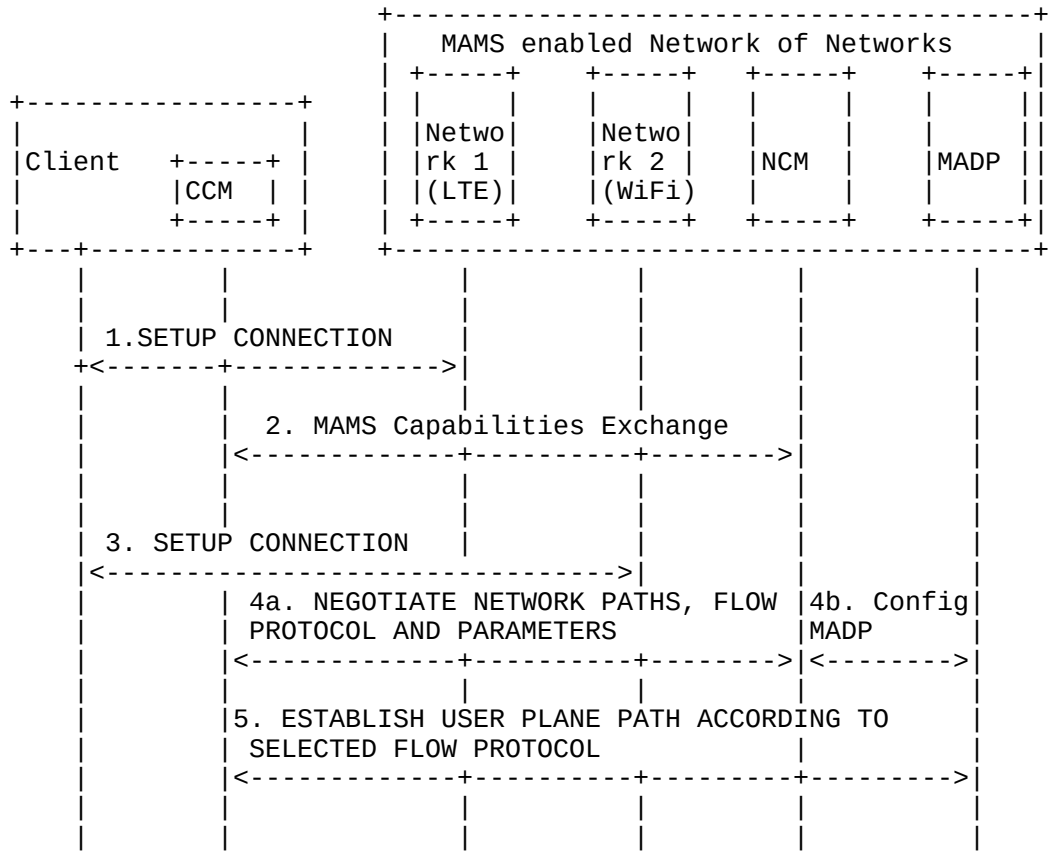


Figure 2: MAMS call flow

Figure 2 illustrates the MAMS signaling mechanism for negotiation of network paths and flow protocols between the client and the network. In this example scenario, the client is connected to two networks (say LTE and WiFi).

1. UE connects to network 1 and gets an IP address assigned by network 1.
2. CCM communicates with NCM functional element via the network 1 connection and exchanges capabilities and parameters for MAMS operation. Note: The NCM credentials can be made known to the UE by pre-provisioning.

3. Client sets up connection with network 2 and gets an IP address assigned by network 2.
4. CCM and MADP negotiate capabilites and parameters with NCM for establishment of network paths.

   4a. CCM and NCM negotiate network paths, flow routing and aggregation protocols, and related parameters.

   4b. NCM communicates with the MADP to exchange and configure flow aggregation and routing protocols, policies and parameters in alignment with those negotiation with the CCM.

5. CCM and MADP establish the user plane paths, e.g. using IKE [RFC7296] signaling, based on the negotiated flow protocol and parameters specified by NCM.

CCM and NCM can further exchange messages containing access link measurements for link maintenance by the NCM. NCM evaluates the link conditions in the UL and DL across LTE and WiFi, based on link measurements reported by CCM and/or link probing techniques and determines the UL and DL user data distribution policy. NCM configures MADP and CCM with these policies for controlling network paths over which the user data is transported. CCM may apply local policies, in addition to the network policy conveyed by the NCM.

8. Security Considerations

This section details the security considerations for the MAMS framework.

8.1. Data and Control plane security

Signaling messages and the user data in MAMS framework rely on the security of the underlying network transport paths. When this cannot be assumed, network connection manager configures use of protocols, like IPsec [RFC4301] [RFC3948], for securing user data and MAMS signaling messages.

9. Contributors

This protocol is the outcome of work by many engineers, not just the authors of this document. In alphabetical order, the contributors to the project are: Barbara Orlandi, Bongho Kim,David Lopez-Perez, Doru Calin, Jonathan Ling, Krishna Pramod A., Lohith Nayak, Michael Scharf.

10. References

10.1. Normative References

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

[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>.

10.2. Informative References

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <http://www.rfc-editor.org/info/rfc3948>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

Authors' Addresses

Satish Kanugovi
Nokia

Email: satish.k@nokia.com

Subramanian Vasudevan
Nokia

Email: vasu.vasudevan@nokia.com

Florin Baboescu
Broadcom

Email: florin.baboescu@broadcom.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2017

Y. Liu
A. Foldes
Ericsson
G. Zheng
Z. Wang
Y. Zhuang
Huawei
A. Wang
China Telecom
July 3, 2016

Yang Data Model for IPIPv4 Tunnel
draft-liu-intarea-ipipv4-tunnel-yang-02

Abstract

This document defines a YANG data model for the management of IP based tunnels. The data model includes configuration data and state data. In default format, it describes managed attributes used for managing tunnels of IPv4 or IPv6 over IPv4 tunnels. And it can also serve as a base model which can be augmented with technology-specific details in other Ip based tunnel models.

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 January 4, 2017.

Table of Contents

1. Introduction

An overview of tunnel is presented at [intarea-tunnel]. Over the past several years, there has been a number of "tunneling" protocols specified by the IETF. And this document defines a YANG data model for the management of IP bases tunnels. In default format, it covers the following tunnel types:

o IPv4 in IPv4, related concepts are defined in [RFC1853]

o IPv6 in IPv4 manual tunnel, related concepts are defined in [RFC2003]

o IPv6 to IPv4 tunnel, related concepts are defined in [RFC3056]

And notice that this model provides a framework and some reusable common attributes where technology-specific IP tunnel YANG models can inherit constructs from it without needing to redefine them within the sub-technology. Therefore it also can serve as a base model which can be extended to include technology specific details.

1.1. 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].

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

o Brackets "[" and "]" enclose list keys.

o Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).

o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Architecture of IP tunnel YANG Model

In this document we define an IP based tunnel model, in default it can describe the IPv6/4-in-IPv4 tunnels including IPv4 in IPv4 [RFC1853], IPv6 in IPv4 [RFC2003], IPv6 to IPv4 [RFC3056].

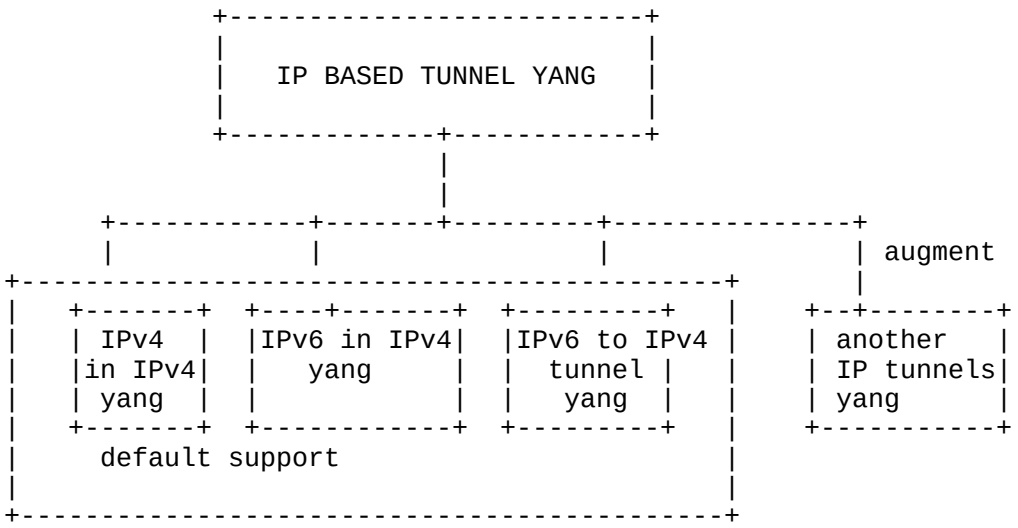
Since some reusable common core attributes of ip tunnels are defined, it can also provide an optional solution for extending to the other IP-based tunnel models. The users of IP-based tunnel model can according to following method to define other technologies specific ip tunnel models:

o The IP based Tunnel YANG model can acts as the root for other Tunnel YANG models.

o Augment the ip based tunnel model by adding the tunnel type is defined as an identity that augments the base " ip-tunnel-type " defined in the IP based model.

o Augment the ip based tunnel model by adding new data nodes with technology specific parameters into proper anchor points of the ip tunnel model.

Figure 1 depicts the relationship of different Tunnel YANG models to the Ip Tunnel YANG Model. It can default support the IPv4-in-IPv4, IPv6-in-IPv4, and IPv6-to-IPv4, and can also be extended to another IP base tunnels encapsulation protocol.



Relationship of technology specific TUNNEL YANG model to IP based tunnel YANG model

3. IP Tunnel Model Design

3.1. Model Overview

This document defines the YANG model "ietf-ip-tunnel". It includes two modules, one for configuration and one for state. At the top of the Model there is Tunnels container for tunnel configuration data. Multiple Tunnel lists keyed using tunnel name and tunnel type within the Tunnels container. To correctly identify an ip based tunnel the bind-interface, local-address, remote-address are defined under one tunnel list. Notice that tunnels are handled by creating a logical interface for each tunnel. Each logical interface (physical or virtual) need to map to interface yang model [RFC7223]. To do so, in this draft, the bind-interface are defined which with a leafref type and it can be used pointing to the corresponding interface-name node in the interface yang [RFC7223].

The data model also includes read-only counter so that the tunnel state can be read.

3.2. IP Tunnel YANG Tree Diagrams

The data model has the following tree diagram for the IP tunnels:

```
module: ietf-ip-tunnel
   +--rw Tunnels
   |  +--rw Tunnel* [name type]
   |     +--rw name                   string
   |     +--rw type                   identityref
   |     +--rw local-address?         inet:ip-address-no-zone
   |     +--rw remote-address?        inet:ip-address-no-zone
   |     +--rw routing-instance?      routing-instance-ref
   |     +--rw description?           string
   |     +--rw bind-interface?        if:interface-ref
   |     +--rw clear-df?              empty
   |     +--rw shutdown?              empty
   |     +--rw tmtu?                  uint16
   |     +--rw mirror-destination?    string
   |     +--rw hop-limit?             uint8
   |     +--rw tos?                   int8
   +--ro tunnel-state
      +--ro tunnels*
         +--ro name?                        string
         +--ro type?                        identityref
         +--ro local-ip?                    inet:ip-address-no-zone
         +--ro remote-ip?                   inet:ip-address-no-zone
         +--ro (state)?
         |  +--:(up)
         |  |  +--ro up?                          empty
         |  +--:(down)
         |  |  +--ro down?                        empty
         |  +--:(shutdown)
         |     +--ro shutdown?                    empty
         +--ro bind-interface?              if:interface-state-ref
         +--ro user-configured?             boolean
         +--ro routing-instance?            routing-instance-ref
         +--ro tmtu?                        uint16
         +--ro clear-df?                    empty
         +--ro down-reason?                 string
         +--ro resolved-interface-name?     string
         +--ro hop-limit?                   uint32
         +--ro tos?                         int32
augment /if:interfaces-state/if:interface:
   +--ro tunnel-protocol?   identityref
```

4. IP Tunnel YANG Data Model

```
<CODE BEGINS> file "ietf-ip-tunnel@2016-06-20.yang"
module ietf-ip-tunnel{

  namespace "urn:ietf:params:xml:ns:yang:ietf-ip-tunnel";
  prefix "iptnl";

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-routing {
    prefix "rt";
  }


  import iana-if-type {
    prefix ianaift;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group.";

  contact
    "Mandy.Liu@ericsson.com
     Adam.Foldes@ericsson.com
     zhengguangying@huawei.com";

  description
    "This YANG model defines the configuration data
     and operational state data for generic IPv4/6-in-IPv4 tunnel.
     It includes the IPv4 in IPv4, 6-to-4, and IPv6 over IPv4 manual
     tunnels.";

  revision 2016-04-27 {
    description
      "Made model more generic in order to allow augmentation by e.g.
       GRE tunnels.";
    reference
      "RFC XXXX: A YANG Data Model for IPv4 Tunnel.";
  }
  revision 2016-03-11 {
    description
```

```
      "Collapsed all tunnel types into a single subtree based on
       suggestions to more closely follow the IP Tunnel MIB.";
    reference
      "RFC XXXX: A YANG Data Model for IPv4 Tunnel.";
  }
  revision 2015-10-14 {
    description
      "Update model based on comments.";
    reference
      "RFC XXXX: A YANG Data Model for IPv4 Tunnel.";
  }

  revision 2015-07-20 {
    description
      "This version adds the following new items:
        - hop-limit
        - tos
        - tunnel-type
       This version changes 'ipv6to4-auto' to 'ipv6to4'";
    reference
      "RFC XXXX: A YANG Data Model for IPv4 Tunnel.";
  }

  revision 2015-05-27 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for IPv4 Tunnel.";
  }

  /* Identities */
  identity ip-tunnel-type {
    description
      "Base identity from which identities describing
       IP tunnel types are derived.";
  }
  identity ip-ip {
    base ip-tunnel-type;
    description
      "This identity represents IPv4-in-IPv4 tunnel type";
  }
  identity ipv6v4-manual {
    base ip-tunnel-type;
    description
      "This identity represents IPv6-to-IPv4 manual tunnel type";
  }
  identity ipv6-to-v4 {
    base ip-tunnel-type;
```

```
    description
      "This identity represents the 6-to-4 tunnel type";
  }

        typedef routing-instance-ref {
        type leafref {
          path "/rt:routing/rt:routing-protocols/rt:routing-protocol/rt:name";
        }
        description
        "This type is used for leafs that reference a routing instance
         configuration.";
      }

  /*Configuration Data*/
  container Tunnels{
    description
     "Configuration data for tunnels.";
    list Tunnel{
      key "name type";
      description
        "Configuration data for tunnels.";
      leaf name {
        type string;
        description
          "Name of the tunnel.";
      }
      leaf type {
        type identityref {
          base ip-tunnel-type;
        }
        description "The encapsulation type of the tunnel.";
      }

      leaf routing-instance {
        type routing-instance-ref;
        description "The routing instance of the local address.";
      }
      uses tunnel-config-components;
    }
  }

  /* Configuration data */
  grouping tunnel-config-components {
      description
        "Configuration data for all tunnels and subtunnels";
      leaf description {
        type string {
          length "1..255";
```

```
        }
        description
          "Textual description for a tunnel. Can be any "+
          "alphanumeric string, including spaces, not to exceed "+
          "255 ASCII characters.";
      }
      leaf bind-interface {
        type if:interface-ref;
        description
          "Bind to an interface.";
      }
          leaf local-address {
        type inet:ip-address-no-zone;
        description "IP address of the local end of the tunnel.";
      }
      leaf remote-address {
        when "type != ipv6-to-v4" {
          description
            "6-to-4 tunnels do not have a fixed remote endpoint.";
        }
        type inet:ip-address-no-zone;
        description "IP address of the remote end of the tunnel.";
      }

      leaf clear-df {
        type empty;
        description
          "If clear-df is absent, it means that fragmentation of
           tunnel packets are permitted. If clear-df is present,
           it means that fragmentation of tunnel packets are not
           permitted.";
      }
      leaf shutdown {
        type empty;
        description
          "Disable/enable the tunnel.";
      }
      leaf tmtu {
        type uint16 {
          range "256..16384";
        }
        description
          "Sets the Maximum Transmission Unit (MTU) size for
           packets sent in a tunnel. The default MTU is the MTU
           for the interface to which the tunnel is bound.";
      }
      leaf mirror-destination {
        type string;
```

```
        description
          "Designate the name of a tunnel as a circuit
           mirror destination. ";
      }
      leaf hop-limit {
        type uint8 {
          range "0|1..255";
        }
        description
          "The IPv4 TTL or IPv6 Hop Limit which is used in the
           outer IP header. A value of 0 indicates that the value
           is copied from the payload's header.";
      }
      leaf tos {
        type int8 {
          range "-1..63";
        }
        description
          "The method used to set the high 6 bits (the
            differentiated services codepoint) of the IPv4 TOS or
            IPv6 Traffic Class in the outer IP header. A value of -1
            indicates that the bits are copied from the payload\u2019s
            header. A value between 0 and 63 inclusive indicates
            that the bit field is set to the indicated value.";
      }

  }

  /*Operational state data*/
  grouping tunnel-oper-states {
    description "Operational states of tunnels";
    choice state {
      description "Choice of operational states";
      case up {
        leaf up {
          type empty;
          description "The tunnel is up.";
        }
      }
      case down {
        leaf down {
          type empty;
          description "The tunnel is down.";
        }
      }
      case shutdown {
        leaf shutdown {
          type empty;
```

```
          description "The tunnel is shut down administratively.";
        }
      }
    }
  }

  grouping tunnel-state-components {
    description
      "The basic tunnel information to be displayed.";

    leaf name {
      type string;
      description
        "Name of the tunnel.";
    }

    leaf type {
      type identityref;
      description
        "The type of the tunnel.";
    }
    leaf local-ip {
      type inet:ip-address-no-zone;
      description
        "IP address of the local end of the tunnel.";
    }
    leaf remote-ip {
      type inet:ip-address-no-zone;
      description
        "IP address of the remote end of the tunnel.";
    }
    uses tunnel-oper-states;
    leaf bind-interface {
      type if:interface-state-ref;
      description
        "Bind to an interface.";
    }
    leaf user-configured {
      type boolean;
          description
        "Indicate the tunnel is user-configured or dynamic.
        False is for dynamic.";
    }
    leaf routing-instance {
      type routing-instance-ref;
      description
        "Name of the reference routing instance. ";
    }
```

```
    leaf tmtu {
      type uint16;
      description
        "The Maximum Transmission Unit (MTU) size for
      packets sent in a tunnel.";
    }
    leaf clear-df {
      type empty;
      description
        "Indicate that the DF bit is cleared.";
    }
    leaf down-reason {
      type string;
      description
        "The reason of the tunnel is down.";
    }
    leaf resolved-interface-name{
      type string;
      description
        "The egress interface name of the tunnel.";
    }
    leaf hop-limit {
      type uint32;
      description
        "The IPv4 TTL or IPv6 Hop Limit which is used in the outer IP
         header. A value of 0 indicates that the calue is copied from
         the payload's header.";
    }
    leaf tos {
      type int32;
      description
        "The high 6 bits (the differentiated
         services codepoint) of the IPv4 TOS or IPv6 Traffic Class in
         the outer IP header. A value of -1 indicates that the bits
         are copied from the payload\u2019s header. A value between 0 and
         63 inclusive indicates that the bit field is set to the
         indicated value.";
    }
  }

  container tunnel-state {
    config "false";
      description
        "Contain the information currently configured tunnels.";
      list tunnels {
        description
           "Operational state data of tunnels.";
         uses tunnel-state-components;
```

```
      }
  }

  //Augment operational state data of IP interfaces
  augment "/if:interfaces-state/if:interface" {
    when "if:type = 'ianaift:tunnel'" {
      description
        "Augment IP interface.";
    }
    description
      "Augment operational state data of IP interfaces.";
    leaf tunnel-protocol {
      type identityref;
      description
        "Indicate the state of the IP tunnel interface.";
    }
  }
}
```

<CODE ENDS>

5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241] . The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242] . The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688] . Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-ip-tunnel

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-tunnel

namespace:urn:ietf:params:xml:ns:yang:ietf-ip-tunnel

prefix: itun reference: RFC XXXX

7. Normative References

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

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

Authors' Addresses

Ying Liu
Ericsson
No.5 Lize East Street
Beijing 100102
China

Email: Mandy.Liu@ericsson.com

Adam Mate Foldes
Ericsson
300 Holger Way
San Jose CA 95134
USA

Email: Adam.Foldes@ericsson.com

Guangying Zheng
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: zhengguangying@huawei.com

Zitao Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Yan Zhuang
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhuangyan.zhuang@huawei.com

Aijun Wang
China Telecom
No.118,Xizhimenneidajie,Xicheng District
Beijing 100035
China

Email: wangaj@ctbri.com.cn

Homenet Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

M. Cullen
Painless Security
M. Zhang
Huawei Technologies

July 8, 2016

Considerations for Bandwidth Aggregation
draft-mrc-banana-considerations-01

Abstract

This document lists a number of architectural and technical topics that should be considered in the design and implementation of Bandwidth Agregation mechanisms.

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 January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

There are currently several bandwidth aggregation solutions being discussed within the IETF or other parts of the Internet industry. This document discusses a number of technical and architectural facts that should be considered in the design and implementation of those solutions. This document is intended to provide useful information to the community, not to state requirements or advocate for a particular solution.

There is one simple thought underlying many of the considerations in this document: the goals of bandwidth aggregation are to increase the effective bandwidth available to customers and improve the reliability of customers' Internet access by using all of the available links, not just one of them. Intuitively, two links should have more bandwidth and reliability than one link, but experience shows that it is actually quite hard to design a bandwidth aggregation solution that will achieve the desired goals in all cases, and quite easy to design a solution that will reduce the effective bandwidth or decrease the reliability of Internet access in an unacceptably high number of cases. Many of the considerations in this document are intended to point out why that happens, so that solutions and implementations can avoid known pitfalls in this area.

[Note: This document is a work in progress. Feedback on the existing content is welcome, as well as feeback on other considerations that should be included. Please send any feedback to the Bandwidth Aggregation mailing list: banana@ietf.org]

2. What is Bandwidth Aggregation?

[TBD]

3. Taxonomy of Solutions

This section attempts to catergorize bandwidth aggregation solutions along several axes, providing a taxonomy tbat we can use to describe and reason about individual solutions. [Note: This section is largely TBD.]

3.1. Tunnel-Based Solutions

Many of the Bandwidth Aggregations currently under discussion are tunnel-based solutions. They tunnel traffic over the links that are being aggregated, and recombine the traffic on the remote end.

[Insert ASCII image of tunnel-based approach.]

There is at least one proposal for Bandwith Aggregation (the MP-TCP-based approach) that does not use tunnels. The considerations for tunnel-based solutions listed below may not apply to non-tunnel-based solutions.

3.2. Per-Packet vs. Per-Flow Multiplexing

The solutions currently under discussion use several different methods to determine which traffic will be sent over which interface.

These methods can be grouped into two categories: per-packet multiplexing and per-flow multiplexing.

Per-packet multiplexing aggregates the bandwidth by sending the desired proportion of packets over each interface. In these solutions, packets from single flow (such as a TCP connection) may be split across multiple interfaces and will need to be recombined at the remote end. However, the ability to multiplex on a per-packet basis makes it possible to most precisely apportion traffic across the available bandwidth.

Per-flow multiplexing involves choosing a single interface for each flow (i.e. TCP connection or application session) and sending all of the packets for a single flow across that interface. In these solutions, the flow do not need to be combined on the remote end. However, the ability to balance traffic between multiple links may be limited if there are only a small number of traffice flows active.

4. Considerations for All Solutions

This section desribes potential issues that should be considered in the design and implementation of all bandwidth aggregation solutions.

4.1. Link Characteristics and Performance

4.2. Bypass Traffic

4.3. Capped or Tariffed Interfaces

In some cases, bandwith aggregation may be performed between dedicated links and links that have traffic caps or tarifs associated with additional use. In these cases, customer may want to use bandwidth aggregation to increase the performance of some applications, while other applications (e.g. firmware upgrades or content downloads) may be limited to using the dedicated link. Solutions that wish to support this capability will need to support having a set of traffic that will be distributed using the bandwidth aggregation algorithms, and a set of traffic that will not.

4.4. Learning from History (Multilink PPP)

The IETF has a venerable, standard, implemented solution to this sort of problem: Multilink PPP. Unfortunately, it is commonly said that experience with Multilink PPP did not find that it increased the effective bandwidth when it was used to share two indentical ISDN lines, compared to the bandwidth that was achieved from using only one line...

[Note: We should attempt to determine if this is true and, if so, find any research papers or other documentation that might help us understand why this was true, so that we might learn from history.]

5. Considerations for Tunnel-Based Solutions

5.1. Tunnel Overhead

Tunneling involves more overhead than sending non-tunnelled traffic for two reasons: the extra IP and tunnel headers that must be included in each packet, and any tunnel management traffic that must be exchanged. This means that, in order to achieve increased effective bandwidth by aggregating traffic across more than one link, the raw bandwidth across multiple links must be higher than the bandwidth on a single link by a large enough margin to compensate for the tunnel overhead, so that increased effective bandwidth will result.

5.2. MTU Issues

There are a number of MTU Issues associated with all tunneling mechanisms, and there is a different set of MTU issues associated with any mechanism that changes the MTU of packets within a given flow.

[Note: This section is TBD.]

5.2.1. Fragmentation Issues

5.2.2. Issues with MTU Changes

6. Considerations for Per-Packet Solutions

6.1. Packet Ordering

6.2. Transport Layer Algorithms

There are transport layer congestion control algorithms implemented in every TCP/IP stack. It is the purpose of these algorithms to ramp up the speed of a TCP connection slowly, and to back off at the first sign of congestion (i.e. packet loss). There are also algorithms which are designed to detect packet loss as quickly as possible by analyzing the protocol round-trip times, and deciding that a packet has been lost whenever there is a longer delay than expected before an acknowledgement is received. Per-packet solutions run the risk of interacting pathologically with these algorithms.

For example, if traffic from a single flow is being demultiplexed across two links with significantly different round-trip times (i.e. differnt latencies), the TCP retransmission algorithms may be triggered for packets that traverse the higher latency link. This may cause the TCP congestion control algorithms to inaccurately detect congestion (even when neither link is congested) and slow down the speed of the TCP connetion. In these cases, the throughput of each TCP connection may be reduced, thus reducing the performance of a customer's applications to the point where their applicaitons would have run faster over a single link.

This problem can potentially be avoided by avoiding aggregation of links with significantly different latencies. However, it may be desirable to perform bandwidth aggregation across those links in some cases.

7. Considerations for Per-Flow Solutions

This section describes some potential issues that should be considered in the design of per-flow bandwidth aggregation solutions.

7.1. Granularity Issues

Per-Flow demultiplexing is in widespread use for traffic engineering and load balancing in carrier and corporate networks. Within those networks, there are a very large number of flows, so being able to direct traffic on a per-flow basis will generally be sufficient to achieve acceptable load-balancing or link aggregation.

However, the number of flows generated by a single home or small office might not provide sufficient granularity to achieve the desire level of bandwidth aggregation. Also some flows, such as streaming video flows, might use far more bandwidth than other, such as downloading a single image on a web page. It is not always possible to predict which flows will be high-bandwidth flows, and which will require less bandwidth.

7.2. Aggregated Flows

Some Internet flows are aggregated into single, larger flows at the end-nodes. This would include VPN traffic that is tunnelled to a corporate intranet, or other tunnelled traffic such as Teredo traffic for IPv6. Use of these mechanisms can prevent proper classification of traffic into separate flows, thus exacerbating the granularity issues described above.

7.3. Encrypted Traffic

In some cases such as secure VPN traffic, the contents of packets may be encrypted in a way that does not allow a middlebox to see the transport-layer flow inforamtion (such as TCP or UDP ports). In these cases, it might not be possible to properly separate multiple flows between a single set of endpoints. This can exacerbate the granularity issues described above.

8. Practical Considerations

8.1. Use Available Information

In many of the environments in which these mechanisms will be deployed, there is already considerable informaiton available about link quality, lost packets, traffic loads and effective bandwidth. It is possible to use that information to actively tune a bandwidth aggregation solution to achieve optimal effective bandwidth. This information can also be used to detect situations in which the link quality of a secondary link is not sufficient to provide enough additional bandwidth to compensate for the bandwidth aggregation overhead.

8.2. Theory is No Substitute for Experience

Because of the complexity of the algorithms implemented at multiple layers of the TCP/IP Stack, many things that would appear to work in theory or in limited simulation do not have the expected results when deployed in a real-world environment. Therefore, it would be highly desirable to have real-world experience running a bandwidth aggregation mechanism in an operational network before standardizing it within the IETF.

9. Security Considerations

9.1. Binding Tunnel Endpoints

10. Appendix A: List of Solutions

This is a (possibly incomplete) list of current or proposed solutions for Bandwidth Aggregation. The descriptions in this section (when present) were provided by the proponents of each solution. This list is provided only as a source of information about possible solutions, not as a recommendation for or against any of these solutions.

[Note: Insert information from Google Doc in this section.]

11. Informative References

[RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, DOI 10.17487/RFC6126, April 2011, <http://www.rfc-editor.org/info/rfc6126>.

Authors' Addresses

Margaret Cullen
Painless Security
14 Summer Street, Suite 202
Malden, MA 02148
USA

Phone: +1 781 405-7464
Email: margaret@painless-security.com
URI: http://www.painless-security.com

Mingui Zhang
Huawei Technologies

Email: zhangmingui@huawei.com

INTAREA
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2017

E. Nordmark
Arista Networks

Jul 2016

IP over Intentionally Partially Partitioned Links
draft-nordmark-intarea-ippl-04

Abstract

IP makes certain assumptions about the L2 forwarding behavior of a multi-access IP link. However, there are several forms of intentional partitioning of links ranging from split-horizon to Private VLANs that violate some of those assumptions. This document specifies that link behavior and how IP handles links with those properties.

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 January 2, 2017.

described in the Simplified BSD License.

Table of Contents

1. Introduction

IPv4 and IPv6 can in general handle two forms of links; point-to-point links when only have two IP nodes (self and remote), and multi-access links with one or more nodes attached to the link. For the multi-access links IP in general, and particular protocols like ARP and IPv6 Neighbor Discovery, makes a few assumptions about transitive and reflexive connectivity i.e., that all nodes attached to the link can send packets to all other nodes.

There are cases where for various reasons and deployments one wants what looks like one link from the perspective of IP and routing, yet the L2 connectivity is restrictive. A key property is that an IP subnet prefix is assigned to the link, and IP routing sees it as a regular multi-access link. But a host attached to the link might not be able to send packets to all other hosts attached to the link. The motivation for this is outside the scope of this document, but in summary the motivation to preserve the subnet view as seen by IP routing is to conserve IP(v4) address space, and the motivation to restrict communication on the link could be due to (security) policy or potentially wireless connectivity approaches.

This intentional and partial partition appears in a few different forms. For DSL [TR-101] and Cable [DOCSIS-MULPI] the pattern is to have a single access router on the link, and all the hosts can send and receive from the access router, but host-to-host communication is blocked. A richer set of restrictions are possible for Private VLANs (PVLAN) [RFC5517], which has a notion of three different ports i.e. attachment points: isolated, community, and promiscuous. Note that other techniques operate at L2/L3 boundary like [RFC4562] but those are out of scope for this document.

The possible connectivity patterns for PVLAN appears to be a superset of the DSL and Cable use of split horizon, thus this document specifies the PVLAN behavior, shows the impact on IP/ARP/ND, and specifies how IP/ARP/ND must operate to work with PVLAN.

If private VLANs, or the split horizon subset, has been configured at layer 2 for the purposes of IPv4 address conservation, then that layer 2 configuration will affect IPv6 even though IPv6 might not have the same need for address conservation.

2. Keywords and Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

The following terms from [RFC4861] are used without modifications:

node	a device that implements IP.
router	a node that forwards IP packets not explicitly addressed to itself.
host	any node that is not a router.
link	a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IP. Examples are Ethernets (simple or bridged), PPP links, X.25, Frame Relay, or ATM networks as well as Internet-layer (or higher-layer) "tunnels", such as tunnels over IPv4 or IPv6 itself.
interface	a node's attachment to a link.
neighbors	nodes attached to the same link.

This document defines the following set of terms:

bridge	a layer-2 device which implements 802.1Q
port	a bridge's attachment to another bridge or to a node.

3. Private VLAN

A private VLAN is a structure which uses two or more 802.1Q (VLAN) values to separate what would otherwise be a single VLAN, viewed by IP as a single broadcast domain, into different types of ports with different L2 forwarding behavior between the different ports. A private VLAN consists of a single primary VLAN and multiple secondary VLANs.

From the perspective of both a single bridge and a collection of interconnected bridges there are three different types of ports use to attach nodes plus an inter-bridge port:

- Promiscuous: A promiscuous port can send packets to all ports that are part of the private VLAN. Such packets are sent using the primary VLAN ID.
- Isolated: Isolated VLAN ports can only send packets to promiscuous ports. Such packets are sent using an isolated VLAN ID.
- Community: A community port is associated with a per-community VLAN ID, and can send packets to both ports in the same community VLAN and promiscuous ports.
- Inter-bridge: A port used to connect a bridge to another bridge.

3.1. Bridge Behavior

Once a bridge or a set of interconnected bridges have been configured with both the primary and isolated VLAN ID, and zero or more community VLAN IDs associated with the private VLAN, the following forward behaviors apply to the bridge:

- o A packet received on an isolated port MUST NOT be forwarded out an isolated or community port; it SHOULD (subject to bandwidth/ resource issues) be forwarded out promiscuous and inter-bridge ports.
- o A packet received on a community port MUST NOT be forwarded out an isolated port or a community port with a different VLAN ID; it SHOULD be forwarded out promiscuous and inter-bridge ports as well as community ports that have the same community VLAN ID.
- o A packet received on a promiscuous port SHOULD be forwarded out all types of ports in the private VLAN.
- o A packet received on an inter-bridge port with an isolated VLAN ID should be forwarded as a packet received on an isolated port.
- o A packet received on an inter-bridge port with a community VLAN ID should be forwarded as a packet received on a community port associated with that VLAN ID.
- o A packet received on an inter-bridge port with a promiscuous VLAN ID should be forwarded as a packet received on a promiscuous port.

In addition to the above VLAN filtering and implied MAC address learning rules, the packet forwarding is also subject to the normal 802.1Q rules with blocking ports due to spanning-tree protocol etc.

4. IP over IPPL

When IP is used over Intentionally Partially Partitioned links like private VLANs the normal usage is to attached routers (and potentially other shared resources like servers) to promiscuous ports, while attaching other hosts to either community or isolated ports. If there is a single host for a given tenant or other domain of separation, then it is most efficient to attach that host to an isolated port. If there are multiple hosts in the private VLAN that should be able to communicate at layer 2, then they should be assigned a common community VLAN ID and attached to ports with that VLAN ID.

The above configuration means that hosts will not be able to communicate with each other unless they are in the same community. However, mechanisms outside of the scope of this document can be used to allow IP communication between such hosts e.g., by having firewall or gateway in or beyond the routers connected to the promiscuous ports. When such a policy is in place it is important that all packets which cross communities are sent to a router, which can have access-control lists or deeper firewall rules to decide which packets to forward.

5. IPv6 over IPPL

IPv6 Neighbor Discovery [RFC4861] can be used to get all the hosts on the link to send all unicast packets except those send to link-local destination addresses to the routers. That is done by setting the L-flag (on-link) to zero for all of the Prefix Information options. Note that this is orthogonal to whether SLAAC (Stateless Address Auto-Configuration) [RFC4862] or DHCPv6 [RFC3315] is used for address autoconfiguration. Setting the L-flag to zero is RECOMMENDED configuration for private VLANs.

If the policy includes allowing some packets that are sent to link-local destinations to cross between different tenants, then some for of NS/NA proxy is needed in the routers, and the routers need to forward packets addressed to link-local destinations out the same interface as REQUIRED in [RFC2460]. If the policy allows for some packets sent to global IPv6 address to cross between tenants then the routers would forward such packets out the same interface. However, with the L=0 setting those global packets will be sent to the default router, while the link-local destinations would result in a Neighbor Solicitation to resolve the IPv6 to link-layer address binding. Handling such a NS when there are multiple promiscuous ports hence multiple routers risks creating loops. If the router already has a neighbor cache entry for the destination it can respond with an NA on behalf of the destination. However, if it does not it MUST NOT send a NS on the link, since the NA will be received by the other router(s) on the link which can cause an unbounded flood of multicast NS packets (all with hoplimit 255), in particular of the host IPv6 address does not respond. Note that such an NS/NA proxy is defined in [RFC4389] under some topological assumptions such as there being a distinct upstream and downstream direction, which is not the case of two or more peer routers on the same IPPL. For that reason NS/NA packet proxies as in [RFC4389] MUST NOT be used with IPPL.

IPv6 includes Duplicate Address Detection [RFC4862], which assumes that a link-local IPv6 multicast can be received by all hosts which share the same subnet prefix. That is not the case in a private VLAN, hence there could potentially be undetected duplicate IPv6 addresses. However, the DAD proxy approach [RFC6957] defined for split-horizon behavior can safely be used even when there are multiple promiscuous ports hence multiple routers attached to the link, since it does not rely on sending Neighbor Solicitations instead merely gathers state from received packets. The use of [RFC6957] with private VLAN is RECOMMENDED.

The Router Advertisements in a private VLAN MUST be sent out on a promiscuous VLAN ID so that all nodes on the link receive them.

6. IPv4 over IPPL

IPv4 [RFC0791] and ARP [RFC0826] do not have a counterpart to the Neighbor Discovery On-link flag. Hence nodes attached to isolated or community ports will always ARP for any destination which is part of its configured subnet prefix, and those ARP request packets will not be forwarded by the bridges to the target nodes. Thus the routers attached to the promiscuous ports MUST provide a robust proxy ARP mechanism if they are to allow any (firewalled) communication between nodes from different tenants or separation domains.

For the ARP proxy to be robust it MUST avoid loops where router1 attached to the link sends an ARP request which is received by router2 (also attached to the link), resulting in an ARP request from router2 to be received by router1. Likewise, it MUST avoids a similar loop involving IP packets, where the reception of an IP packet results in sending a ARP request from router1 which is proxied by router2. At a minimum, the reception of an ARP request MUST NOT result in sending an ARP request, and the routers MUST either be configured to know each others MAC addresses, or receive the VLAN tagged packets so they can avoid proxying when the packet is received on with the promiscuous VLAN ID. Note that should there be an IP forwarding loop due to proxying back and forth, the IP TTL will expire avoiding unlimited loops.

Any proxy ARP approach MUST work correctly with Address Conflict Detection [RFC5227]. ACD depends on ARP probes only receiving responses if there is a duplicate IP address, thus the ARP probes MUST NOT be proxied. These ARP probes have a Sender Protocol Address of zero, hence they are easy to identify.

When proxying an ARP request (with a non-zero Sender Protocol Address) the router needs to respond by placing its own MAC address in the Sender Hardware Address field. When there are multiple routers attached to the private VLAN this will not only result in multiple ARP replies for each ARP request, those replies would have a different Sender Hardware Address. That might seem surprising to the requesting node, but does not cause an issue with ARP implementations that follow the pseudo-code in [RFC0826].

If the two or more routers attached to the private VLAN implement VRRP [RFC5798] the routers MAY use their VRRP MAC address as the Sender Hardware Address in the proxied ARP replies, since this reduces the risk nodes that do not follow the pseudo-code in [RFC0826]. However, if they do so it can cause flapping of the MAC tables in the bridges between the routers and the ARPing node. Thus such use is NOT RECOMMENDED in general topologies of bridges but can be used when there are no intervening bridges.

7. Multiple routers

In addition to the above issues when multiple routers are attached to the same PVLAN, the routers need to avoid potential routing loops for packets entering the subnet. When such a packet arrives the router might need to send a ARP request (or Neighbor Solicitation) for the host, which can trigger the other router to send a proxy ARP (or Neighbor Advertisement). The host, if present, will also respond to the ARP/NS. This issue is described in [PVLAN-HOSTING] in the particular case of HSRP.

When multiple routers are attached to the same PVLAN, wheter they are using VRRP, HSRP, or neither, they SHOULD NOT proxy ARP/ND respond to a request from another router. At a minimum a router MUST be configurable with a list of IP addresses to which it should not proxy respond. Thus the user can configure that list with the IP address(es) of the other router(s) attached to the PVLAN.

8. Multicast over IPPL

Layer 2 multicast or broadcast is used by protocols like ARP [RFC0826], IPv6 Neighbor Discovery [RFC4861] and Multicast DNS [RFC6762] with link-local scope. The first two have been discussed above.

Multicast DNS can be handled by implementing using some proxy such as [I-D.ietf-dnssd-hybrid] but that is outside of the scope of this document.

IP Multicast which spans across multiple IP links and that have senders that are on community or isolated ports require additional forwarding mechanisms in the routers that are attached to the promiscuous ports, since the routers need to forward such packets out to any allowed receivers in the private VLAN without resulting in packet duplication. For multicast senders on isolated ports such forwarding would result in the sender potentially receiving the packet it transmitted. For multicast senders on community ports, any receivers in the same community VLAN are subject to receiving duplicate packets; one copy directly from layer 2 from the sender and a second copy forwarded by the multicast router.

For that reason it is NOT RECOMMENDED to configure outbound multicast forwarding from private VLANs.

9. Security Considerations

In general DAD is subject to a Denial of Service attack since a malicious host can claim all the IPv6 addresses [RFC3756]. Same issue applies to IPv4/ARP when Address Conflict Detection [RFC5227] is implemented.

10. IANA Considerations

There are no IANA actions needed for this document.

11. Acknowledgements

The author is grateful for the comments from Mikael Abrahamsson, Fred Baker, Wes Beebee, Hemant Singh, Dave Thaler, and Sowmini Varadhan.

12. References

12.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <http://www.rfc-editor.org/info/rfc826>.

[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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless

Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC6957] Costa, F., Combes, J-M., Ed., Pougnard, X., and H. Li, "Duplicate Address Detection Proxy", RFC 6957, DOI 10.17487/RFC6957, June 2013, <http://www.rfc-editor.org/info/rfc6957>.

12.2. Informative References

[DOCSIS-MULPI] "DOCSIS 3.0: MAC and Upper Layer Protocols Interface Specification", August 2015, <http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.0-I28-150827.pdf>.

[I-D.ietf-dnssd-hybrid] Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-03 (work in progress), February 2016.

[PVLAN-HOSTING] "PVLANs in a Hosting Environment", March 2010, <https://puck.nether.net/pipermail/cisco-nsp/2010-March/068469.html>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, DOI 10.17487/RFC3756, May 2004, <http://www.rfc-editor.org/info/rfc3756>.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <http://www.rfc-editor.org/info/rfc4389>.

[RFC4562] Melsen, T. and S. Blake, "MAC-Forced Forwarding: A Method for Subscriber Separation on an Ethernet Access Network", RFC 4562, DOI 10.17487/RFC4562, June 2006, <http://www.rfc-editor.org/info/rfc4562>.

[RFC5227] Cheshire, S., "IPv4 Address Conflict Detection", RFC 5227, DOI 10.17487/RFC5227, July 2008,

<http://www.rfc-editor.org/info/rfc5227>.

[RFC5517] HomChaudhuri, S. and M. Foschiano, "Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment", RFC 5517, DOI 10.17487/RFC5517, February 2010, <http://www.rfc-editor.org/info/rfc5517>.

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <http://www.rfc-editor.org/info/rfc5798>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[TR-101] "Migration to Ethernet-Based DSL Aggregation", The Broadband Forum Technical Report TR-101, July 2011, <http://www.broadband-forum.org/technical/download/TR-101_Issue-2.pdf>.

Author's Address

Erik Nordmark
Arista Networks
Santa Clara, CA
USA

Email: nordmark@arista.com

Internet Engineering Task Force R. Winter
Internet-Draft M. Faath
Intended status: Informational F. Weisshaar
Expires: January 9, 2017 University of Applied Sciences Augsburg
July 8, 2016

Privacy considerations for IP broadcast and multicast protocol designers
draft-winfaa-intarea-broadcast-consider-02

Abstract

A number of application-layer protocols make use of IP broadcasts or multicast messages for functions such as local service discovery or name resolution. Some of these functions can only be implemented efficiently using such mechanisms. When using broadcasts or multicast messages, a passive observer in the same broadcast domain can trivially record these messages and analyze their content. Therefore, broadcast/multicast protocol designers need to take special care when designing their protocols.

Table of Contents

1. Introduction

Broadcast and multicast messages have a large receiver group by design. Because of that, these two mechanisms are vital for a number of basic network functions such as auto-configuration. Application developers use broadcast/multicast messages to implement things like local service or peer discovery and it appears that an increasing number of applications make use of it [TRAC2016].

Using broadcast/multicast can become problematic if the information that is being distributed can be regarded as sensitive or when the information that is distributed by multiple of these protocols can be correlated in a way that sensitive data can be derived. This is clearly true for any protocol, but broadcast/multicast is special in at least two respects: a) the aforementioned large receiver group which makes it trivial for anybody on a LAN to collect the information without special privileges or a special location in the network and b) encryption is more difficult when broadcasting/ multicasting messages.

Privacy considerations of IETF-specified protocols have received some attention in the recent past(e.g. [RFC7721] or [I-D.ietf-dhc-dhcp-privacy]). This draft documents a number of privacy considerations for broadcast/multicast protocol designers that are intended to reduce the likelihood that a broadcast protocol can be misused to collect sensitive data about devices, users and

groups of users on a LAN. These considerations particularly apply to protocols designed outside the IETF for two reasons. For one, non-standard protocols will likely not receive operational attention and support in making them more secure such as e.g. DHCP snooping does for DHCP because they typically are not documented. The other reason is that these protocols have been designed in isolation, where a set of considerations to follow is useful in the absence of a larger community providing feedback. In particular, carelessly designed broadcast/multicast protocols can break privacy efforts at different layers of the protocol stack such as MAC address or IP address randomization [RFC4941].

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 RFC 2119 [RFC2119].

2. Privacy considerations

There are a few obvious and a few not necessarily obvious things designers of broadcast/multicast protocols should consider in respect to the privacy implications of their protocol. Most of these items are based on protocol behaviour observed as part of experiments on operational networks [TRAC2016].

2.1. Message frequency

Frequent broadcast/multicast traffic caused by an application can give user behaviour and online times away. This allows a passive observer to potentially deduce a user’s current activity (e.g. a game) and it allows to create an online profile (i.e. times the user is on the network). The higher the frequency of these messages, the more accurate this profile will be. Given that broadcasts are only visible in the same broadcast domain, these messages also give the rough location of the user away (e.g. a campus or building).

Besides the privacy implications of frequent broadcasting, it also represents a performance problem. In particular in certain wireless technologies such as 802.11, broadcast and multicast are transmitted at a much lower rate (the lowest common denominator rate) compared to unicast and therefore have a much bigger impact on the overall available airtime. Further, it will limit the ability for devices to go to sleep if frequent broadcasts are being sent. A similar problem in respect to Router Advertisements is addressed in [I-D.ietf-v6ops-reducing-ra-energy-consumption].

If a protocol relies on frequent or periodic broadcast/multicast messages, the frequency SHOULD be chosen conservatively, in particular if the messages contain persistent identifiers (see next subsection). Also, intelligent message suppression mechanisms such as the ones employed in mDNS [RFC6762] SHOULD be implemented.

2.2. Persistent identifiers

A few broadcast/multicast protocols observed in the wild make use of persistent identifiers. This includes the use of hostnames or more abstract persistent identifiers such as a UUID or similar. These IDs, which e.g. identify the installation of a certain application might not change across updates of the software and are therefore extremely long lived. This allows a passive observer to track a user precisely if broadcast/multicast messages are frequent. This is even true in case the IP and/or MAC address changes. Such identifiers also allow two different interfaces (e.g. Wifi and Ethernet) to be correlated to the same device. If the application makes use of persistent identifiers for multiple installations of the same application for the same user, this even allows to infer that different devices belong to the same user.

If a broadcast/multicast protocol relies on IDs to be transmitted, it SHOULD be considered if frequent ID rotations are possible in order to make user tracking more difficult. Persistent IDs are considered bad practice in general as persistent application layer IDs will make efforts on lower layers to randomize identifiers (e.g. [I-D.huitema-6man-random-addresses]) useless or at least much more difficult.

2.3. Anticipate user behaviour

A large number of users name their device after themselves, either using their first name, last name or both. Often a hostname includes the type, model or maker of a device, its function or includes language specific information. Based on gathered data, this appears currently to be prevalent user behaviour [TRAC2016]. For protocols using the hostname as part of the messages, this clearly will reveal personally identifiable information to everyone on the local network.

Where possible, the use of hostnames in broadcast/multicast protocols SHOULD be avoided. If only a persistent ID is needed, this can be generated. An application might want to display the information it will broadcast on the LAN at install/config time, so the user is at least aware of the application’s behaviour. More hostname considerations can be found in [I-D.ietf-intarea-hostname-practice].

2.4. Consider potential correlation

A large number of services and applications make use of the broadcast/multicast mechanism. That means there are various sources of information that are easily accessible by a passive observer. In isolation, the information these protocols reveal might seem harmless, but given multiple such protocols, it might be possible to correlate this information. E.g. a protocol that uses frequent messages including a UUID to identify the particular installation does not give the identity of the user away. But a single message including the user's hostname might just do that and it can be correlated using e.g. the MAC address of the device's interface.

A broadcast protocol designer should be aware of the fact that even if - in isolation - the information a protocol leaks seems harmless, there might be ways to correlate that information with other broadcast protocol information to reveal sensitive information about a user.

2.5. Configurability

A lot of applications and services using broadcast protocols do not include the means to declare "safe" environments (e.g. based on the SSID of a WiFi network). E.g. a device connected to a public WiFi will likely broadcast the same information as when connected to the home network. It would be beneficial if certain behaviour could be restricted to "safe" environments.

An application developer making use of broadcasts as part of the application SHOULD make the broadcast feature, if possible, configurable, so that potentially sensitive information does not leak on public networks.

3. Operational considerations

Besides changing end-user behavior, choosing sensible defaults as an operating system vendor (e.g. for suggesting host names) and the considerations for protocol designers mentioned in this document, there are things that the network administrators/operators can do to limit the above mentioned problems.

A feature not uncommonly found on access points e.g. is to filter broadcast and multicast traffic. This will potentially break certain applications or some of their functionality but will also protect the users from potentially leaking sensitive information.

4. Acknowledgements

This work was partly supported by the European Commission under grant agreement FP7-318627 mPlane. Support does not imply endorsement.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This document deals with privacy-related considerations of broadcast- and multicast-based protocols. It contains advice for designers of such protocols to minimize the leakage of privacy-sensitive information. The intent of the advice is to make sure that identities will remain anonymous and user tracking will be made difficult.

7. Informative References

[I-D.huitema-6man-random-addresses]
Huitema, C., "Implications of Randomized Link Layers Addresses for IPv6 Address Assignment", draft-huitema-6man-random-addresses-03 (work in progress), March 2016.

[I-D.ietf-dhc-dhcp-privacy]
Krishnan, S., Mrugalski, T., and S. Jiang, "Privacy considerations for DHCP", draft-ietf-dhc-dhcp-privacy-05 (work in progress), February 2016.

[I-D.ietf-intarea-hostname-practice]
Huitema, C. and D. Thaler, "Current Hostname Practice Considered Harmful", draft-ietf-intarea-hostname-practice-00 (work in progress), October 2015.

[I-D.ietf-v6ops-reducing-ra-energy-consumption]
Yourtchenko, A. and L. Colitti, "Reducing energy consumption of Router Advertisements", draft-ietf-v6ops-reducing-ra-energy-consumption-03 (work in progress), November 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <http://www.rfc-editor.org/info/rfc7721>.

[TRAC2016] Faath, M., Weisshaar, F., and R. Winter, "How Broadcast Data Reveals Your Identity and Social Graph", 7th International Workshop on TRaffic Analysis and Characterization IEEE TRAC 2016, September 2016.

Authors' Addresses

Rolf Winter
University of Applied Sciences Augsburg
Augsburg
DE

Email: rolf.winter@hs-augsburg.de

Michael Faath
University of Applied Sciences Augsburg
Augsburg
DE

Email: michael.faath@hs-augsburg.de

Fabian Weisshaar
University of Applied Sciences Augsburg
Augsburg
DE

Email: fabian.weisshaar@hs-augsburg.de