Routing Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 2, 2017

Anil Kumar S N
Gaurav Agrawal
Vinod Kumar S
Huawei Technologies
C. Bowers
Juniper Networks

August 29, 2016

Maximally Redundant Trees in Segment Routing
draft-agv-rtgwg-spring-segment-routing-mrt-03

Abstract

[RFC7812] defines an architecture for IP/LDP Fast Reroute using Maximally Redundant Trees (MRT-FRR). This document extends the use of MRT to provide link and node protection for networks that use segment routing. This document makes use of the inherent support in segment routing for associating segments with different algorithms for computing next hops to reach prefixes. It assigns new Segment Routing Algorithms values corresponding to the MRT-Red and MRT-Blue next-hop computations defined in [RFC7811].

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 March 2, 2017.

Table of Contents

1. Introduction

MRT is a fast re-route technology that provides 100% coverage for link and nodes failures. This document describes how to use MRT as a FRR technology in a segment routing network.

2. Requirements Language

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

3. Terminology

Redundant Trees (RT): A pair of trees where the path from any node X to the root R along the first tree is node-disjoint with the path from the same node X to the root along the second tree. These can be computed in 2-connected graphs.

Maximally Redundant Trees (MRT): A pair of trees where the path from any node X to the root R along the first tree and the path from the same node X to the root along the second tree share the minimum number of nodes and the minimum number of links. Each such shared node is a cut-vertex. Any shared links are cut-links. Any RT is an MRT but many MRTs are not RTs. The two MRTs are referred to as MRT-Blue and MRT-Red.

MRT-Red: MRT-Red is used to describe one of the two MRTs; it is used to described the associated forwarding topology and MT-ID. Specifically, MRT-Red is the decreasing MRT where links in the GADAG are taken in the direction from a higher topologically ordered node to a lower one.

MRT-Blue: MRT-Blue is used to describe one of the two MRTs; it is used to described the associated forwarding topology and MT-ID. Specifically, MRT-Blue is the increasing MRT where links in the GADAG are taken in the direction from a lower topologically ordered node to a higher one.

MRT Island: From the computing router, the set of routers that support a particular MRT profile and are connected via MRT-eligible links.

Island Border Router (IBR): A router in the MRT Island that is connected to a router not in the MRT Island and both routers are in a common area or level.

Island Neighbor (IN): A router that is not in the MRT Island but is adjacent to an IBR and in the same area/level as the IBR.

4. MRT for Segment Routing Network

4.1. Overview

Segment Routing (SR) allows a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF). Prefix segments represent an ECMP-aware shortest-path to a prefix (or a node), as per the state of the IGP topology. Adjacency segments represent a hop over a specific adjacency between two nodes in the IGP.

MRT Fast Reroute requires that packets to be forwarded not only on the shortest-path tree, but also on two Maximally Redundant Trees (MRTs), referred to as the MRT-Blue and the MRT-Red. A router that experiences a local failure must also have predetermined which alternate to use. The MRT algorithm is defined in [RFC7811]. The

Default MRT Profile path calculation uses Lowpoint algorithm to calculate Maximally Redundant Trees.

To use MRT in Segment Routing network below mentioned capabilities needs to be incorporated in SR node.

1. SR nodes MUST support IGP extensions for MRT.

2. SR nodes MUST support MRT-RED and MRT-BLUE Algorithms.

3. SR nodes MUST advertise it’s MRT capability.

4. SR nodes SHOULD send and receive SID/Label for MRT topologies (MRT-RED and MRT-BLUE) for SR segment(s).

5. SR nodes MUST support computation of MRTs.

4.2. IGP extensions for MRT

These extensions are to support the distributed computation of Maximally Redundant Trees (MRT). These extensions indicate the MRT profile(s) each router supports. Different MRT profiles can be defined to support different uses and to allow transition of capabilities.

To support MRT for SR, a new SR MRT Profile is defined (as defined in section 5 of this document). IGP extensions for MRT[I-D.ietf-isis-mrt] / [I-D.ietf-ospf-mrt]MUST carry SR MRT profile.

4.3. SR Algorithm value for MRT-Blue and MRT-Red

Segment routing supports the use of different algorithms for computing paths to reach nodes and prefixes. To accomplish this, every Prefix-SID has a mandatory algorithm field. This Prefix-SID identifies an instruction to forward a packet along the path computed using the algorithm identified in the algorithm field.

[I-D.ietf-spring-segment-routing] defines two algorithms, Shortest Path and Strict Shortest Path.
[I-D.ietf-isis-segment-routing-extensions] and
[I-D.ietf-ospf-segment-routing-extensions] each assign the algorithm values of 0 and 1 to identify these two algorithms.

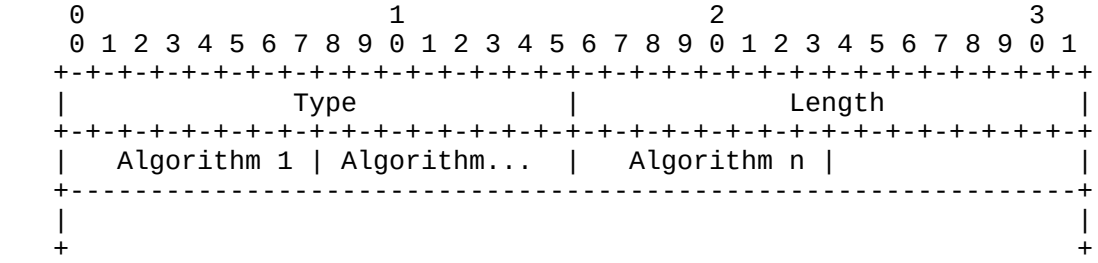
This document defines two additional algorithm values to support MRT-FRR using Segment Routing. The two algorithms correspond to the MRT-Red and MRT-Blue for the Default MRT Profile.

4.4. MRT capability advertisement for SR

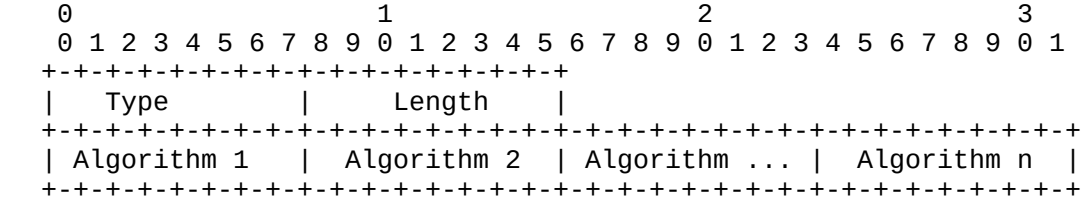
As packets routed on a hop-by-hop basis require that each router compute a shortest-path tree that is consistent, it is necessary for each router to compute the MRT-Blue next hops and MRT-Red next hops in a consistent fashion. For this each SR node needs to know which of the SR nodes in the SR domain supports MRT. This MUST be communicated using SR MRT Capability Advertisement.

Both OSPF [I-D.ietf-ospf-segment-routing-extensions] and ISIS [I-D.ietf-isis-segment-routing-extensions] supports segment routing capabilities advertisement. MRT algorithm capabilities need to be advertised in SR-Algorithm TLV for OSPF extension for segment routing and SR-Algorithm Sub-TLV for ISIS extension for segment routing.

SR-Algorithm TLV [I-D.ietf-ospf-segment-routing-extensions]



SR-Algorithm Sub-TLV[I-D.ietf-isis-segment-routing-extensions]



SR node is considered as MRT capable node if it advertises MRT capability in IGP extension of MRT as well as IGP extension of SR.

4.5. SR MRT SID/Label advertisement

In a link-state IGP domain, an SR-capable IGP node advertises segments for its attached prefixes and adjacencies. These segments are called IGP segments or IGP SIDs. They play a key role in Segment Routing as they enable the expression of any topological path throughout the IGP domain. Such a topological path is either expressed as a single IGP segment or a list of multiple IGP segments.

SR nodes supporting MRT MUST advertise two additional labels corresponding to MRT-BLUE and MRT-RED for each prefix segment.

These labels are carried as a part of prefix SID sub-tlv (OSPF Section 5 of [I-D.ietf-ospf-segment-routing-extensions], ISIS Section 2.1 of [I-D.ietf-isis-segment-routing-extensions]) with algorithm field set to algorithm value corresponding to the MRT forwarding topology as described in section Section 4.3.

4.6. MRT computation for SR

An SR node that advertise support for the Segment Routing MRT Profile MUST support MRT-RED and MRT-BLUE forwarding topology creation and support the computation of FRR paths as per the MRT algorithm described in [RFC7811].

5. MRT-FRR for destination-based and traffic-engineered SR

In addition to the Prefix-SIDs for Shortest Path algorithm, the IGP distributes Prefix-SIDs for MRT-Red and MRT-Blue. This allows each node to install transit forwarding entries for the MRT-Red and MRT-Blue paths for those prefixes. In normal operation, the traffic for a given destination will be forwarded based on the Shortest Path algorithm Prefix-SID for that destination. Upon detecting a link failure, a node can act as a point-of-local repair (PLR) by forwarding the traffic using either the MRT-Red or MRT-Blue Prefix-SID for the same destination. Following the appropriate MRT path, the traffic will the destination without crossing the failed link or the remote node attached to the failed link, if such a path exists.

The description above is independent of the segment routing forwarding plane. With the MRT-Red and MRT-Blue Segment Routing Algorithm values defined in this document, MRT-FRR can provide protection for traffic using either the MPLS or the IPv6 forwarding plane for segment routing. For clarity, we also describe the MRT-FRR mechanism when realized using the MPLS forwarding plane for segment routing.

In the MPLS-specific description, each node uses the index information contained in Prefix-SIDs and the SRGBs advertised by its neighbors to install transit forwarding entries for the Shortest Path, MRT-Red path, and MRT-Blue path for each destination prefix. As an example, the transit forwarding entry for a destination prefix on the MRT-Red path is a label swap operation where the both the incoming and outgoing labels correspond to the MRT-Red labels on the MRT-Red path.

In the absence of failures, traffic flows on transit forwarding entries corresponding to the Shortest Path algorithm where an incoming Shortest Path label is swapped to an outgoing Shortest Path label for a given destination. Upon the failure of a link, the PLR may change some forwarding entries to swap an incoming Shortest Path label to an outgoing MRT-Red or MRT-Blue label.

MRT Prefix Segments are applicable to both destination-based SR as well as traffic-engineered SR. In the case of destination-based SR, the Segment List consists of a single Prefix Segment with an MRT-Red or MRT-Blue algorithm value. In this case Prefix Segment identifies the complete MRT-Red or MRT-Blue path to the destination node or prefix. In the case of traffic-engineered SR, a Prefix Segment with an MRT algorithm value may form part of a larger Segment List. In this case, the MRT Prefix Segment identifies a portion of the entire end-to-end path in the SR domain. That portion of the path corresponds to the MRT-Red or MRT-Blue path for that prefix.

6. SR MRT Profile

The following set of options defines the SR MRT Profile. The SR MRT Profile is indicated by the MRT Profile ID.

MRT Algorithm: MRT Lowpoint algorithm defined in [RFC7811].

MRT-Red SR Algorithm ID: The MRT-Red SR Algorithm ID is indicated by the MRT-Red SR Algorithm ID.

MRT-Blue SR Algorithm ID: The MRT-Blue SR Algorithm ID is indicated by the MRT-BLue SR Algorithm ID.

GADAG Root Selection Policy: Among the routers in the MRT Island with the lowest numerical value advertised for GADAG Root Selection Priority, an implementation MUST pick the router with the highest Router ID to be the GADAG root. Note that a lower numerical value for GADAG Root Selection Priority indicates a higher preference for selection.

Forwarding Mechanisms: MRT SR Label Option 1A

Recalculation: Recalculation of MRTs SHOULD occur as described in Section 12.2 of [RFC7812] with SR label.

Area/Level Border Behavior: As described in Section 10 of [RFC7812], ABRs/LBRs SHOULD ensure that traffic leaving the area also exits the MRT-Red or MRT-Blue forwarding topology.

7. IANA Considerations

IANA is requested to allocate a value from the MRT Profile Identifier Registry for the Segment Routing MRT Profile with a value of TDB-1.

Value	Description	Reference
TBD-1	Segment Routing MRT Profile	This document

Currently, there is no IANA registry defined for SR Algorithm values carried in the Prefix-SID sub-TLV and the SR Algorithm sub-TLV for IS-IS or the Prefix-SID sub-TLV and SR Algorithm TLV for OSPF [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-segment-routing-extensions] define the Segment Routing algorithms for values 0 and 1.

This document requests IANA to create a registry entitled "Segment Routing Algorithm Values". This should appear in the IANA registry under a new top-level entry entitled "IGP-Independent Segment Routing Parameters". The initial registry is shown below.

Value	SR Algorithm	References
0	Shortest Path	I-D.ietf-isis-segment-routing-extensions I-D.ietf-ospf-segment-routing-extensions
1	Strict Shortest Path	I-D.ietf-isis-segment-routing-extensions I-D.ietf-ospf-segment-routing-extensions
TBD-2	MRT-Red	This document
TBD-3	MRT-Blue	This document

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

Security consideration of MRT and SR are applicable here. None of the additional security consideration are identified.

9. Acknowledgements

None

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>.

[RFC7811] Enyedi, G., Csaszar, A., Atlas, A., Bowers, C., and A. Gopalan, "An Algorithm for Computing IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7811, DOI 10.17487/RFC7811, June 2016, <http://www.rfc-editor.org/info/rfc7811>.

[RFC7812] Atlas, A., Bowers, C., and G. Enyedi, "An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7812, DOI 10.17487/RFC7812, June 2016, <http://www.rfc-editor.org/info/rfc7812>.

10.2. Informative References

[I-D.ietf-isis-mrt]
Li, Z., Wu, N., <>, Q., Atlas, A., Bowers, C., and J. Tantsura, "Intermediate System to Intermediate System (IS-IS) Extensions for Maximally Redundant Trees (MRT)", draft-ietf-isis-mrt-02 (work in progress), May 2016.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-07 (work in progress), June 2016.

[I-D.ietf-ospf-mrt]
Atlas, A., Hegde, S., Bowers, C., Tantsura, J., and Z. Li, "OSPF Extensions to Support Maximally Redundant Trees", draft-ietf-ospf-mrt-02 (work in progress), May 2016.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-09 (work in progress), July 2016.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

Authors' Addresses

Anil Kumar S N
Huawei Technologies
Near EPIP Industrial Area, Kundalahalli Village
Whitefield, Bangalore, Karnataka 560066
India

Email: anil.ietf@gmail.com

Gaurav Agrawal
Huawei Technologies
Near EPIP Industrial Area, Kundalahalli Village
Whitefield, Bangalore, Karnataka 560066
India

Email: gaurav.agrawal@huawei.com

Vinod Kumar S
Huawei Technologies
Near EPIP Industrial Area, Kundalahalli Village
Whitefield, Bangalore, Karnataka 560066
India

Email: vinods.kumar@huawei.com

Chris Bowers
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: cbowers@juniper.net

Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

F. Baker
Cisco Systems
C. Bowers
Juniper Networks
J. Linkova
Google

October 31, 2016

Enterprise Multihoming using Provider-Assigned Addresses without Network Prefix Translation: Requirements and Solution
draft-bowbakova-rtgwg-enterprise-pa-multihoming-01

Abstract

Connecting an enterprise site to multiple ISPs using provider-assigned addresses is difficult without the use of some form of Network Address Translation (NAT). Much has been written on this topic over the last 10 to 15 years, but it still remains a problem without a clearly defined or widely implemented solution. Any multihoming solution without NAT requires hosts at the site to have addresses from each ISP and to select the egress ISP by selecting a source address for outgoing packets. It also requires routers at the site to take into account those source addresses when forwarding packets out towards the ISPs.

This document attempts to define a complete solution to this problem. It covers the behavior of routers to forward traffic taking into account source address, and it covers the behavior of host to select appropriate source addresses. It also covers any possible role that routers might play in providing information to hosts to help them select appropriate source addresses. In the process of exploring potential solutions, this documents also makes explicit requirements for how the solution would be expected to behave from the perspective of an enterprise site network administrator .

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 May 4, 2017.

Table of Contents

1. Introduction

Site multihoming, the connection of a subscriber network to multiple upstream networks using redundant uplinks, is a common enterprise architecture for improving the reliability of its Internet connectivity. If the site uses provider-independent (PI) addresses, all traffic originating from the enterprise can use source addresses from the PI address space. Site multihoming with PI addresses is commonly used with both IPv4 and IPv6, and does not present any new technical challenges.

It may be desirable for an enterprise site to connect to multiple ISPs using provider-assigned (PA) addresses, instead of PI addresses. Multihoming with provider-assigned addresses is typically less expensive for the enterprise relative to using provider-independent

addresses. PA multihoming is also a practice that should be facilitated and encouraged because it does not add to the size of the Internet routing table, whereas PI multihoming does. Note that PA is also used to mean "provider-aggregatable". In this document we assume that provider-assigned addresses are always provider-aggregatable.

With PA multihoming, for each ISP connection, the site is assigned a prefix from within an address block allocated to that ISP by its National or Regional Internet Registry. In the simple case of two ISPs (ISP-A and ISP-B), the site will have two different prefixes assigned to it (prefix-A and prefix-B). This arrangement is problematic. First, packets with the "wrong" source address may be dropped by one of the ISPs. In order to limit denial of service attacks using spoofed source addresses, BCP38 [RFC2827] recommends that ISPs filter traffic from customer sites to only allow traffic with a source address that has been assigned by that ISP. So a packet sent from a multihomed site on the uplink to ISP-B with a source address in prefix-A may be dropped by ISP-B.

However, even if ISP-B does not implement BCP38 or ISP-B adds prefix-A to its list of allowed source addresses on the uplink from the multihomed site, two-way communication may still fail. If the packet with source address in prefix-A was sent to ISP-B because the uplink to ISP-A failed, then if ISP-B does not drop the packet and the packet reaches its destination somewhere on the Internet, the return packet will be sent back with a destination address in prefix-A. The return packet will be routed over the Internet to ISP-A, but it will not be delivered to the multihomed site because its link with ISP-A has failed. Two-way communication would require some arrangement for ISP-B to advertise prefix-A when the uplink to ISP-A fails.

Note that the same may be true with a provider that does not implement BCP 38, if his upstream provider does, or has no corresponding route. The issue is not that the immediate provider implements ingress filtering; it is that someone upstream does, or lacks a route.

With IPv4, this problem is commonly solved by using [RFC1918] private address space within the multi-homed site and Network Address Translation (NAT) or Network Address/Port Translation (NAPT) on the uplinks to the ISPs. However, one of the goals of IPv6 is to eliminate the need for and the use of NAT or NAPT. Therefore, requiring the use of NAT or NAPT for an enterprise site to multihome with provider-assigned addresses is not an attractive solution.

[RFC6296] describes a translation solution specifically tailored to meet the requirements of multi-homing with provider-assigned IPv6 addresses. With the IPv6-to-IPv6 Network Prefix Translation (NPTv6) solution, within the site an enterprise can use Unique Local Addresses [RFC4193] or the prefix assigned by one of the ISPs. As traffic leaves the site on an uplink to an ISP, the source address gets translated to an address within the prefix assigned by the ISP on that uplink in a predictable and reversible manner. [RFC6296] is currently classified as Experimental, and it has been implemented by several vendors. See Section 5.2, for more discussion of NPTv6.

This document defines routing requirements for enterprise multihoming using provider-assigned IPv6 addresses. We have made no attempt to write these requirements in a manner that is agnostic to potential solutions. Instead, this document focuses on the following general class of solutions.

Each host at the enterprise has multiple addresses, at least one from each ISP-assigned prefix. Each host, as discussed in Section 4.1 and [RFC6724], is responsible for choosing the source address applied to each packet it sends. A host SHOULD be able respond dynamically to the failure of an uplink to a given ISP by no longer sending packets with the source address corresponding to that ISP. Potential mechanisms for the communication of changes in the network to the host are Neighbor Discovery Router Advertisements, DHCPv6, and ICMPv6.

The routers in the enterprise network are responsible for ensuring that packets are delivered to the "correct" ISP uplink based on source address. This requires that at least some routers in the site network are able to take into account the source address of a packet when deciding how to route it. That is, some routers must be capable of some form of Source Address Dependent Routing (SADR), if only as described in [RFC3704]. At a minimum, the routers connected to the ISP uplinks (the site exit routers or SERs) must be capable of Source Address Dependent Routing. Expanding the connected domain of routers capable of SADR from the site exit routers deeper into the site network will generally result in more efficient routing of traffic with external destinations.

The document first looks in more detail at the enterprise networking environments in which this solution is expected to operate. It then discusses existing and proposed mechanisms for hosts to select the source address applied to packets. Finally, it looks at the requirements for routing that are needed to support these enterprise network scenarios and the mechanisms by which hosts are expected to select source addresses dynamically based on network state.

2. Enterprise Multihoming Requirements

2.1. Simple ISP Connectivity with Connected SERs

We start by looking at a scenario in which a site has connections to two ISPs, as shown in Figure 1. The site is assigned the prefix 2001:db8:0:a000::/52 by ISP-A and prefix 2001:db8:0:b000::/52 by ISP-B. We consider three hosts in the site. H31 and H32 are on a LAN that has been assigned subnets 2001:db8:0:a010::/64 and 2001:db8:0:b010::/64. H31 has been assigned the addresses 2001:db8:0:a010::31 and 2001:db8:0:b010::31. H32 has been assigned 2001:db8:0:a010::32 and 2001:db8:0:b010::32. H41 is on a different subnet that has been assigned 2001:db8:0:a020::/64 and 2001:db8:0:b020::/64.

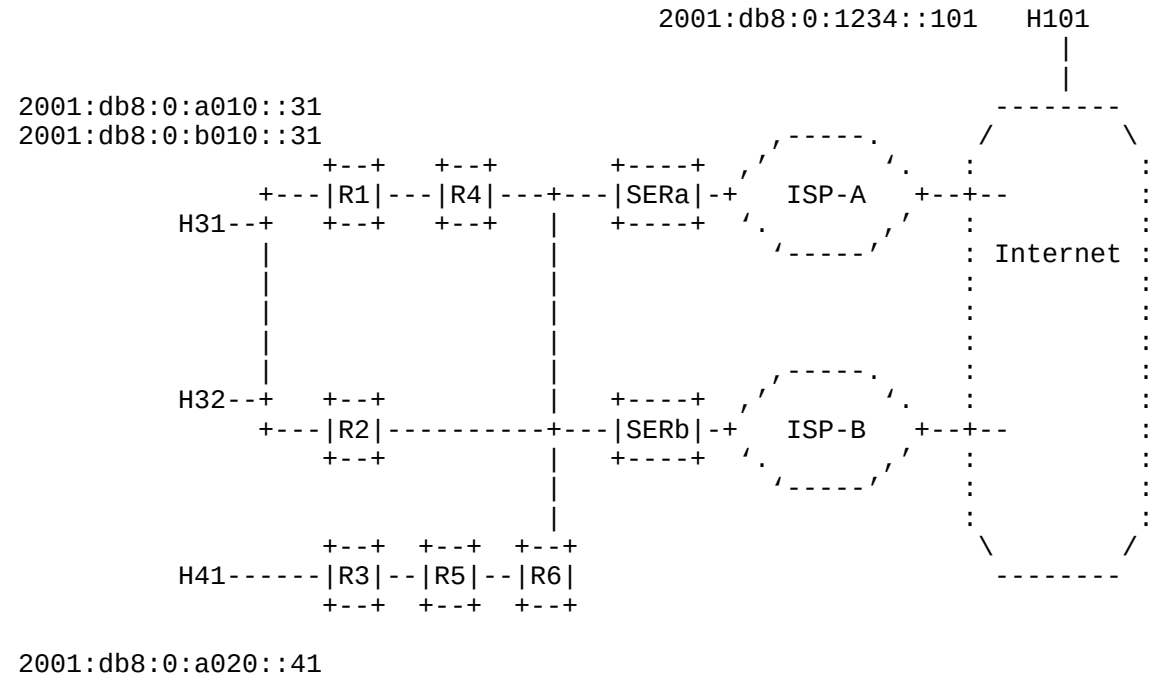


Figure 1: Simple ISP Connectivity With Connected SERs

We refer to a router that connects the site to an ISP as a site edge router(SER). Several other routers provide connectivity among the internal hosts (H31, H32, and H41), as well as connecting the internal hosts to the Internet through SERa and SERb. In this

example SERa and SERb share a direct connection to each other. In Section 2.2, we consider a scenario where this is not the case.

For the moment, we assume that the hosts are able to make good choices about which source addresses through some mechanism that doesn't involve the routers in the site network. Here, we focus on primary task of the routed site network, which is to get packets efficiently to their destinations, while sending a packet to the ISP that assigned the prefix that matches the source address of the packet. In Section 4, we examine what role the routed network may play in helping hosts make good choices about source addresses for packets.

With this solution, routers will need form of Source Address Dependent Routing, which will be new functionality. It would be useful if an enterprise site does not need to upgrade all routers to support the new SADR functionality in order to support PA multi-homing. We consider if this is possible and what are the tradeoffs of not having all routers in the site support SADR functionality.

In the topology in Figure 1, it is possible to support PA multihoming with only SERa and SERb being capable of SADR. The other routers can continue to forward based only on destination address, and exchange routes that only consider destination address. In this scenario, SERa and SERb communicate source-scoped routing information across their shared connection. When SERa receives a packet with a source address matching prefix 2001:db8:0:b000::/52 , it forwards the packet to SERb, which forwards it on the uplink to ISP-B. The analogous behaviour holds for traffic that SERb receives with a source address matching prefix 2001:db8:0:a000::/52.

In Figure 1, when only SERa and SERb are capable of source address dependent routing, PA multi-homing will work. However, the paths over which the packets are sent will generally not be the shortest paths. The forwarding paths will generally be more efficient as more routers are capable of SADR. For example, if R4, R2, and R6 are upgraded to support SADR, then can exchange source-scoped routes with SERa and SERb. They will then know to send traffic with a source address matching prefix 2001:db8:0:b000::/52 directly to SERb, without sending it to SERa first.

2.2. Simple ISP Connectivity Where SERs Are Not Directly Connected

In Figure 2, we modify the topology slightly by inserting R7, so that SERa and SERb are no longer directly connected. With this topology, it is not enough to just enable SADR routing on SERa and SERb to support PA multi-homing. There are two solutions to ways to enable PA multihoming in this topology.

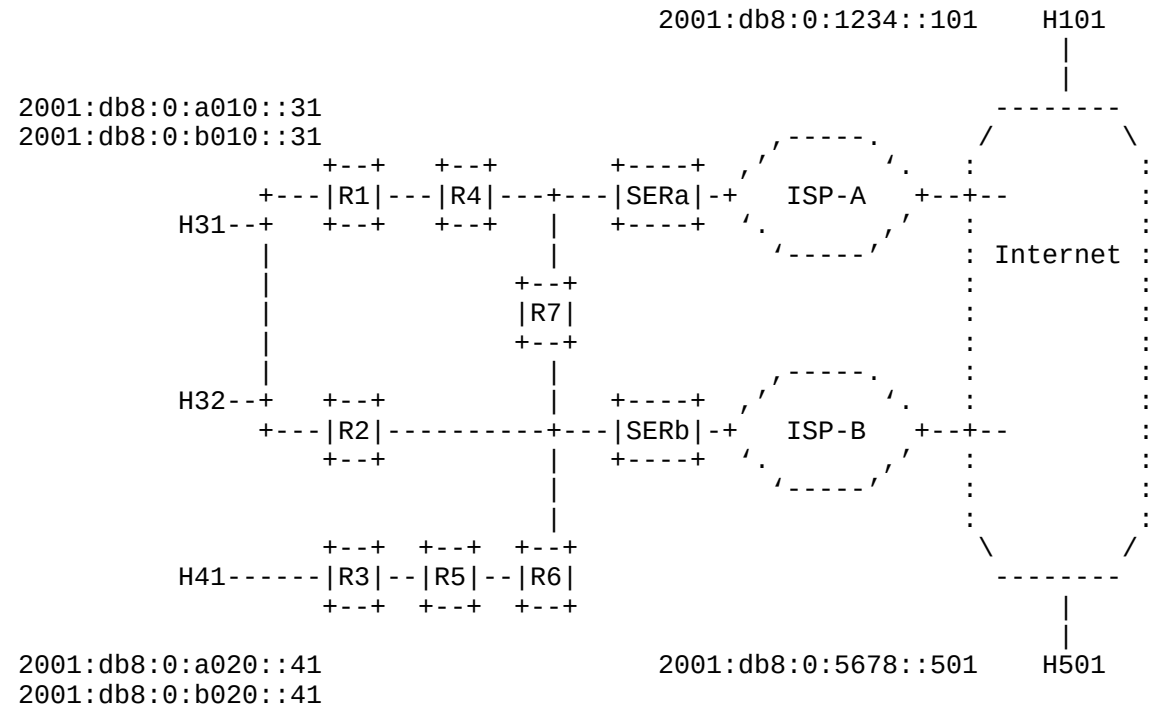


Figure 2: Simple ISP Connectivity Where SERs Are Not Directly Connected

One option is to effectively modify the topology by creating a logical tunnel between SERa and SERb, using GRE for example. Although SERa and SERb are not directly connected physically in this topology, they can be directly connected logically by a tunnel.

The other option is to enable SADR functionality on R7. In this way, R7 will exchange source-scoped routes with SERa and SERb, making the three routers act as a single SADR domain. This illustrates the basic principle that the minimum requirement for the routed site network to support PA multi-homing is having all of the site exit routers be part of a connected SADR domain. Extending the connected SADR domain beyond that point can produce more efficient forwarding paths.

2.3. Enterprise Network Operator Expectations

Before considering a more complex scenario, let’s look in more detail at the reasonably simple multihoming scenario in Figure 2 to understand what can reasonably be expected from this solution. As a

general guiding principle, we assume an enterprise network operator will expect a multihomed network to behave as close as to a single-homed network as possible. So a solution that meets those expectations where possible is a good thing.

For traffic between internal hosts and traffic from outside the site to internal hosts, an enterprise network operator would expect there to be no visible change in the path taken by this traffic, since this traffic does not need to be routed in a way that depends on source address. It is also reasonable to expect that internal hosts should be able to communicate with each other using either of their source addresses without restriction. For example, H31 should be able to communicate with H41 using a packet with S=2001:db8:0:a010::31, D=2001:db8:0:b010::41, regardless of the state of uplink to ISP-B.

These goals can be accomplished by having all of the routers in the network continue to originate normal unscoped destination routes for their connected networks. If we can arrange so that these unscoped destination routes get used for forwarding this traffic, then we will have accomplished the goal of keeping forwarding of traffic destined for internal hosts, unaffected by the multihoming solution.

For traffic destined for external hosts, it is reasonable to expect that traffic with an source address from the prefix assigned by ISP-A to follow the path to that the traffic would follow if there is no connection to ISP-B. This can be accomplished by having SERa originate a source-scoped route of the form (S=2001:db8:0:a000::/52, D=::/0) . If all of the routers in the site support SADR, then the path of traffic exiting via ISP-A can match that expectation. If some routers don’t support SADR, then it is reasonable to expect that the path for traffic exiting via ISP-A may be different within the site. This is a tradeoff that the enterprise network operator may decide to make.

It is important to understand how this multihoming solution behaves when an uplink to one of the ISPs fails. To simplify this discussion, we assume that all routers in the site support SADR. We first start by looking at how the network operates when the uplinks to both ISP-A and ISP-B are functioning properly. SERa originates a source-scoped route of the form (S=2001:db8:0:a000::/52, D=::/0), and SERb is originates a source-scoped route of the form (S=2001:db8:0:b000::/52, D=::/0). These routes are distributed through the routers in the site, and they establish within the routers two set of forwarding paths for traffic leaving the site. One set of forwarding paths is for packets with source address in 2001:db8:0:a000::/52. The other set of forwarding paths is for packets with source address in 2001:db8:0:b000::/52. The normal destination routes which are not scoped to these two source prefixes

play no role in the forwarding. Whether a packet exits the site via SERa or via SERb is completely determined by the source address applied to the packet by the host. So for example, when host H31 sends a packet to host H101 with (S=2001:db8:0:a010::31, D=2001:db8:0:1234::101), the packet will only be sent out the link from SERa to ISP-A.

Now consider what happens when the uplink from SERa to ISP-A fails. The only way for the packets from H31 to reach H101 is for H31 to start using the source address for ISP-B. H31 needs to send the following packet: (S=2001:db8:0:b010::31, D=2001:db8:0:1234::101).

This behavior is very different from the behavior that occurs with site multihoming using PI addresses or with PA addresses using NAT. In these other multi-homing solutions, hosts do not need to react to network failures several hops away in order to regain Internet access. Instead, a host can be largely unaware of the failure of an uplink to an ISP. When multihoming with PA addresses and NAT, existing sessions generally need to be re-established after a failure since the external host will receive packets from the internal host with a new source address. However, new sessions can be established without any action on the part of the hosts.

Another example where the behavior of this multihoming solution differs significantly from that of multihoming with PI address or with PA addresses using NAT is in the ability of the enterprise network operator to route traffic over different ISPs based on destination address. We still consider the fairly simple network of Figure 2 and assume that uplinks to both ISPs are functioning. Assume that the site is multihomed using PA addresses and NAT, and that SERa and SERb each originate a normal destination route for D=::/0, with the route origination dependent on the state of the uplink to the respective ISP.

Now suppose it is observed that an important application running between internal hosts and external host H101 experience much better performance when the traffic passes through ISP-A (perhaps because ISP-A provides lower latency to H101.) When multihoming this site with PI addresses or with PA addresses and NAT, the enterprise network operator can configure SERa to originate into the site network a normal destination route for D=2001:db8:0:1234::/64 (the destination prefix to reach H101) that depends on the state of the uplink to ISP-A. When the link to ISP-A is functioning, the destination route D=2001:db8:0:1234::/64 will be originated by SERa, so traffic from all hosts will use ISP-A to reach H101 based on the longest destination prefix match in the route lookup.

Implementing the same routing policy is more difficult with the PA multihoming solution described in this document since it doesn't use NAT. By design, the only way to control where a packet exits this network is by setting the source address of the packet. Since the network cannot modify the source address without NAT, the host must set it. To implement this routing policy, each host needs to use the source address from the prefix assigned by ISP-A to send traffic destined for H101. Mechanisms have been proposed to allow hosts to choose the source address for packets in a fine grained manner. We will discuss these proposals in Section 4. However, interacting with host operating systems in some manner to ensure a particular source address is chosen for a particular destination prefix is not what an enterprise network administrator would expect to have to do to implement this routing policy.

2.4. More complex ISP connectivity

The previous sections considered two variations of a simple multihoming scenario where the site is connected to two ISPs offering only Internet connectivity. It is likely that many actual enterprise multihoming scenarios will be similar to this simple example. However, there are more complex multihoming scenarios that we would like this solution to address as well.

It is fairly common for an ISP to offer a service in addition to Internet access over the same uplink. Two variation of this are reflected in Figure 3. In addition to Internet access, ISP-A offers a service which requires the site to access host H51 at 2001:db8:0:5555::51. The site has a single physical and logical connection with ISP-A, and ISP-A only allows access to H51 over that connection. So when H32 needs to access the service at H51 it needs to send packets with (S=2001:db8:0:a010::32, D=2001:db8:0:5555::51) and those packets need to be forward out the link from SERa to ISP-A.

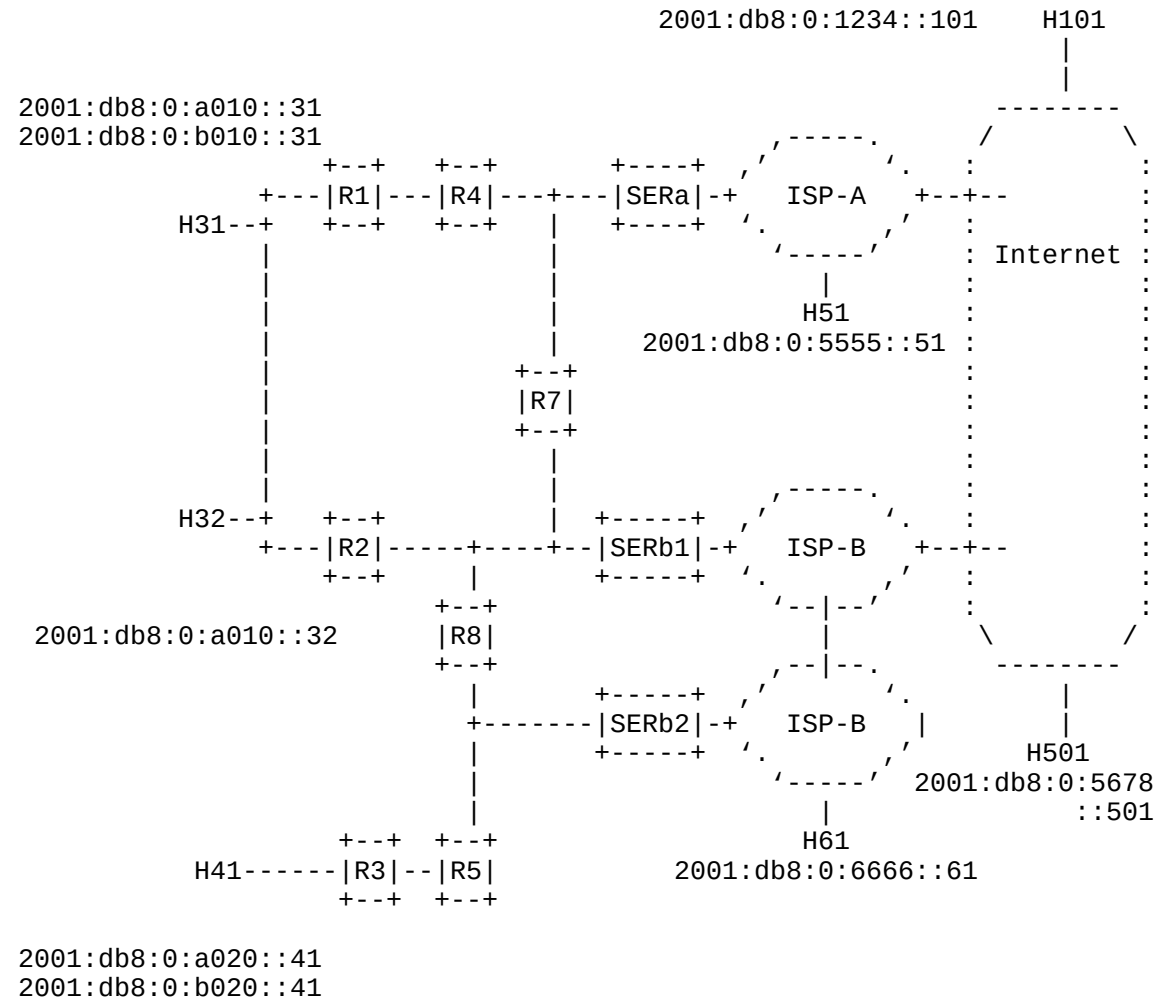


Figure 3: Internet access and services offered by ISP-A and ISP-B

ISP-B illustrates a variation on this scenario. In addition to Internet access, ISP-B also offers a service which requires the site to access host H61. The site has two connections to two different parts of ISP-B (shown as SERb1 and SERb2 in Figure 3). ISP-B expects Internet traffic to use the uplink from SERb1, while it expects it expects traffic destined for the service at H61 to use the uplink from SERb2. For either uplink, ISP-B expects the ingress traffic to have a source address matching the prefix it assigned to the site, 2001:db8:0:b000::/52.

As discussed before, we rely completely on the internal host to set the source address of the packet properly. In the case of a packet sent by H31 to access the service in ISP-B at H61, we expect the packet to have the following addresses: (S=2001:db8:0:b010::31, D=2001:db8:0:6666::61). The routed network has two potential ways of distributing routes so that this packet exits the site on the uplink at SERb2.

We could just rely on normal destination routes, without using source-prefix scoped routes. If we have SERb2 originate a normal unscoped destination route for D=2001:db8:0:6666::/64, the packets from H31 to H61 will exit the site at SERb2 as desired. We should not have to worry about SERa needing to originate the same route, because ISP-B should choose a globally unique prefix for the service at H61.

The alternative is to have SERb2 originate a source-prefix-scoped destination route of the form (S=2001:db8:0:b000::/52, D=2001:db8:0:6666::/64). From a forwarding point of view, the use of the source-prefix-scoped destination route would result in traffic with source addresses corresponding only to ISP-B being sent to SERb2. Instead, the use of the unscoped destination route would result in traffic with source addresses corresponding to ISP-A and ISP-B being sent to SERb2, as long as the destination address matches the destination prefix. It seems like either forwarding behavior would be acceptable.

However, from the point of view of the enterprise network administrator trying to configure, maintain, and trouble-shoot this multihoming solution, it seems much clearer to have SERb2 originate the source-prefix-scoped destination route correspond to the service offered by ISP-B. In this way, all of the traffic leaving the site is determined by the source-prefix-scoped routes, and all of the traffic within the site or arriving from external hosts is determined by the unscoped destination routes. Therefore, for this multihoming solution we choose to originate source-prefix-scoped routes for all traffic leaving the site.

2.5. ISPs and Provider-Assigned Prefixes

While we expect that most site multihoming involves connecting to only two ISPs, this solution allows for connections to an arbitrary number of ISPs to be supported. However, when evaluating scalable implementations of the solution, it would be reasonable to assume that the maximum number of ISPs that a site would connect to is five.

It is also useful to note that the prefixes assigned to the site by different ISPs will not overlap. This must be the case , since the provider-assigned addresses have to be globally unique.

2.6. Simplified Topologies

The topologies of many enterprise sites using this multihoming solution may in practice be simpler than the examples that we have used. The topology in Figure 1 could be further simplified by having all hosts directly connected to the LAN connecting the two site exit routers, SERa and SERb. The topology could also be simplified by having the uplinks to ISP-A and ISP-B both connected to the same site exit router. However, it is the aim of this draft to provide a solution that applies to a broad a range of enterprise site network topologies, so this draft focuses on providing a solution to the more general case. The simplified cases will also be supported by this solution, and there may even be optimizations that can be made for simplified cases. This solution however needs to support more complex topologies.

We are starting with the basic assumption that enterprise site networks can be quite complex from a routing perspective. However, even a complex site network can be multihomed to different ISPs with PA addresses using IPv4 and NAT. It is not reasonable to expect an enterprise network operator to change the routing topology of the site in order to deploy IPv6.

3. Generating Source-Prefix-Scoped Forwarding Tables

So far we have described in general terms how the routers in this solution that are capable of Source Address Dependent Routing will forward traffic using both normal unscoped destination routes and source-prefix-scoped destination routes. Here we give a precise method for generating a source-prefix-scoped forwarding table on a router that supports SADR.

1. Compute the next-hops for the source-prefix-scoped destination prefixes using only routers in the connected SADR domain. These are the initial source-prefix-scoped forwarding table entries.

2. Compute the next-hops for the unscoped destination prefixes using all routers in the IGP. This is the unscoped forwarding table.

3. Augment each source-prefix-scoped forwarding table with unscoped forwarding table entries based on the following rule. If the destination prefix of the unscoped forwarding entry exactly matches the destination prefix of an existing source-prefix-scoped forwarding entry (including destination prefix length),

then do not add the unscoped forwarding entry. If the destination prefix does NOT match an existing entry, then add the entry to the source-prefix-scoped forwarding table.

The forward tables produced by this process are used in the following way to forward packets.

1. If the source address of the packet matches one of the source prefixes, then look up the destination address of the packet in the corresponding source-prefix-scoped forwarding table to determine the next-hop for the packet.

2. If the source address of the packet does NOT match one of the source prefixes, then look up the destination address of the packet in unscoped forwarding table to determine the next-hop for the packet.

The following example illustrates how this process is used to create a forwarding table for each provider-assigned source prefix. We consider the multihomed site network in Figure 3. Initially we assume that all of the routers in the site network support SADR. Figure 4 shows the routes that are originated by the routers in the site network.

```
Routes originated by SERa:
(S=2001:db8:0:a000::/52, D=2001:db8:0:5555/64)
(S=2001:db8:0:a000::/52, D=::/0)
(D=2001:db8:0:5555::/64)
(D=::/0)

Routes originated by SERb1:
(S=2001:db8:0:b000::/52, D=::/0)
(D=::/0)

Routes originated by SERb2:
(S=2001:db8:0:b000::/52, D=2001:db8:0:6666::/64)
(D=2001:db8:0:6666::/64)

Routes originated by R1:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R2:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R3:
(D=2001:db8:0:a020::/64)
(D=2001:db8:0:b020::/64)
```

Figure 4: Routes Originated by Routers in the Site Network

Each SER originates destination routes which are scoped to the source prefix assigned by the ISP that the SER connects to. Note that the SERs also originate the corresponding unscoped destination route. This is not needed when all of the routers in the site support SADR. However, it is required when some routers do not support SADR. This will be discussed in more detail later.

We focus on how R8 constructs its source-prefix-scoped forwarding tables from these route advertisements. R8 computes the next hops for destination routes which are scoped to the source prefix 2001:db8:0:a000::/52. The results are shown in the first table in Figure 5. (In this example, the next hops are computed assuming that all links have the same metric.) Then, R8 computes the next hops for destination routes which are scoped to the source prefix 2001:db8:0:b000::/52. The results are shown in the second table in Figure 5 . Finally, R8 computes the next hops for the unscoped destination prefixes. The results are shown in the third table in Figure 5.

```
forwarding entries scoped to
source prefix = 2001:db8:0:a000::/52
==========================================
D=2001:db8:0:5555/64      NH=R7
D=::/0                    NH=R7

forwarding entries scoped to
source prefix = 2001:db8:0:b000::/52
==========================================
D=2001:db8:0:6666/64      NH=SERb2
D=::/0                    NH=SERb1

unscoped forwarding entries
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1
```

Figure 5: Forwarding Entries Computed at R8

The final step is for R8 to augment the source-prefix-scoped forwarding entries with unscoped forwarding entries. If an unscoped forwarding entry has the exact same destination prefix as an source-prefix-scoped forwarding entry (including destination prefix length), then the source-prefix-scoped forwarding entry wins.

As as an example of how the source scoped forwarding entries are augmented with unscoped forwarding entries, we consider how the two entries in the first table in Figure 5 (the table for source prefix = 2001:db8:0:a000::/52) are augmented with entries from the third table in Figure 5 (the table of unscoped forwarding entries). The first four unscoped forwarding entries (D=2001:db8:0:a010::/64, D=2001:db8:0:b010::/64, D=2001:db8:0:a020::/64, and D=2001:db8:0:b020::/64) are not an exact match for any of the existing entries in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, these four entries are added to the final forwarding table for source prefix 2001:db8:0:a000::/52. The result of adding these entries is reflected in first four entries the first table in Figure 6.

The next unscoped forwarding table entry is for D=2001:db8:0:5555::/64. This entry is an exact match for the existing entry in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we do not replace the existing

entry with the entry from the unscoped forwarding table. This is reflected in the fifth entry in the first table in Figure 6. (Note that since both scoped and unscoped entries have R7 as the next hop, the result of applying this rule is not visible.)

The next unscoped forwarding table entry is for D=2001:db8:0:6666::/64. This entry is not an exact match for any existing entries in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we add this entry. This is reflected in the sixth entry in the first table in Figure 6.

The next unscoped forwarding table entry is for D=::/0. This entry is an exact match for the existing entry in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we do not overwrite the existing source-prefix-scoped entry, as can be seen in the last entry in the first table in Figure 6.

```
if source address matches 2001:db8:0:a000::/52
then use this forwarding table
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=R7

else if source address matches 2001:db8:0:b000::/52
then use this forwarding table
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1

else use this forwarding table
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1
```

Figure 6: Complete Forwarding Tables Computed at R8

The forwarding tables produced by this process at R8 have the desired properties. A packet with a source address in 2001:db8:0:a000::/52 will be forwarded based on the first table in Figure 6. If the packet is destined for the Internet at large or the service at D=2001:db8:0:5555/64, it will be sent to R7 in the direction of SERa. If the packet is destined for an internal host, then the first four entries will send it to R2 or R5 as expected. Note that if this packet has a destination address corresponding to the service offered by ISP-B (D=2001:db8:0:5555::/64), then it will get forwarded to SERb2. It will be dropped by SERb2 or by ISP-B, since it the packet has a source address that was not assigned by ISP-B. However, this is expected behavior. In order to use the service offered by ISP-B, the host needs to originate the packet with a source address assigned by ISP-B.

In this example, a packet with a source address that doesn't match 2001:db8:0:a000::/52 or 2001:db8:0:b000::/52 must have originated from an external host. Such a packet will use the unscoped forwarding table (the last table in Figure 6). These packets will flow exactly as they would in absence of multihoming.

We can also modify this example to illustrate how it supports deployments where not all routers in the site support SADR. Continuing with the topology shown in Figure 3, suppose that R3 and R5 do not support SADR. Instead they are only capable of understanding unscoped route advertisements. The SADR routers in the network will still originate the routes shown in Figure 4. However, R3 and R5 will only understand the unscoped routes as shown in Figure 7.

```
Routes originated by SERa:
(D=2001:db8:0:5555::/64)
(D=::/0)

Routes originated by SERb1:
(D=::/0)

Routes originated by SERb2:
(D=2001:db8:0:6666::/64)

Routes originated by R1:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R2:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R3:
(D=2001:db8:0:a020::/64)
(D=2001:db8:0:b020::/64)
```

Figure 7: Routes Advertisements Understood by Routers that do no Support SADR

With these unscoped route advertisements, R5 will produce the forwarding table shown in Figure 8.

```
forwarding table
=========================================
D=2001:db8:0:a010::/64    NH=R8
D=2001:db8:0:b010::/64    NH=R8
D=2001:db8:0:a020::/64    NH=R3
D=2001:db8:0:b020::/64    NH=R3
D=2001:db8:0:5555::/64    NH=R8
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=R8
```

Figure 8: Forwarding Table For R5, Which Doesn't Understand Source-Prefix-Scoped Routes

Any traffic that needs to exit the site will eventually hit a SADR-capable router. Once that traffic enters the SADR-capable domain, then it will not leave that domain until it exits the site. This property is required in order to guarantee that there will not be routing loops involving SADR-capable and non-SADR-capable routers.

Note that the mechanism described here for converting source-prefix-scoped destination prefix routing advertisements into forwarding state is somewhat different from that proposed in [I-D.ietf-rtgwg-dst-src-routing]. The method described in this document is intended to be easy to understand for network enterprise operators while at the same time being functionally correct. Another difference is that the method in this document assumes that source prefix will not overlap. Other differences between the two approaches still need to be understood and reconciled.

An interesting side-effect of deploying SADR is if all routers in a given network support SADR and have a scoped forwarding table, then the unscoped forwarding table can be eliminated which ensures that packets with legitimate source addresses only can leave the network (as there are no scoped forwarding tables for spoofed/bogon source addresses). It would prevent accidental leaks of ULA/reserved/link-local sources to the Internet as well as ensures that no spoofing is possible from the SADR-enabled network.

4. Mechanisms For Hosts To Choose Good Source Addresses In A Multihomed Site

Until this point, we have made the assumption that hosts are able to choose the correct source address using some unspecified mechanism. This has allowed us to just focus on what the routers in a multihomed site network need to do in order to forward packets to the correct ISP based on source address. Now we look at possible mechanisms for hosts to choose the correct source address. We also look at what

role, if any, the routers may play in providing information that helps hosts to choose source addresses.

Any host that needs to be able to send traffic using the uplinks to a given ISP is expected to be configured with an address from the prefix assigned by that ISP. The host will control which ISP is used for its traffic by selecting one of the addresses configured on the host as the source address for outgoing traffic. It is the responsibility of the site network to ensure that a packet with the source address from an ISP is not sent on an uplink to that ISP.

If all of the ISP uplinks are working, the choice of source address by the host may be driven by the desire to load share across ISP uplinks, or it may be driven by the desire to take advantage of certain properties of a particular uplink or ISP. If any of the ISP uplinks is not working, then the choice of source address by the host can determine if packets get dropped.

How a host should make good decisions about source address selection in a multihomed site is not a solved problem. We do not attempt to solve this problem in this document. Instead we discuss the current state of affairs with respect to standardized solutions and implementation of those solutions. We also look at proposed solutions for this problem.

An external host initiating communication with a host internal to a PA multihomed site will need to know multiple addresses for that host in order to communicate with it using different ISPs to the multihomed site. These addresses are typically learned through DNS. (For simplicity, we assume that the external host is single-homed.) The external host chooses the ISP that will be used at the remote multihomed site by setting the destination address on the packets it transmits. For a sessions originated from an external host to an internal host, the choice of source address used by the internal host is simple. The internal host has no choice but to use the destination address in the received packet as the source address of the transmitted packet.

For a session originated by a host internal to the multi-homed site, the decision of what source address to select is more complicated. We consider three main methods for hosts to get information about the network. The two proactive methods are Neighbor Discovery Router Advertisements and DHCPv6. The one reactive method we consider is ICMPv6. Note that we are explicitly excluding the possibility of having hosts participate in or even listen directly to routing protocol advertisements.

First we look at how a host is currently expected to select the source and destination address with which it sends a packet.

4.1. Source Address Selection Algorithm on Hosts

[RFC6724] defines the algorithms that hosts are expected to use to select source and destination addresses for packets. It defines an algorithm for selecting a source address and a separate algorithm for selecting a destination address. Both of these algorithms depend on a policy table. [RFC6724] defines a default policy which produces certain behavior.

The rules in the two algorithms in [RFC6724] depend on many different properties of addresses. While these are needed for understanding how a host should choose addresses in an arbitrary environment, most of the rules are not relevant for understanding how a host should choose among multiple source addresses in mulitihomed envinronment when sending a packet to a remote host. Returning to the example in Figure 3, we look at what the default algorithms in [RFC6724] say about the source address that internal host H31 should use to send traffic to external host H101, somewhere on the Internet. Let’s look at what rules in [RFC6724] are actually used by H31 in this case.

There is no choice to be made with respect to destination address. H31 needs to send a packet with D=2001:db8:0:1234::101 in order to reach H101. So H31 have to choose between using S=2001:db8:0:a010::31 or S=2001:db8:0:b010::31 as the source address for this packet. We go through the rules for source address selection in Section 5 of [RFC6724]. Rule 1 (Prefer same address) is not useful to break the tie between source addresses, because neither the candidate source addresses equals the destination address. Rule 2 (Prefer appropriate scope) is also not used in this scenario, because both source addresses and the destination address have global scope.

Rule 3 (Avoid deprecated addresses) applies to an address that has been autoconfigured by a host using stateless address autoconfiguration as defined in [RFC4862]. An address autoconfigured by a host has a preferred lifetime and a valid lifetime. The address is preferred until the preferred lifetime expires, after which it becomes deprecated. A deprecated address can still be used, but it is better to use a preferred address. When the valid lifetime expires, the address cannot be used at all. The preferred and valid lifetimes for an autoconfigured address are set based on the corresponding lifetimes in the Prefix Information Option in Neighbor Discovery Router Advertisements. So a possible tool to control source address selection in this scenario would be to a host to make an address deprecated by having routers on that link, R1 and R2 in

Figure 3, send Prefix Information Option messages with the preferred lifetime for the source prefix to be discouraged (or prohibited) set to zero. This is a rather blunt tool, because it discourages or prohibits the use of that source prefix for all destinations. However, it may be useful in some scenarios.

Rule 4 (Avoid home addresses) does not apply here because we are not considering Mobile IP.

Rule 5 (Prefer outgoing interface) is not useful in this scenario, because both source addresses are assigned to the same interface.

Rule 5.5 (Prefer addresses in a prefix advertised by the next-hop) is not useful in the scenario when both R1 and R2 will advertise both source prefixes. However potentially this rule may allow a host to select the correct source prefix by selecting a next-hop. The most obvious way would be to make R1 to advertise itself as a default router and send PIO for 2001:db8:0:a010::/64, while R2 is advertising itself as a default router and sending PIO for 2001:db8:0:b010::/64. We'll discuss later how Rule 5.5 can be used to influence a source address selection in single-router topologies (e.g. when H41 is sending traffic using R3 as a default gateway).

Rule 6 (Prefer matching label) refers to the Label value determined for each source and destination prefix as a result of applying the policy table to the prefix. With the default policy table defined in Section 2.1 of [RFC6724], Label(2001:db8:0:a010::31) = 5, Label(2001:db8:0:b010::31) = 5, and Label(2001:db8:0:1234::101) = 5. So with the default policy, Rule 6 does not break the tie. However, the algorithms in [RFC6724] are defined in such as way that non-default address selection policy tables can be used. [RFC7078] defines a way to distribute a non-default address selection policy table to hosts using DHCPv6. So even though the application of rule 6 to this scenario using the default policy table is not useful, rule 6 may still be a useful tool.

Rule 7 (Prefer temporary addresses) has to do with the technique described in [RFC4941] to periodically randomize the interface portion of an IPv6 address that has been generated using stateless address autoconfiguration. In general, if H31 were using this technique, it would use it for both source addresses, for example creating temporary addresses 2001:db8:0:a010:2839:9938:ab58:830f and 2001:db8:0:b010:4838:f483:8384:3208, in addition to 2001:db8:0:a010::31 and 2001:db8:0:b010::31. So this rule would prefer the two temporary addresses, but it would not break the tie between the two source prefixes from ISP-A and ISP-B.

Rule 8 (Use longest matching prefix) dictates that between two candidate source addresses the one which has longest common prefix length with the destination address. For example, if H31 were selecting the source address for sending packets to H101, this rule would not be a tie breaker as for both candidate source addresses 2001:db8:0:a101::31 and 2001:db8:0:b101::31 the common prefix length with the destination is 48. However if H31 were selecting the source address for sending packets H41 address 2001:db8:0:a020::41, then this rule would result in using 2001:db8:0:a101::31 as a source (2001:db8:0:a101::31 and 2001:db8:0:a020::41 share the common prefix 2001:db8:0:a000::/58, while for '2001:db8:0:b101::31 and 2001:db8:0:a020::41 the common prefix is 2001:db8:0:a000::/51). Therefore rule 8 might be useful for selecting the correct source address in some but not all scenarios (for example if ISP-B services belong to 2001:db8:0:b000::/59 then H31 would always use 2001:db8:0:b010::31 to access those destinations).

So we can see that of the 8 source selection address rules from [RFC6724], five actually apply to our basic site multihoming scenario. The rules that are relevant to this scenario are summarized below.

- Rule 3: Avoid deprecated addresses.
- Rule 5.5: Prefer addresses in a prefix advertised by the next-hop.
- Rule 6: Prefer matching label.
- Rule 8: Prefer longest matching prefix.

The two methods that we discuss for controlling the source address selection through the four relevant rules above are SLAAC Router Advertisement messages and DHCPv6.

We also consider a possible role for ICMPv6 for getting traffic-driven feedback from the network. With the source address selection algorithm discussed above, the goal is to choose the correct source address on the first try, before any traffic is sent. However, another strategy is to choose a source address, send the packet, get feedback from the network about whether or not the source address is correct, and try another source address if it is not.

We consider four scenarios where a host needs to select the correct source address. The first is when both uplinks are working. The second is when one uplink has failed. The third one is a situation when one failed uplink has recovered. The last one is failure of both (all) uplinks.

4.2. Selecting Source Address When Both Uplinks Are Working

Again we return to the topology in Figure 3. Suppose that the site administrator wants to implement a policy by which all hosts need to use ISP-A to reach H01 at D=2001:db8:0:1234::101. So for example, H31 needs to select S=2001:db8:0:a010::31.

4.2.1. Distributing Address Selection Policy Table with DHCPv6

This policy can be implemented by using DHCPv6 to distribute an address selection policy table that assigns the same label to destination address that match 2001:db8:0:1234::/64 as it does to source addresses that match 2001:db8:0:a000::/52. The following two entries accomplish this.

Prefix	Precedence	Label
2001:db8:0:1234::/64	50	33
2001:db8:0:a000::/52	50	33

Figure 9: Policy table entries to implement a routing policy

This requires that the hosts implement [RFC6724], the basic source and destination address framework, along with [RFC7078], the DHCPv6 extension for distributing a non-default policy table. Note that it does NOT require that the hosts use DHCPv6 for address assignment. The hosts could still use stateless address autoconfiguration for address configuration, while using DHCPv6 only for policy table distribution (see [RFC3736]). However this method has a number of disadvantages:

- DHCPv6 support is not a mandatory requirement for IPv6 hosts, so this method might not work for all devices.
- Network administrators are required to explicitly configure the desired network access policies on DHCPv6 servers.

4.2.2. Controlling Source Address Selection With Router Advertisements

Neighbor Discovery currently has two mechanisms to communicate prefix information to hosts. The base specification for Neighbor Discovery (see [RFC4861]) defines the Prefix Information Option (PIO) in the Router Advertisement (RA) message. When a host receives a PIO with the A-flag set, it can use the prefix in the PIO as source prefix from which it assigns itself an IP address using stateless address autoconfiguration (SLAAC) procedures described in [RFC4862]. In the example of Figure 3, if the site network is using SLAAC, we would expect both R1 and R2 to send RA messages with PIOs for both source prefixes 2001:db8:0:a010::/64 and 2001:db8:0:b010::/64 with the

A-flag set. H31 would then use the SLAAC procedure to configure itself with the 2001:db8:0:a010::31 and 2001:db8:0:b010::31.

Whereas a host learns about source prefixes from PIO messages, hosts can learn about a destination prefix from a Router Advertisement containing Route Information Option (RIO), as specified in [RFC4191]. The destination prefixes in RIOs are intended to allow a host to choose the router that it uses as its first hop to reach a particular destination prefix.

As currently standardized, neither PIO nor RIO options contained in Neighbor Discovery Router Advertisements can communicate the information needed to implement the desired routing policy. PIO's communicate source prefixes, and RIO communicate destination prefixes. However, there is currently no standardized way to directly associate a particular destination prefix with a particular source prefix.

[I-D.pfister-6man-sadr-ra] proposes a Source Address Dependent Route Information option for Neighbor Discovery Router Advertisements which would associate a source prefix and with a destination prefix. The details of [I-D.pfister-6man-sadr-ra] might need tweaking to address this use case. However, in order to be able to use Neighbor Discovery Router Advertisements to implement this routing policy, an extension that allows a R1 and R2 to explicitly communicate to H31 an association between S=2001:db8:0:a000::/52 D=2001:db8:0:1234::/64 would be needed.

However the Rule 5.5 of the source address selection (discussed above) together with default router preference (specified in [RFC4191]) and RIO can be used to influence a source address selection on a host as described below. Let's look at source address selection on the host H41. It receives RAs from R3 with PIOs for 2001:db8:0:a020::/64 and 2001:db8:0:b020::/64. At that point all traffic would use the same next-hop (R3 link-local address) so Rule 5.5 does not apply. Now let's assume that R3 supports SADR and has two scoped forwarding tables, one scoped to S=2001:db8:0:a000::/52 and another scoped to S=2001:db8:0:b000::/52. If R3 generates two different link-local addresses for its interface facing H41 (one for each scoped forwarding table, LLA_A and LLA_B) and starts sending two different RAs: one is sent from LLA_A and includes PIO for 2001:db8:0:a020::/64, another us sent from LLA_B and includes PIO for 2001:db8:0:b020::/64. Now it is possible to influence H41 source address selection for destinations which follow the default route by setting default router preference in RAs. If it is desired that H41 reaches H101 (or any destinations in the Internet) via ISP-A, then RAs sent from LLA_A should have default router preference set to 01 (high priority), while RAs sent from LLA_B should have preference set

to 11 (low). Then LLA_A would be chosen as a next-hop for H101 and therefore (as per rule 5.5) 2001:db8:0:a020::41 would be selected as the source address. If, at the same time, it is desired that H61 is accessible via ISP-B then R3 should include a RIO for 2001:db8:0:6666::/64 to its RA sent from LLA_B. H41 would chose LLA_B as a next-hop for all traffic to H61 and then as per Rule 5.5, 2001:db8:0:b020::41 would be selected as a source address.

If in the above mentioned scenario it is desirable that all Internet traffic leaves the network via ISP-A and the link to ISP-B is used for accessing ISP-B services only (not as ISP-A link backup), then RAs sent by R3 from LLA_B should have Router Lifetime set to 0 and should include RIOs for ISP-B address space. It would instruct H41 to use LLA_A for all Internet traffic but use LLA_B as a next-hop while sending traffic to ISP-B addresses.

The proposed solution relies on SADR support by first-hop routers as well as SERs.

4.2.3. Controlling Source Address Selection With ICMPv6

We now discuss how one might use ICMPv6 to implement the routing policy to send traffic destined for H101 out the uplink to ISP-A, even when uplinks to both ISPs are working. If H31 started sending traffic to H101 with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101, it would be routed through SER-b1 and out the uplink to ISP-B. SERb1 could recognize that this is traffic is not following the desired routing policy and react by sending an ICMPv6 message back to H31.

In this example, we could arrange things so that SERb1 drops the packet with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101, and then sends to H31 an ICMPv6 Destination Unreachable message with Code 5 (Source address failed ingress/egress policy). When H31 receives this packet, it would then be expected to try another source address to reach the destination. In this example, H31 would then send a packet with S=2001:db8:0:a010::31 and D=2001:db8:0:1234::101, which will reach SERa and be forwarded out the uplink to ISP-A.

However, we would also want it to be the case that SERb1 does not enforce this routing policy when the uplink from SERa to ISP-A has failed. This could be accomplished by having SERa originate a source-prefix-scoped route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) and have SERb1 monitor the presence of that route. If that route is not present (because SERa has stopped originating it), then SERb1 will not enforce the routing policy, and it will forward packets with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101 out its uplink to ISP-B.

We can also use this source-prefix-scoped route originated by SERa to communicate the desired routing policy to SERb1. We can define an EXCLUSIVE flag to be advertised together with the IGP route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64). This would allow SERa to communicate to SERb that SERb should reject traffic for D=2001:db8:0:1234::/64 and respond with an ICMPv6 Destination Unreachable Code 5 message, as long as the route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) is present.

Finally, if we are willing to extend ICMPv6 to support this solution, then we could create a mechanism for SERb1 to tell the host what source address it should be using to successfully forward packets that meet the policy. In its current form, when SERb1 sends an ICMPv6 Destination Unreachable Code 5 message, it is basically saying, "This source address is wrong. Try another source address." It would be better is if the ICMPv6 message could say, "This source address is wrong. Instead use a source address in S=2001:db8:0:a000::/52."

However using ICMPv6 for signalling source address information back to hosts introduces new challenges. Most routers currently have software or hardware limits on generating ICMP messages. An site administrator deploying a solution that relies on the SERs generating ICMP messages could try to improve the performance of SERs for generating ICMP messages. However, in a large network, it is still likely that ICMP message generation limits will be reached. As a result hosts would not receive ICMPv6 back which in turns leads to traffic blackholing and poor user experience. To improve the scalability of ICMPv6-based signalling hosts SHOULD cache the preferred source address (or prefix) for the given destination. In addition, the same source prefix SHOULD be used for other destinations in the same /64 as the original destination address. The source prefix SHOULD have a specific lifetime. Expiration of the lifetime SHOULD trigger the source address selection algorithm again.

Using ICMPv6 Code 5 message for influencing source address selection allows an attacker to exhaust the list of candidate source addresses on the host by sending spoofed ICMPv6 Code 5 for all prefixes known on the network (therefore preventing a victim from establishing a communication with the destination host). To protect from such attack hosts SHOULD verify that the original packet header included into ICMPv6 error message was actually sent by the host.

4.2.4. Summary of Methods For Controlling Source Address Selection To Implement Routing Policy

So to summarize this section, we have looked at three methods for implementing a simple routing policy where all traffic for a given destination on the Internet needs to use a particular ISP, even when the uplinks to both ISPs are working.

The default source address selection policy cannot distinguish between the source addresses needed to enforce this policy, so a non-default policy table using associating source and destination prefixes using Label values would need to be installed on each host. A mechanism exists for DHCPv6 to distribute a non-default policy table but such solution would heavily rely on DHCPv6 support by host operating system. Moreover there is no mechanism to translate desired routing/traffic engineering policies into policy tables on DHCPv6 servers. Therefore using DHCPv6 for controlling address selection policy table is not recommended and SHOULD NOT be used.

At the same time Router Advertisements provide a reliable mechanism to influence source address selection process via PIO, RIO and default router preferences. As all those options have been standardized by IETF and are supported by various operating systems, no changes are required on hosts. First-hop routers in the enterprise network need to be able of sending different RAs for different SLAAC prefixes (either based on scoped forwarding tables or based on pre-configured policies).

SERs can enforce the routing policy by sending ICMPv6 Destination Unreachable messages with Code 5 (Source address failed ingress/ egress policy) for traffic that is being sent with the wrong source address. The policy distribution can be automated by defining an EXCLUSIVE flag for the source-prefix-scoped route which can be set on the SER that originates the route. As ICMPv6 message generation can be rate-limited on routers, it SHOULD NOT be used as the only mechanism to influence source address selection on hosts. While hosts SHOULD select the correct source address for a given destination the network SHOULD signal any source address issues back to hosts using ICMPv6 error messages.

4.3. Selecting Source Address When One Uplink Has Failed

Now we discuss if DHCPv6, Neighbor Discovery Router Advertisements, and ICMPv6 can help a host choose the right source address when an uplink to one of the ISPs has failed. Again we look at the scenario in Figure 3. This time we look at traffic from H31 destined for external host H501 at D=2001:db8:0:5678::501. We initially assume

that the uplink from SERa to ISP-A is working and that the uplink from SERb1 to ISP-B is working.

We assume there is no particular routing policy desired, so H31 is free to send packets with S=2001:db8:0:a010::31 or S=2001:db8:0:b010::31 and have them delivered to H501. For this example, we assume that H31 has chosen S=2001:db8:0:b010::31 so that the packets exit via SERb to ISP-B. Now we see what happens when the link from SERb1 to ISP-B fails. How should H31 learn that it needs to start sending the packet to H501 with S=2001:db8:0:a010::31 in order to start using the uplink to ISP-A? We need to do this in a way that doesn't prevent H31 from still sending packets with S=2001:db8:0:b010::31 in order to reach H61 at D=2001:db8:0:6666::61.

4.3.1. Controlling Source Address Selection With DHCPv6

For this example we assume that the site network in Figure 3 has a centralized DHCP server and all routers act as DHCP relay agents. We assume that both of the addresses assigned to H31 were assigned via DHCP.

We could try to have the DHCP server monitor the state of the uplink from SERb1 to ISP-B in some manner and then tell H31 that it can no longer use S=2001:db8:0:b010::31 by settings its valid lifetime to zero. The DHCP server could initiate this process by sending a Reconfigure Message to H31 as described in Section 19 of [RFC3315]. Or the DHCP server can assign addresses with short lifetimes in order to force clients to renew them often.

This approach would prevent H31 from using S=2001:db8:0:b010::31 to reach the a host on the Internet. However, it would also prevent H31 from using S=2001:db8:0:b010::31 to reach H61 at D=2001:db8:0:6666::61, which is not desirable.

Another potential approach is to have the DHCP server monitor the uplink from SERb1 to ISP-B and control the choice of source address on H31 by updating its address selection policy table via the mechanism in [RFC7078]. The DHCP server could initiate this process by sending a Reconfigure Message to H31. Note that [RFC3315] requires that Reconfigure Message use DHCP authentication. DHCP authentication could be avoided by using short address lifetimes to force clients to send Renew messages to the server often. If the host is not obtaining its IP addresses from the DHCP server, then it would need to use the Information Refresh Time option defined in [RFC4242].

If the following policy table can be installed on H31 after the failure of the uplink from SERb1, then the desired routing behavior

should be achieved based on source and destination prefix being matched with label values.

Prefix	Precedence	Label
::/0	50	44
2001:db8:0:a000::/52	50	44
2001:db8:0:6666::/64	50	55
2001:db8:0:b000::/52	50	55

Figure 10: Policy Table Needed On Failure Of Uplink From SERb1

The described solution has a number of significant drawbacks, some of them already discussed in Section 4.2.1.

- o DHCPv6 support is not required for an IPv6 host and there are operating systems which do not support DHCPv6. Besides that, it does not appear that [RFC7078] has been widely implemented on host operating systems.

- o [RFC7078] does not clearly specify this kind of a dynamic use case where address selection policy needs to be updated quickly in response to the failure of a link. In a large network it would present scalability issues as many hosts need to be reconfigured in very short period of time.

- o No mechanism exists for making DHCPv6 servers aware of network topology/routing changes in the network. In general DHCPv6 servers monitoring network-related events sounds like a bad idea as completely new functionality beyond the scope of DHCPv6 role is required.

4.3.2. Controlling Source Address Selection With Router Advertisements

The same mechanism as discussed in Section 4.2.2 can be used to control the source address selection in the case of an uplink failure. If a particular prefix should not be used as a source for any destinations, then the router needs to send RA with Preferred Lifetime field for that prefix set to 0.

Let’s consider a scenario when all uplinks are operational and H41 receives two different RAs from R3: one from LLA_A with PIO for 2001:db8:0:a020::/64, default router preference set to 11 (low) and another one from LLA_B with PIO for 2001:db8:0:a020::/64, default router preference set to 01 (high) and RIO for 2001:db8:0:6666::/64. As a result H41 is using 2001:db8:0:b020::41 as a source address for all Internet traffic and those packets are sent by SERs to ISP-B. If SERb1 uplink to ISP-B failed, the desired behavior is that H41 stops

using 2001:db8:0:b020::41 as a source address for all destinations but H61. To achieve that R3 should react to SERb1 uplink failure (which could be detected as the scoped route (S=2001:db8:0:b000::/52, D=::/0) disappearance) by withdrawing itself as a default router. R3 sends a new RA from LLA_B with Router Lifetime value set to 0 (which means that it should not be used as default router). That RA still contains PIO for 2001:db8:0:b020::/64 (for SLAAC purposes) and RIO for 2001:db8:0:6666::/64 so H41 can reach H61 using LLA_B as a next-hop and 2001:db8:0:b020::41 as a source address. For all traffic following the default route, LLA_A will be used as a next-hop and 2001:db8:0:a020::41 as a source address.

If all uplinks to ISP-B have failed and therefore source addresses from ISP-B address space should not be used at all, the forwarding table scoped S=2001:db8:0:b000::/52 contains no entries. Hosts can be instructed to stop using source addresses from that block by sending RAs containing PIO with Preferred Lifetime set to 0.

4.3.3. Controlling Source Address Selection With ICMPv6

Now we look at how ICMPv6 messages can provide information back to H31. We assume again that at the time of the failure H31 is sending packets to H501 using (S=2001:db8:0:b010::31, D=2001:db8:0:5678::501). When the uplink from SERb1 to ISP-B fails, SERb1 would stop originating its source-prefix-scoped route for the default destination (S=2001:db8:0:b000::/52, D=::/0) as well as its unscoped default destination route. With these routes no longer in the IGP, traffic with (S=2001:db8:0:b010::31, D=2001:db8:0:5678::501) would end up at SERa based on the unscoped default destination route being originated by SERa. Since that traffic has the wrong source address to be forwarded to ISP-A, SERa would drop it and send a Destination Unreachable message with Code 5 (Source address failed ingress/egress policy) back to H31. H31 would then know to use another source address for that destination and would try with (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501). This would be forwarded to SERa based on the source-prefix-scoped default destination route still being originated by SERa, and SERa would forward it to ISP-A. As discussed above, if we are willing to extend ICMPv6, SERa can even tell H31 what source address it should use to reach that destination. The expected host behaviour has been discussed in Section 4.2.3. Potential issue with using ICMPv6 for signalling source address issues back to hosts is that uplink to an ISP-B failure immediately invalidates source addresses from 2001:db8:0:b000::/52 for all hosts which triggers a large number of ICMPv6 being sent back to hosts - the same scalability/rate limiting issues discussed in Section 4.2.3 would apply.

4.3.4. Summary Of Methods For Controlling Source Address Selection On The Failure Of An Uplink

It appears that DHCPv6 is not particularly well suited to quickly changing the source address used by a host in the event of the failure of an uplink, which eliminates DHCPv6 from the list of potential solutions. On the other hand Router Advertisements provides a reliable mechanism to dynamically provide hosts with a list of valid prefixes to use as source addresses as well as prevent particular prefixes to be used. While no additional new features are required to be implemented on hosts, routers need to be able to send RAs based on the state of scoped forwarding tables entries and to react to network topology changes by sending RAs with particular parameters set.

The use of ICMPv6 Destination Unreachable messages generated by the SER (or any SADR-capable) routers seem like they have the potential to provide a support mechanism together with RAs to signal source address selection errors back to hosts, however scalability issues may arise in large networks in case of sudden topology change. Therefore it is highly desirable that hosts are able to select the correct source address in case of uplinks failure with ICMPv6 being an additional mechanism to signal unexpected failures back to hosts.

The current behavior of different host operating system when receiving ICMPv6 Destination Unreachable message with code 5 (Source address failed ingress/egress policy) is not clear to the authors. Information from implementers, users, and testing would be quite helpful in evaluating this approach.

4.4. Selecting Source Address Upon Failed Uplink Recovery

The next logical step is to look at the scenario when a failed uplink on SERb1 to ISP-B is coming back up, so hosts can start using source addresses belonging to 2001:db8:0:b000::/52 again.

4.4.1. Controlling Source Address Selection With DHCPv6

The mechanism to use DHCPv6 to instruct the hosts (H31 in our example) to start using prefixes from ISP-B space (e.g. S=2001:db8:0:b010::31 for H31) to reach hosts on the Internet is quite similar to one discussed in Section 4.3.1 and shares the same drawbacks.

4.4.2. Controlling Source Address Selection With Router Advertisements

Let’s look at the scenario discussed in Section 4.3.2. If the uplink(s) failure caused the complete withdrawal of prefixes from 2001:db8:0:b000::/52 address space by setting Preferred Lifetime value to 0, then the recovery of the link should just trigger new RA being sent with non-zero Preferred Lifetime. In another scenario discussed in Section 4.3.2, the SERb1 uplink to ISP-B failure leads to disappearance of the (S=2001:db8:0:b000::/52, D=::/0) entry from the forwarding table scoped to S=2001:db8:0:b000::/52 and, in turn, caused R3 to send RAs from LLA_B with Router Lifetime set to 0. The recovery of the SERb1 uplink to ISP-B leads to (S=2001:db8:0:b000::/52, D=::/0) scoped forwarding entry re-appearance and instructs R3 that it should advertise itself as a default router for ISP-B address space domain (send RAs from LLA_B with non-zero Router Lifetime).

4.4.3. Controlling Source Address Selection With ICMP

It looks like ICMPv6 provides a rather limited functionality to signal back to hosts that particular source addresses have become valid again. Unless the changes in the uplink state a particular (S,D) pair, hosts can keep using the same source address even after an ISP uplink has come back up. For example, after the uplink from SERb1 to ISP-B had failed, H31 received ICMPv6 Code 5 message (as described in Section 4.3.3) and allegedly started using (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501) to reach H501. Now when the SERb1 uplink comes back up, the packets with that (S,D) pair are still routed to SERa1 and sent to the Internet. Therefore H31 is not informed that it should stop using 2001:db8:0:a010::31 and start using 2001:db8:0:b010::31 again. Unless SERa has a policy configured to drop packets (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501) and send ICMPv6 back if SERb1 uplink to ISP-B is up, H31 will be unaware of the network topology change and keep using S=2001:db8:0:a010::31 for Internet destinations, including H51.

One of the possible option may be using a scoped route with EXCLUSIVE flag as described in Section 4.2.3. SERa1 uplink recovery would cause (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) route to reappear in the routing table. In the absence of that route packets to H101 which were sent to ISP-B (as ISP-A uplink was down) with source addresses from 2001:db8:0:b000::/52. When the route re-appears SERb1 would reject those packets and sends ICMPv6 back as discussed in Section 4.2.3. Practically it might lead to scalability issues which have been already discussed in Section 4.2.3 and Section 4.4.3.

4.4.4. Summary Of Methods For Controlling Source Address Selection Upon Failed Uplink Recovery

Once again DHCPv6 does not look like reasonable choice to manipulate source address selection process on a host in the case of network topology changes. Using Router Advertisement provides the flexible mechanism to dynamically react to network topology changes (if routers are able to use routing changes as a trigger for sending out RAs with specific parameters). ICMPv6 could be considered as a supporting mechanism to signal incorrect source address back to hosts but should not be considered as the only mechanism to control the address selection in multihomed environments.

4.5. Selecting Source Address When All Uplinks Failed

One particular tricky case is a scenario when all uplinks have failed. In that case there is no valid source address to be used for any external destinations while it might be desirable to have intra-site connectivity.

4.5.1. Controlling Source Address Selection With DHCPv6

From DHCPv6 perspective uplinks failure should be treated as two independent failures and processed as described in Section 4.3.1. At this stage it is quite obvious that it would result in quite complicated policy table which needs to be explicitly configured by administrators and therefore seems to be impractical.

4.5.2. Controlling Source Address Selection With Router Advertisements

As discussed in Section 4.3.2 an uplink failure causes the scoped default entry to disappear from the scoped forwarding table and triggers RAs with zero Router Lifetime. Complete disappearance of all scoped entries for a given source prefix would cause the prefix being withdrawn from hosts by setting Preferred Lifetime value to zero in PIO. If all uplinks (SERa, SERb1 and SERb2) failed, hosts either lost their default routers and/or have no global IPv6 addresses to use as a source. (Note that ’uplink failure’ might mean ’IPv6 connectivity failure with IPv4 still being reachable’, in which case hosts might fall back to IPv4 if there is IPv4 connectivity to destinations). As a results intra-site connectivity is broken. One of the possible way to solve it is to use ULAs.

All hosts have ULA addresses assigned in addition to GUAs and used for intra-site communication even if there is no GUA assigned to a host. To avoid accidental leaking of packets with ULA sources SADR-capable routers SHOULD have a scoped forwarding table for ULA source for internal routes but MUST NOT have an entry for D=::/0 in that

table. In the absence of (S=ULA_Prefix; D=::/0) first-hop routers will send dedicated RAs from a unique link-local source LLA_ULA with PIO from ULA address space, RIO for the ULA prefix and Router Lifetime set to zero. The behaviour is consistent with the situation when SERb1 lost the uplink to ISP-B (so there is no Internet connectivity from 2001:db8:0:b000::/52 sources) but those sources can be used to reach some specific destinations. In the case of ULA there is no Internet connectivity from ULA sources but they can be used to reach another ULA destinations. Note that ULA usage could be particularly useful if all ISPs assign prefixes via DHCP-PD. In the absence of ULAs uplinks failure hosts would lost all their GUAs upon prefix lifetime expiration which again makes intra-site communication impossible.

4.5.3. Controlling Source Address Selection With ICMPv6

In case of all uplinks failure all SERs will drop outgoing IPv6 traffic and respond with ICMPv6 error message. In the large network when many hosts are trying to reach Internet destinations it means that SERs need to generate an ICMPv6 error to every packet they receive from hosts which presents the same scalability issues discussed in Section 4.3.3

4.5.4. Summary Of Methods For Controlling Source Address Selection When All Uplinks Failed

Again, combining SADR with Router Advertisements seems to be the most flexible and scalable way to control the source address selection on hosts.

4.6. Summary Of Methods For Controlling Source Address Selection

To summarize the scenarios and options discussed above:

While DHCPv6 allows administrators to manipulate source address selection policy tables, this method has a number of significant disadvantages which eliminates DHCPv6 from a list of potential solutions:

1. It required hosts to support DHCPv6 and its extension (RFC7078);

2. DHCPv6 server need to monitor network state and detect routing changes.

3. Network topology/routing policy changes could trigger simultaneous re-configuration of large number of hosts which present serious scalability issues.

The use of Router Advertisements to influence the source address selection on hosts seem to be the most reliable, flexible and scalable solution. It has the following benefits:

1. no new (non-standard) functionality needs to be implemented on hosts (except for [RFC4191] support);

2. no changes in RA format;

3. Routers can react to routing table changes by sending RAs which would minimize the failover time in the case of network topology changes;

4. information required for source address selection is broadcast to all affected hosts in case of topology change event which improves the scalability of the solution (comparing to DHCPv6 reconfiguration or ICMPv6 error messages).

To fully benefit from the RA-based solution, first-hop routers need to implement SADR and be able to send dedicated RAs per scoped forwarding table as discussed above, reacting to network changes with sending new RAs. It should be noted that the proposed solution would work even if first-hop routers are not SADR-capable but still able to send individual RAs for each ISP prefix and react to topology changes as discussed above

The RA-based solution relies heavily on hosts correctly implementing default address selection algorith as defined in [RFC6724] and in particular, Rule 5.5. There are some evidences that not all host OSes have that rule implemented currenly (it should be noted that [I-D.ietf-6man-multi-homed-host] states that Rule 5.5 SHOULD be implemented.

ICMPv6 Code 5 error message SHOULD be used to complement RA-based solution to signal incorrect source address selection back to hosts, but it SHOULD NOT be considered as the stand-alone solution. To prevent scenarios when hosts in multihomed envinronments incorrectly identify onlink/offlink destimations, hosts should treat ICMPv6 Redirects as discussed in [I-D.ietf-6man-multi-homed-host].

4.7. Other Configuration Parameters

4.7.1. DNS Configuration

In mutihomed envinronment each ISP might provide their own list of DNS servers. E.g. in the topology show on Figure 3, ISP-A might provide recursive DNS server H51 2001:db8:0:5555::51, while ISP-B might provide H61 2001:db8:0:6666::61 as a recursive DNS server. If

the multihomed enterprise network is not running their own recursive resolver then hosts need to be configured with DNS server IPv6 addresses. [RFC6106] defines IPv6 Router Advertisement options to allow IPv6 routers to advertise a list of DNS recursive server addresses and a DNS Search List to IPv6 hosts. Using RDNSS together with 'scoped' RAs as described above would allow a first-hop router (R3 in the Figure 3) to send DNS server addresses and search lists provided by each ISPs.

As discussed in Section 4.5.2, failure of all ISP uplinks would cause depreaction of all addresses assigned to a host from ISPs address space. Most likely intra-site IPv6 connectivity would be still desirable so Section 4.5.2 proposes a usage of ULAs to enable intra-site communication. In such scenario the enterprise network should run its own recursive DNS server(s) and provide its ULA addresses to hosts via RDNSS mechanism in RAs send for ULA-scoped forwarding table as described in Section 4.5.2.

It should be noted that [RFC6106] explicitly prohibits using DNS information if the RA router Lifetime expired: "An RDNSS address or a DNSSL domain name MUST be used only as long as both the RA router Lifetime (advertised by a Router Advertisement message) and the corresponding option Lifetime have not expired.". Therefore hosts might ignore RDNSS information provided in ULA-scoped RAs as those RAs would have router lifetime set to 0. However the updated version of RFC6106 ([I-D.ietf-6man-rdnss-rfc6106bis]) has that requirement removed.

5. Other Solutions

5.1. Shim6

The Shim6 working group specified the Shim6 protocol [RFC5533] which allows a host at a multihomed site to communicate with an external host and exchange information about possible source and destination address pairs that they can use to communicate. It also specified the REAP protocol [RFC5534] to detect failures in the path between working address pairs and find new working address pairs. A fundamental requirement for Shim6 is that both internal and external hosts need to support Shim6. That is, both the host internal to the multihomed site and the host external to the multihomed site need to support Shim6 in order for there to be any benefit for the internal host to run Shim6. The Shim6 protocol specification was published in 2009, but it has not been implemented on widely used operating systems.

We do not consider Shim6 to be a viable solution. It suffers from the fact that it requires widespread deployment of Shim6 on hosts all

over the Internet before the host at a PA multihomed site sees significant benefit. However, there appears to be no motivation for the vast majority of hosts on the Internet (which are not at PA multihomed sites) to deploy Shim6. This may help explain why Shim6 has not been widely implemented.

5.2. IPv6-to-IPv6 Network Prefix Translation

IPv6-to-IPv6 Network Prefix Translation (NPTv6) [RFC6296] is not the focus of this document. This document describes a solution where a host in a multihomed site determines which ISP a packet will be sent to based on the source address it applies to the packet. This solution has many moving parts. It requires some routers in the enterprise site to support some form of Source Address Dependent Routing (SADR). It requires a host to be able to learn when the uplink to an ISP fails so that it can stop using the source address corresponding to that ISP. Ongoing work to create mechanisms to accomplish this are discussed in this document, but they are still a work in progress.

This document attempts to create a PA multihoming solution that is as easy as possible for an enterprise to deploy. However, the success of this solution will depend greatly on whether or not the mechanisms for hosts to select source addresses based on the state of ISP uplinks gets implemented across a wide range of operating systems as the default mode of operation. Until that occurs, NPTv6 should still be considered a viable option to enable PA multihoming for enterprises.

6. IANA Considerations

This memo asks the IANA for no new parameters.

7. Security Considerations

7.1. Privacy Considerations

8. Acknowledgements

The original outline was suggested by Ole Troan.

9. References

9.1. Normative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[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>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[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>.

[RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, DOI 10.17487/RFC3582, August 2003, <http://www.rfc-editor.org/info/rfc3582>.

[RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, DOI 10.17487/RFC4116, July 2005, <http://www.rfc-editor.org/info/rfc4116>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <http://www.rfc-editor.org/info/rfc4191>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC4218] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, DOI 10.17487/RFC4218, October 2005, <http://www.rfc-editor.org/info/rfc4218>.

[RFC4219] Lear, E., "Things Multihoming in IPv6 (MULTI6) Developers Should Think About", RFC 4219, DOI 10.17487/RFC4219, October 2005, <http://www.rfc-editor.org/info/rfc4219>.

[RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, DOI 10.17487/RFC4242, November 2005, <http://www.rfc-editor.org/info/rfc4242>.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <http://www.rfc-editor.org/info/rfc6106>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <http://www.rfc-editor.org/info/rfc6296>.

[RFC7157] Troan, O., Ed., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation", RFC 7157, DOI 10.17487/RFC7157, March 2014, <http://www.rfc-editor.org/info/rfc7157>.

9.2. Informative References

[I-D.baker-ipv6-isis-dst-src-routing]
Baker, F. and D. Lamparter, "IPv6 Source/Destination Routing using IS-IS", draft-baker-ipv6-isis-dst-src-routing-06 (work in progress), October 2016.

[I-D.baker-rtgwg-src-dst-routing-use-cases]
Baker, F., Xu, M., Yang, S., and J. Wu, "Requirements and Use Cases for Source/Destination Routing", draft-baker-rtgwg-src-dst-routing-use-cases-02 (work in progress), April 2016.

[I-D.boutier-babel-source-specific]
Boutier, M. and J. Chroboczek, "Source-Specific Routing in Babel", draft-boutier-babel-source-specific-01 (work in progress), May 2015.

[I-D.huitema-shim6-ingress-filtering]
Huitema, C., "Ingress filtering compatibility for IPv6 multihomed sites", draft-huitema-shim6-ingress-filtering-00 (work in progress), September 2005.

[I-D.ietf-6man-multi-homed-host]
Baker, F. and B. Carpenter, "First-hop router selection by hosts in a multi-prefix network", draft-ietf-6man-multi-homed-host-10 (work in progress), October 2016.

[I-D.ietf-6man-rdnss-rfc6106bis]
Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", draft-ietf-6man-rdnss-rfc6106bis-14 (work in progress), June 2016.

[I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-11 (work in progress), March 2015.

[I-D.ietf-mptcp-experience]
Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", draft-ietf-mptcp-experience-07 (work in progress), October 2016.

[I-D.ietf-rtgwg-dst-src-routing]
Lamparter, D. and A. Smirnov, "Destination/Source Routing", draft-ietf-rtgwg-dst-src-routing-02 (work in progress), May 2016.

[I-D.pfister-6man-sadr-ra]
Pfister, P., "Source Address Dependent Route Information Option for Router Advertisements", draft-pfister-6man-sadr-ra-01 (work in progress), June 2015.

[I-D.xu-src-dst-bgp]
Xu, M., Yang, S., and J. Wu, "Source/Destination Routing Using BGP-4", draft-xu-src-dst-bgp-00 (work in progress), March 2016.

[PATRICIA]
Morrison, D., "Practical Algorithm to Retrieve Information Coded in Alphanumeric", Journal of the ACM 15(4) pp514-534, October 1968.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <http://www.rfc-editor.org/info/rfc3704>.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, DOI 10.17487/RFC3736, April 2004, <http://www.rfc-editor.org/info/rfc3736>.

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

[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>.

[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>.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, DOI 10.17487/RFC5533, June 2009, <http://www.rfc-editor.org/info/rfc5533>.

[RFC5534] Arkko, J. and I. van Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming", RFC 5534, DOI 10.17487/RFC5534, June 2009, <http://www.rfc-editor.org/info/rfc5534>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing Address Selection Policy Using DHCPv6", RFC 7078, DOI 10.17487/RFC7078, January 2014, <http://www.rfc-editor.org/info/rfc7078>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <http://www.rfc-editor.org/info/rfc7788>.

Appendix A. Change Log

Initial Version: July 2016

Authors' Addresses

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Chris Bowers
Juniper Networks
Sunnyvale, California 94089
USA

Email: cbowers@juniper.net

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com

ippm
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2018

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JPMC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
D. Bernier
Bell Canada

July 2, 2017

Data Fields for In-situ OAM
draft-brockners-inband-oam-data-07

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document discusses the data fields and associated data types for in-situ OAM. In-situ OAM data fields can be embedded into a variety of transports such as NSH, Segment Routing, Geneve, native IPv6 (via extension header), or IPv4. In-situ OAM can be used to complement OAM mechanisms based on e.g. ICMP or other types of probe packets.

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/.

Copyright Notice

Table of Contents

1. Introduction

This document defines data fields for "in-situ" Operations, Administration, and Maintenance (IOAM). In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. IOAM is to complement mechanisms such as Ping or Traceroute, or more recent active probing mechanisms as described in [I-D.lapukhov-dataplane-probe]. In terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, IOAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] IOAM could be portrayed as Hybrid Type 1. "In-situ" mechanisms do not require extra packets to be sent and hence don't change the packet traffic mix within the network. IOAM mechanisms can be leveraged where mechanisms using e.g. ICMP do not apply or do not offer the desired results, such as proving that a certain traffic flow takes a pre-defined path, SLA verification for the live data traffic, detailed statistics on traffic distribution paths in networks that distribute traffic across multiple paths, or scenarios in which probe traffic is potentially handled differently from regular data traffic by the network devices.

2. Conventions

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].

Abbreviations used in this document:

E2E Edge to Edge

Geneve: Generic Network Virtualization Encapsulation [I-D.ietf-nvo3-geneve]

IOAM: In-situ Operations, Administration, and Maintenance

MTU: Maximum Transmit Unit

NSH: Network Service Header [I-D.ietf-sfc-nsh]

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

SID: Segment Identifier

SR: Segment Routing

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension [I-D.ietf-nvo3-vxlan-gpe]

3. Scope, Applicability, and Assumptions

IOAM deployment assumes a set of constraints, requirements, and guiding principles which are described in this section.

Scope: This document defines the data fields and associated data types for in-situ OAM. The in-situ OAM data field can be transported by a variety of transport protocols, including NSH, Segment Routing, Geneve, IPv6, or IPv4. Specification details for these different transport protocols are outside the scope of this document.

Deployment domain (or scope) of in-situ OAM deployment: IOAM is a network domain focused feature, with "network domain" being a set of network devices or entities within a single administration. For example, a network domain can include an enterprise campus using physical connections between devices or an overlay network using virtual connections / tunnels for connectivity between said devices. A network domain is defined by its perimeter or edge. Designers of carrier protocols for IOAM must specify mechanisms to ensure that IOAM data stays within an IOAM domain. In addition, the operator of such a domain is expected to put provisions in place to ensure that IOAM data does not leak beyond the edge of an IOAM domain, e.g. using for example packet filtering methods. The operator should consider potential operational impact of IOAM to mechanisms such as ECMP processing (e.g. load-balancing schemes based on packet length could be impacted by the increased packet size due to IOAM), path MTU (i.e. ensure that the MTU of all links within a domain is sufficiently large to support the increased packet size due to IOAM) and ICMP message handling (i.e. in case of a native IPv6 transport, IOAM support for ICMPv6 Echo Request/Reply could desired which would translate into ICMPv6 extensions to enable IOAM data fields to be copied from an Echo Request message to an Echo Reply message).

IOAM control points: IOAM data fields are added to or removed from the live user traffic by the devices which form the edge of a domain.

Devices within an IOAM domain can update and/or add IOAM data-fields. Domain edge devices can be hosts or network devices.

Traffic-sets that IOAM is applied to: IOAM can be deployed on all or only on subsets of the live user traffic. It SHOULD be possible to enable IOAM on a selected set of traffic (e.g., per interface, based on an access control list or flow specification defining a specific set of traffic, etc.) The selected set of traffic can also be all traffic.

Encapsulation independence: Data formats for IOAM SHOULD be defined in a transport-independent manner. IOAM applies to a variety of encapsulating protocols. A definition of how IOAM data fields are carried by different transport protocols is outside the scope of this document.

Layering: If several encapsulation protocols (e.g., in case of tunneling) are stacked on top of each other, IOAM data-records could be present at every layer. The behavior follows the ships-in-the-night model.

Combination with active OAM mechanisms: IOAM should be usable for active network probing, enabling for example a customized version of traceroute. Decapsulating IOAM nodes may have an ability to send the IOAM information retrieved from the packet back to the source address of the packet or to the encapsulating node.

IOAM implementation: The IOAM data-field definitions take the specifics of devices with hardware data-plane and software data-plane into account.

4. IOAM Data Types and Formats

This section defines IOAM data types and data fields and associated data types required for IOAM. The different uses of IOAM require the definition of different types of data. The IOAM data fields for the data being carried corresponds to the three main categories of IOAM data defined in [I-D.brockners-inband-oam-requirements], which are: edge-to-edge, per node, and for selected nodes only.

Transport options for IOAM data are outside the scope of this memo, and are discussed in [I-D.brockners-inband-oam-transport]. IOAM data fields are fixed length data fields. A bit field determines the set of OAM data fields embedded in a packet. Depending on the type of the encapsulation, a counter field indicates how many data fields are included in a particular packet.

IOAM is expected to be deployed in a specific domain rather than on the overall Internet. The part of the network which employs IOAM is referred to as the "IOAM-domain". IOAM data is added to a packet upon entering the IOAM-domain and is removed from the packet when exiting the domain. Within the IOAM-domain, the IOAM data may be updated by network nodes that the packet traverses. The device which adds an IOAM data container to the packet to capture IOAM data is called the "IOAM encapsulating node", whereas the device which removes the IOAM data container is referred to as the "IOAM decapsulating node". Nodes within the domain which are aware of IOAM data and read and/or write or process the IOAM data are called "IOAM transit nodes". IOAM nodes which add or remove the IOAM data container can also update the IOAM data fields at the same time. Or in other words, IOAM encapsulation or decapsulating nodes can also serve as IOAM transit nodes at the same time. Note that not every node in an IOAM domain needs to be an IOAM transit node. For example, a Segment Routing deployment might require the segment routing path to be verified. In that case, only the SR nodes would also be IOAM transit nodes rather than all nodes.

4.1. IOAM Tracing Options

"IOAM tracing data" is expected to be collected at every node that a packet traverses to ensure visibility into the entire path a packet takes within an IOAM domain, i.e., in a typical deployment all nodes in an in-situ OAM-domain would participate in IOAM and thus be IOAM transit nodes, IOAM encapsulating or IOAM decapsulating nodes. If not all nodes within a domain are IOAM capable, IOAM tracing information will only be collected on those nodes which are IOAM capable. Nodes which are not IOAM capable will forward the packet without any changes to the IOAM data fields. The maximum number of hops and the minimum path MTU of the IOAM domain is assumed to be known.

To optimize hardware and software implementations tracing is defined as two separate options. Any deployment MAY choose to configure and support one or both of the following options. An implementation of the transport protocol that carries these in-situ OAM data MAY choose to support only one of the options. In the event that both options are utilized at the same time, the Incremental Trace Option MUST be placed before the Pre-allocated Trace Option. Given that the operator knows which equipment is deployed in a particular IOAM, the operator will decide by means of configuration which type(s) of trace options will be enabled for a particular domain.

Pre-allocated Trace Option: This trace option is defined as a container of node data fields with pre-allocated space for each node to populate its information. This option is useful for

software implementations where it is efficient to allocate the space once and index into the array to populate the data during transit. The IOAM encapsulating node allocates the option header and sets the fields in the option header. The in situ OAM encapsulating node allocates an array which is used to store operational data retrieved from every node while the packet traverses the domain. IOAM transit nodes update the content of the array. A pointer which is part of the IOAM trace data points to the next empty slot in the array, which is where the next IOAM transit node fills in its data.

Incremental Trace Option: This trace option is defined as a container of node data fields where each node allocates and pushes its node data immediately following the option header. The maximum length of the node data list is written into the option header. This type of trace recording is useful for some of the hardware implementations as this eliminates the need for the transit network elements to read the full array in the option and allows for arbitrarily long packets as the MTU allows. The in-situ OAM encapsulating node allocates the option header. The in-situ OAM encapsulating node based on operational state and configuration sets the fields in the header to control how large the node data list can grow. IOAM transit nodes push their node data to the node data list and increment the number of node data fields in the header.

Every node data entry is to hold information for a particular IOAM transit node that is traversed by a packet. The in-situ OAM decapsulating node removes the IOAM data and processes and/or exports the metadata. IOAM data uses its own name-space for information such as node identifier or interface identifier. This allows for a domain-specific definition and interpretation. For example: In one case an interface-id could point to a physical interface (e.g., to understand which physical interface of an aggregated link is used when receiving or transmitting a packet) whereas in another case it could refer to a logical interface (e.g., in case of tunnels).

The following IOAM data is defined for IOAM tracing:

- Identification of the IOAM node. An IOAM node identifier can match to a device identifier or a particular control point or subsystem within a device.

- Identification of the interface that a packet was received on, i.e. ingress interface.

- Identification of the interface that a packet was sent out on, i.e. egress interface.

- Time of day when the packet was processed by the node. Different definitions of processing time are feasible and expected, though it is important that all devices of an in-situ OAM domain follow the same definition.

- Generic data: Format-free information where syntax and semantic of the information is defined by the operator in a specific deployment. For a specific deployment, all IOAM nodes should interpret the generic data the same way. Examples for generic IOAM data include geo-location information (location of the node at the time the packet was processed), buffer queue fill level or cache fill level at the time the packet was processed, or even a battery charge level.

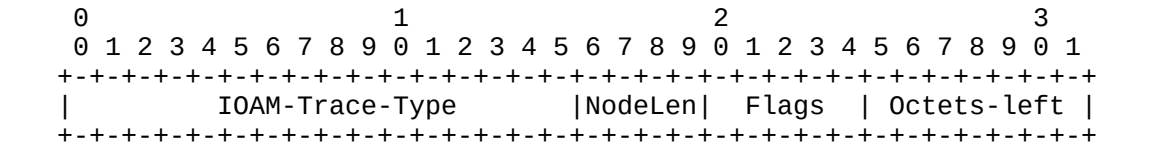
- A mechanism to detect whether IOAM trace data was added at every hop or whether certain hops in the domain weren't in-situ OAM transit nodes.

The "node data list" array in the packet is populated iteratively as the packet traverses the network, starting with the last entry of the array, i.e., "node data list [n]" is the first entry to be populated, "node data list [n-1]" is the second one, etc.

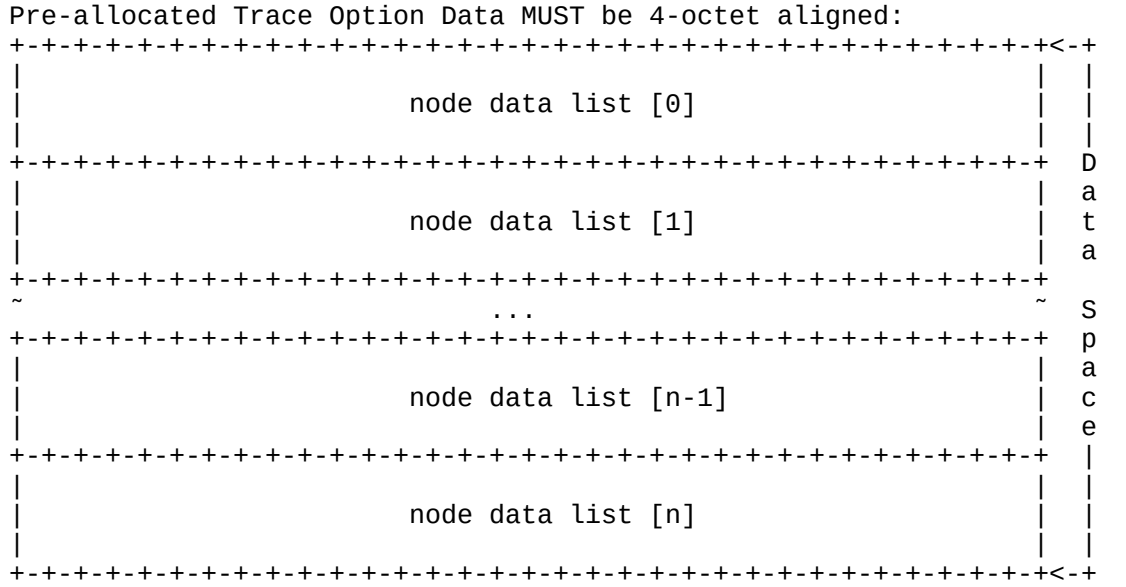
4.1.1. Pre-allocated Trace Option

In-situ OAM pre-allocated trace option:

Pre-allocated trace option header:



Pre-allocated Trace Option Data MUST be 4-octet aligned:



IOAM-Trace-Type: A 16-bit identifier which specifies which data types are used in this node data list.

The IOAM-Trace-Type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 4.1.3. The order of packing the data fields in each node data element follows the bit order of the IOAM-Trace-Type field, as follows:

Bit 0 (Most significant bit) When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1 When set indicates presence of ingress_if_id and egress_if_id (short format) in the node data.

Bit 2 When set indicates presence of timestamp seconds in the node data

Bit 3 When set indicates presence of timestamp nanoseconds in the node data.

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data (short format) in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id in wide format in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.

Bit 10 When set indicates presence of app_data wide in the node data.

Bit 11 When set indicates presence of the Checksum Complement node data.

Bit 12-15 Undefined in this draft.

Section 4.1.3 describes the IOAM data types and their formats. Within an in-situ OAM domain possible combinations of these bits making the IOAM-Trace-Type can be restricted by configuration knobs.

Node Data Length: 4-bit unsigned integer. This field specifies the length of data added by each node in multiples of 4-octets. For example, if 3 IOAM-Trace-Type bits are set and none of them is wide, then the Node Data Length would be 3. If 3 IOAM-Trace-Type bits are set and 2 of them are wide, then the Node Data Length would be 5.

Flags 5-bit field. Following flags are defined:

Bit 0 "Overflow" (O-bit) (most significant bit). This bit is set by the network element if there is not enough number of octets

left to record node data, no field is added and the overflow "O-bit" must be set to "1" in the header. This is useful for transit nodes to ignore further processing of the option.

Bit 1 "Loopback" (L-bit). Loopback mode is used to send a copy of a packet back towards the source. Loopback mode assumes that a return path from transit nodes and destination nodes towards the source exists. The encapsulating node decides (e.g. using a filter) which packets loopback mode is enabled for by setting the loopback bit. The encapsulating node also needs to ensure that sufficient space is available in the IOAM header for loopback operation. The loopback bit when set indicates to the transit nodes processing this option to create a copy of the packet received and send this copy of the packet back to the source of the packet while it continues to forward the original packet towards the destination. The source address of the original packet is used as destination address in the copied packet. The address of the node performing the copy operation is used as the source address. The L-bit MUST be cleared in the copy of the packet a nodes sends it back towards the source. On its way back towards the source, the packet is processed like a regular packet with IOAM information. Once the return packet reaches the IOAM domain boundary IOAM decapsulation occurs as with any other packet containing IOAM information.

Bit 2-4 Reserved: Must be zero.

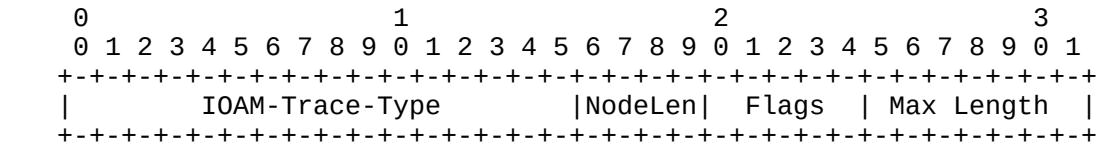
Octets-left: 7-bit unsigned integer. It is the data space in multiples of 4-octets remaining for recording the node data. This is used as an offset in data space to record the node data element.

Node data List [n]: Variable-length field. The type of which is determined by the IOAM-Trace-Type representing the n-th node data in the node data list. The node data list is encoded starting from the last node data of the path. The first element of the node data list (node data list [0]) contains the last node of the path while the last node data of the node data list (node data list[n]) contains the first node data of the path traced. The index contained in "Octets-left" identifies the offset for current active node data to be populated.

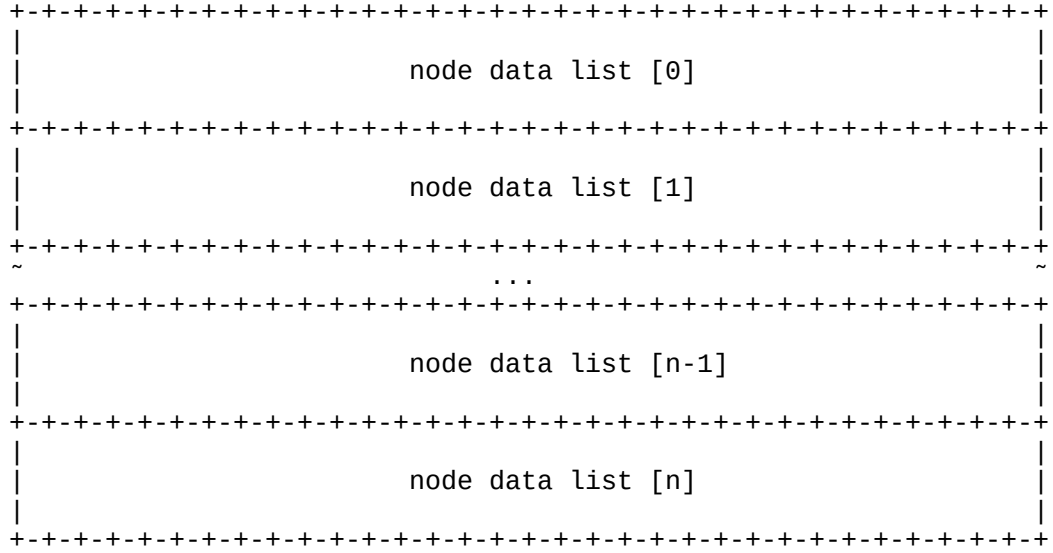
4.1.2. Incremental Trace Option

In-situ OAM incremental trace option:

In-situ OAM incremental trace option Header:



IOAM Incremental Trace Option Data MUST be 4-octet aligned:



IOAM-trace-type: A 16-bit identifier which specifies which data types are used in this node data list.

The IOAM-Trace-Type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 4.1.3. The order of packing the data fields in each node data element follows the bit order of the IOAM-Trace-Type field, as follows:

Bit 0 (Most significant bit) When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1 When set indicates presence of ingress_if_id and egress_if_id (short format) in the node data.

Bit 2 When set indicates presence of timestamp seconds in the node data.

Bit 3 When set indicates presence of timestamp nanoseconds in the node data.

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id wide in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.

Bit 10 When set indicates presence of app_data wide in the node data.

Bit 11 When set indicates presence of the Checksum Complement node data.

Bit 12-15 Undefined in this draft.

Section 4.1.3 describes the IOAM data types and their formats.

Node Data Length: 4-bit unsigned integer. This field specifies the length of data added by each node in multiples of 4-octets. For example, if 3 IOAM-Trace-Type bits are set and none of them is wide, then the Node Data Length would be 3. If 3 IOAM-Trace-Type bits are set and 2 of them are wide, then the Node Data Length would be 5.

Flags 5-bit field. Following flags are defined:

Bit 0 "Overflow" (O-bit) (least significant bit). This bit is set by the network element if there is not enough number of octets left to record node data, no field is added and the overflow "O-bit" must be set to "1" in the header. This is useful for transit nodes to ignore further processing of the option.

Bit 1 "Loopback" (L-bit). This bit when set indicates to the transit nodes processing this option to send a copy of the packet back to the source of the packet while it continues to forward the original packet towards the destination. The L-bit MUST be cleared in the copy of the packet before sending it.

Bit 2-4 Reserved. Must be zero.

Maximum Length: 7-bit unsigned integer. This field specifies the maximum length of the node data list in multiples of 4-octets. Given that the sender knows the minimum path MTU, the sender can set the maximum length according to the number of node data bytes allowed before exceeding the MTU. Thus, a simple comparison between "Opt data Len" and "Max Length" allows to decide whether or not data could be added.

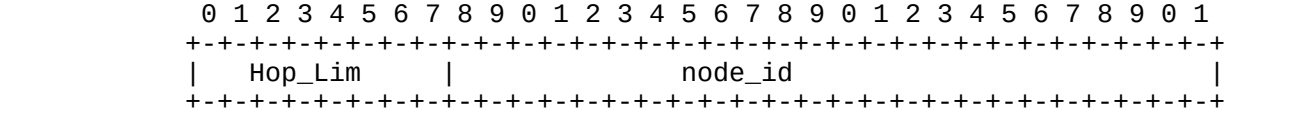
Node data List [n]: Variable-length field. The type of which is determined by the OAM Type representing the n-th node data in the node data list. The node data list is encoded starting from the last node data of the path. The first element of the node data list (node data list [0]) contains the last node of the path while the last node data of the node data list (node data list[n]) contains the first node data of the path traced.

4.1.3. IOAM node data fields and associated formats

All the data fields MUST be 4-octet aligned. The IOAM encapsulating node MUST initialize data fields that it adds to the packet to zero. If a node which is supposed to update an IOAM data field is not capable of populating the value of a field set in the IOAM-Trace-Type, the field value MUST be left unaltered except when explicitly specified in the field description below. In the description of data below if zero is valid value then a non-zero value to mean not populated is specified.

Data field and associated data type for each of the data field is shown below:

Hop_Lim and node_id: 4-octet field defined as follows:

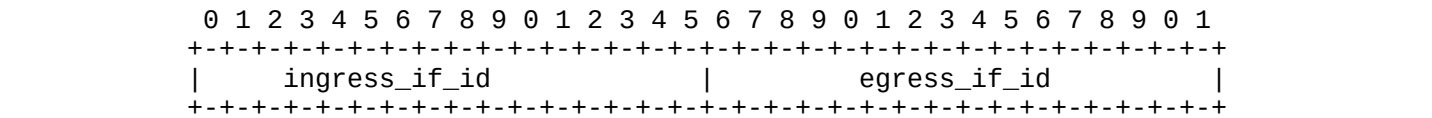


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node

in the communication path. This is copied from the lower layer, e.g., TTL value in IPv4 header or hop limit field from IPv6 header of the packet when the packet is ready for transmission. The semantics of the Hop_Lim field depend on the lower layer protocol that IOAM is encapsulated over, and therefore its specific semantics are outside the scope of this memo.

node_id: 3-octet unsigned integer. Node identifier field to uniquely identify a node within in-situ OAM domain. The procedure to allocate, manage and map the node_ids is beyond the scope of this document.

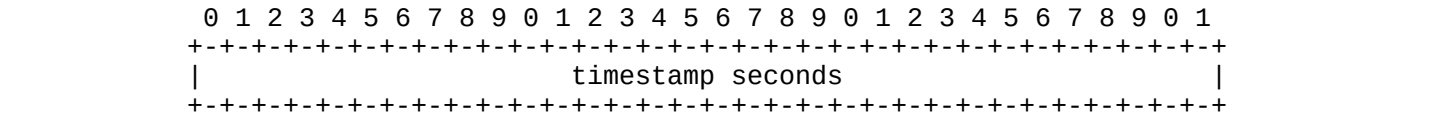
ingress_if_id and egress_if_id: 4-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the position in the field must be filled with value 0xFFFFFFFF to mean not populated.



ingress_if_id: 2-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

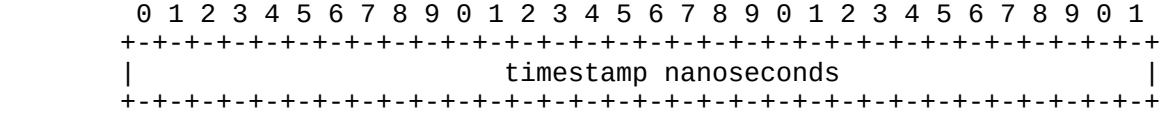
egress_if_id: 2-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

timestamp seconds: 4-octet unsigned integer. Absolute timestamp in seconds that specifies the time at which the packet was received by the node. The structure of this field is identical to the most significant 32 bits of the 64 least significant bits of the [IEEE1588v2] timestamp. This truncated field consists of a 32-bit seconds field. As defined in [IEEE1588v2], the timestamp specifies the number of seconds elapsed since 1 January 1970 00:00:00 according to the International Atomic Time (TAI).

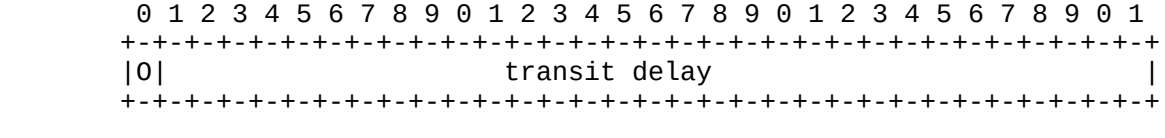


timestamp nanoseconds: 4-octet unsigned integer in the range 0 to 10^9-1. This timestamp specifies the fractional part of the wall clock time at which the packet was received by the node in units of nanoseconds. This field is identical to the 32 least significant bits of the [IEEE1588v2] timestamp. This fields

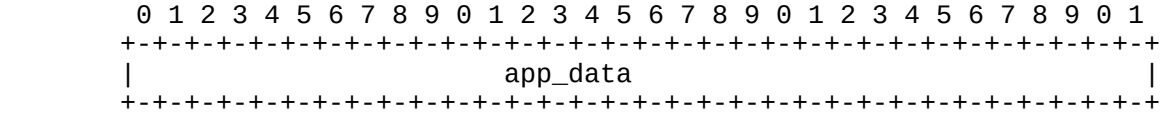
allows for delay computation between any two nodes in the network when the nodes are time synchronized. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.



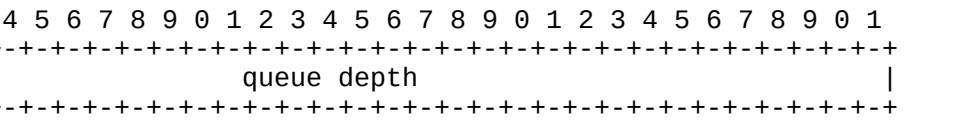
transit delay: 4-octet unsigned integer in the range 0 to 2^30-1. It is the time in nanoseconds the packet spent in the transit node. This can serve as an indication of the queuing delay at the node. If the transit delay exceeds 2^30-1 nanoseconds then the top bit 'O' is set to indicate overflow. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.



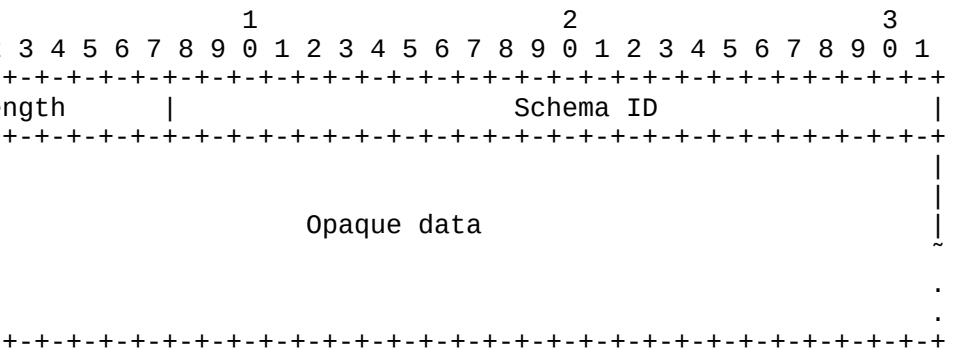
app_data: 4-octet placeholder which can be used by the node to add application specific data. App_data represents a "free-format" 4-octet bit field with its semantics defined by a specific deployment.



queue depth: 4-octet unsigned integer field. This field indicates the current length of the egress interface queue of the interface from where the packet is forwarded out. The queue depth is expressed as the current number of memory buffers used by the queue (a packet may consume one or more memory buffers, depending on its size). When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.



Opaque State Snapshot: Variable length field. It allows the network element to store an arbitrary state in the node data field , without a pre-defined schema. The schema needs to be made known to the analyzer by some out-of-band mechanism. The specification of this mechanism is beyond the scope of this document. The 24-bit "Schema Id" field in the field indicates which particular schema is used, and should be configured on the network element by the operator.

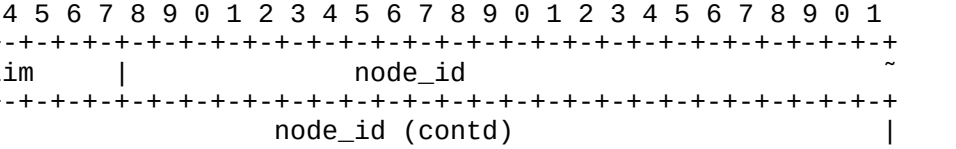


Length: 1-octet unsigned integer. It is the length in octets of the Opaque data field that follows Schema Id. It MUST always be a multiple of 4.

Schema ID: 3-octet unsigned integer identifying the schema of Opaque data.

Opaque data: Variable length field. This field is interpreted as specified by the schema identified by the Schema ID.

Hop_Lim and node_id wide: 8-octet field defined as follows:

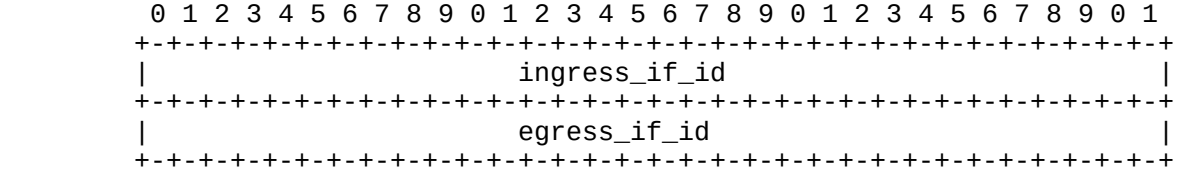


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop

Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer for e.g. TTL value in IPv4 header or hop limit field from IPv6 header of the packet. The semantics of the Hop_Lim field depend on the lower layer protocol that IOAM is encapsulated over, and therefore its specific semantics are outside the scope of this memo.

node_id: 7-octet unsigned integer. Node identifier field to uniquely identify a node within in-situ OAM domain. The procedure to allocate, manage and map the node_ids is beyond the scope of this document.

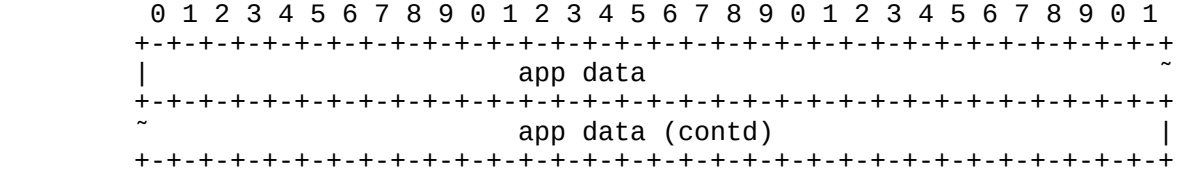
ingress_if_id and egress_if_id wide: 8-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFFFFFFFFFF to mean not populated.



ingress_if_id: 4-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

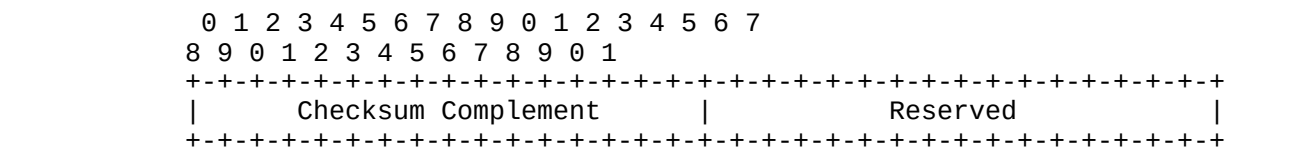
egress_if_id: 4-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

app_data wide: 8-octet placeholder which can be used by the node to add application specific data. App data represents a "free-format" 8-octed bit field with its semantics defined by a specific deployment.



Checksum Complement: 4-octet node data which contains a two-octet Checksum Complement field, and a 2-octet reserved field. The Checksum Complement can be used when IOAM is transported over encapsulations that make use of a UDP transport, such as VXLAN-GPE

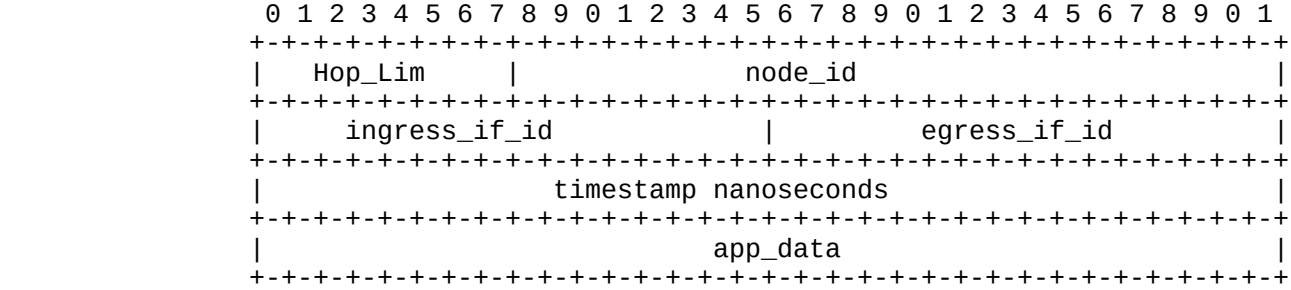
or Geneve. In this case, incorporating the IOAM node data requires the UDP Checksum field to be updated. Rather than to recompute the Chekcsum field, a node can use the Checksum Complement to make a checksum-neutral update in the UDP payload; the Checksum Complement is assigned a value that complements the rest of the node data fields that were added by the current node, causing the existing UDP Checksum field to remain correct. Checksum Complement fields are used in a similar manner in [RFC7820] and [RFC7821].



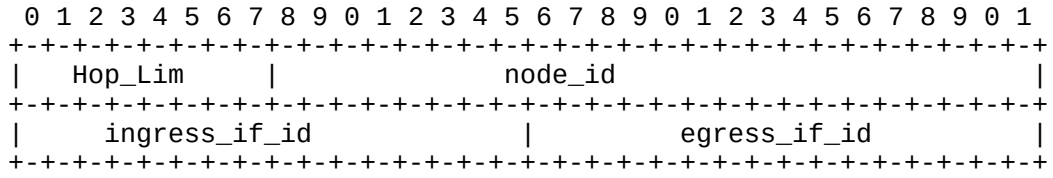
4.1.4. Examples of IOAM node data

An entry in the "node data list" array can have different formats, following the needs of the deployment. Some deployments might only be interested in recording the node identifiers, whereas others might be interested in recording node identifier and timestamp. The section defines different types that an entry in "node data list" can take.

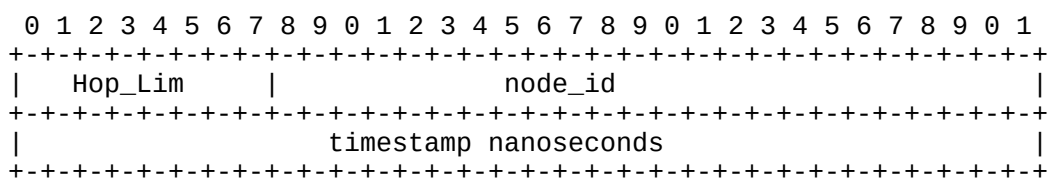
0x002B: IOAM-Trace-Type is 0x2B then the format of node data is:



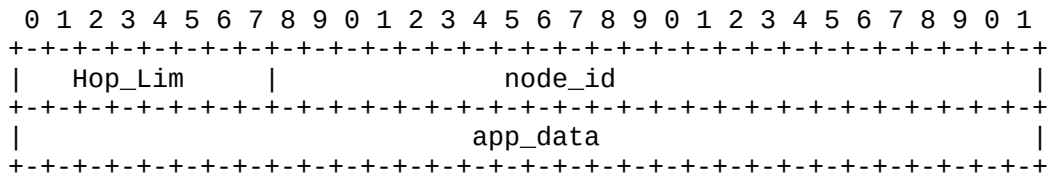
0x0003: IOAM-Trace-Type is 0x0003 then the format is:



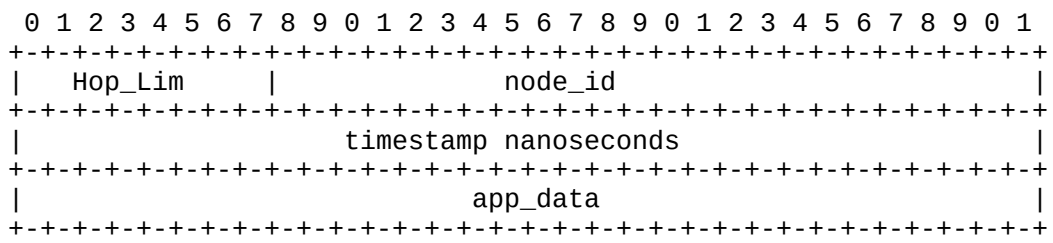
0x0009: IOAM-Trace-Type is 0x0009 then the format is:



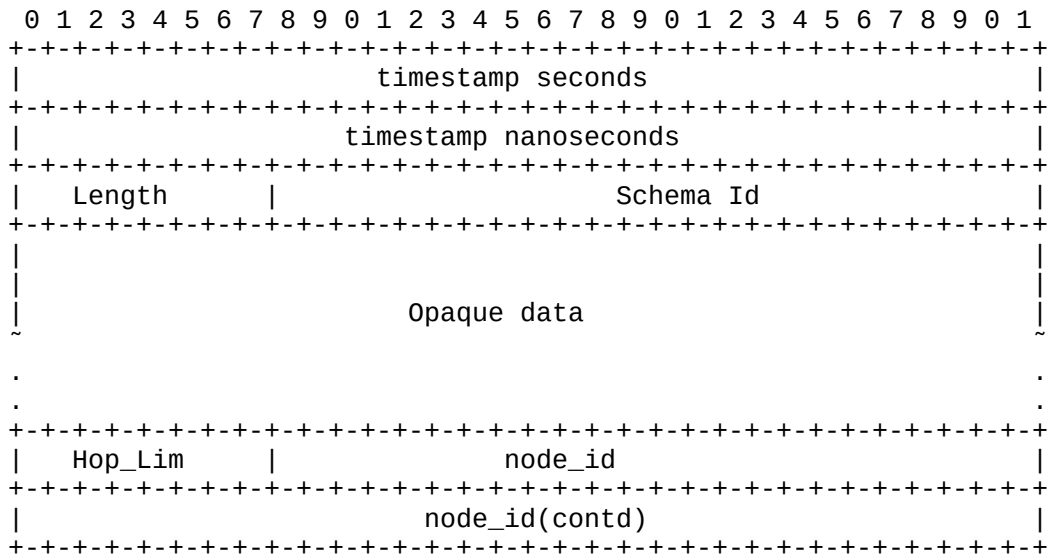
0x0021: IOAM-Trace-Type is 0x0021 then the format is:



0x0029: IOAM-Trace-Type is 0x0029 then the format is:



0x018C: IOAM-Trace-Type is 0x104D then the format is:



4.2. IOAM Proof of Transit Option

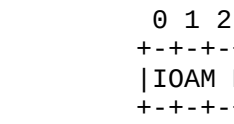
IOAM Proof of Transit data is to support the path or service function chain [RFC7665] verification use cases. Proof-of-transit uses methods like nested hashing or nested encryption of the IOAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the IOAM data for the proof of transit option is processed at IOAM encapsulating, decapsulating and transit nodes are outside the scope of the document, all of these approaches share the need to uniquely identify a packet as well as iteratively operate on a set of information that is handed from node to node. Correspondingly, two pieces of information are added as IOAM data to the packet:

- Random: Unique identifier for the packet (e.g., 64-bits allow for the unique identification of 2^64 packets).

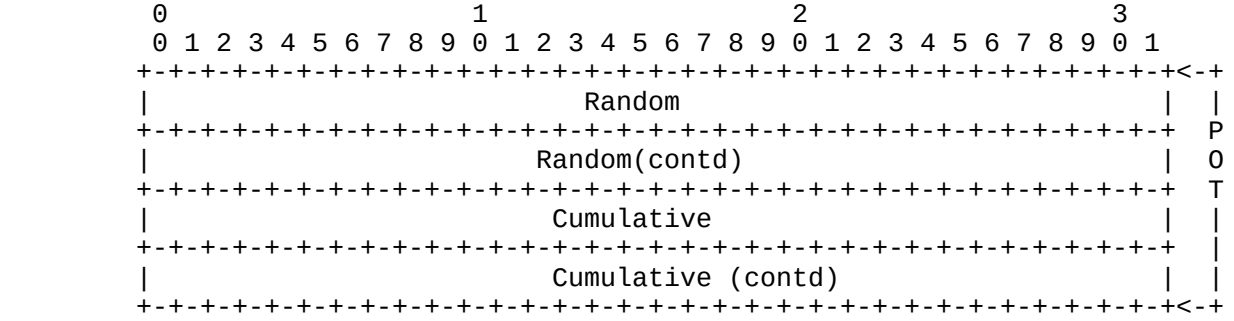
- Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

IOAM proof of transit option:

IOAM proof of transit option header:



IOAM proof of transit option data MUST be 4-octet aligned:



IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included. This document defines POT Type 0:

0: POT data is a 16 Octet field as described below.

Profile to use (P): 1-bit. Indicates which POT-profile is used to generate the Cumulative. Any node participating in POT will have a maximum of 2 profiles configured that drive the computation of cumulative. The two profiles are numbered 0, 1. This bit conveys whether profile 0 or profile 1 is used to compute the Cumulative.

Random: 64-bit Per packet Random number.

Cumulative: 64-bit Cumulative that is updated at specific nodes by processing per packet Random number field and configured parameters.

Note: Larger or smaller sizes of "Random" and "Cumulative" data are feasible and could be required for certain deployments (e.g. in case of space constraints in the transport protocol used). Future versions of this document will address different sizes of data for "proof of transit".

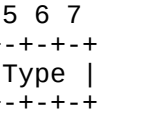
4.3. IOAM Edge-to-Edge Option

The IOAM edge-to-edge option is to carry data that is added by the IOAM encapsulating node and interpreted by IOAM decapsulating node. The IOAM transit nodes MAY process the data without modifying it.

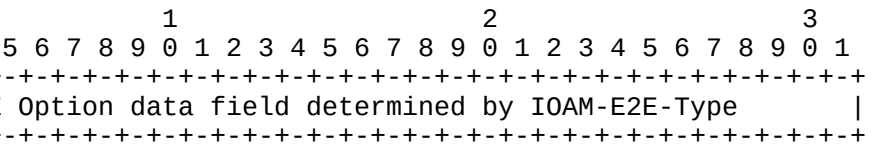
Currently only sequence numbers use the IOAM edge-to-edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-situ OAM-domain, sequence numbers can be added to packets of a particular tube (see [I-D.hildebrand-spud-prototype]). Each tube leverages a dedicated namespace for its sequence numbers.

IOAM edge-to-edge option:

IOAM edge-to-edge option header:



IOAM edge-to-edge option data MUST be 4-octet aligned:



IOAM-E2E-Type: 8-bit identifier of a particular in situ OAM E2E variant.

0: E2E option data is a 64-bit sequence number added to a specific tube which is used to identify packet loss and reordering for that tube.

5. IOAM Data Export

IOAM nodes collect information for packets traversing a domain that supports IOAM. IOAM decapsulating nodes as well as IOAM transit nodes can choose to retrieve IOAM information from the packet, process the information further and export the information using e.g., IPFIX.

The discussion of IOAM data processing and export is left for a future version of this document.

6. IANA Considerations

This document requests the following IANA Actions.

6.1. Creation of a New In-Situ OAM (IOAM) Protocol Parameters IANA registry

IANA is requested to create a new protocol registry for "In-Situ OAM (IOAM) Protocol Parameters". This is the common registry that will include registrations for all IOAM namespaces. Each Registry, whose names are listed below:

IOAM Trace Type

IOAM Trace flags

IOAM POT Type

IOAM E2E Type

will contain the current set of possibilities defined in this document. New registries in this name space are created via RFC Required process as per [RFC8126].

The subsequent sub-sections detail the registries herein contained.

6.2. IOAM Trace Type Registry

This registry defines code point for each bit in the 16-bit IOAM-Trace-Type field for Pre-allocated trace option and Incremental trace option defined in Section 4.1. The meaning of Bit 0 - 11 for trace type are defined in this document in Paragraph 1 of (Section 4.1.1). The meaning for Bit 12 - 15 are available for assignment via RFC Required process as per [RFC8126].

6.3. IOAM Trace Flags Registry

This registry defines code point for each bit in the 5 bit flags for Pre-allocated trace option and Incremental trace option defined in Section 4.1. The meaning of Bit 0 - 1 for trace flags are defined in this document in Paragraph 5 of Section 4.1.1. The meaning for Bit 2 - 4 are available for assignment via RFC Required process as per [RFC8126].

6.4. IOAM POT Type Registry

This registry defines 128 code points to define IOAM POT Type for IOAM proof of transit option Section 4.2. The code point value 0 is defined in this document, 1 - 127 are available for assignment via RFC Required process as per [RFC8126].

6.5. IOAM E2E Type Registry

This registry defines 256 code points to define IOAM-E2E-Type for IOAM E2E option Section 4.3. The code point value 0 is defined in this document, 1 - 255 are available for assignments via RFC Required process as per [RFC8126].

7. Manageability Considerations

Manageability considerations will be addressed in a later version of this document..

8. Security Considerations

Security considerations will be addressed in a later version of this document. For a discussion of security requirements of in-situ OAM, please refer to [I-D.brockners-inband-oam-requirements].

9. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, LJ Wobker, Erik Nordmark, Vengada Prasad Govindan, and Andrew Yourtchenko for the comments and advice.

This document leverages and builds on top of several concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

The authors would like to gracefully acknowledge useful review and insightful comments received from Joe Clarke, Al Morton, and Mickey Spiegel.

10. References

10.1. Normative References

[IEEE1588v2]
Institute of Electrical and Electronics Engineers, "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, 2008, <http://standards.ieee.org/findstds/standard/1588-2008.html>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <http://www.rfc-editor.org/info/rfc8126>.

10.2. Informative References

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., and R. <>, "Encapsulations for In-situ OAM Data", draft-brockners-inband-oam-transport-03 (work in progress), March 2017.

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-04 (work in progress), March 2017.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-13 (work in progress), June 2017.

[I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.

[I-D.lapukhov-dataplane-probe]
Lapukhov, P. and r. remy@barefootnetworks.com, "Data-plane probe for in-band telemetry collection", draft-lapukhov-dataplane-probe-01 (work in progress), June 2016.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

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

[RFC7820] Mizrahi, T., "UDP Checksum Complement in the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7820, DOI 10.17487/RFC7820, March 2016, <http://www.rfc-editor.org/info/rfc7820>.

[RFC7821] Mizrahi, T., "UDP Checksum Complement in the Network Time Protocol (NTP)", RFC 7821, DOI 10.17487/RFC7821, March 2016, <http://www.rfc-editor.org/info/rfc7821>.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 2066721
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Petr Lapukhov
Facebook
1 Hacker Way
Menlo Park, CA 94025
US

Email: petr@fb.com

Remy Chang
Barefoot Networks
2185 Park Boulevard
Palo Alto, CA 94306
US

Daniel
Bell Canada

Email: daniel.bernier@bell.ca

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

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
D. Mozes
Mellanox Technologies Ltd.
T. Mizrahi
Marvell
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
March 13, 2017

Requirements for In-situ OAM
draft-brockners-inband-oam-requirements-03

Abstract

This document discusses the motivation and requirements for including specific operational and telemetry information into data packets while the data packet traverses a path between two points in the network. This method is referred to as "in-situ" Operations, Administration, and Maintenance (OAM), given that the OAM information is carried with the data packets as opposed to in "out-of-band" packets dedicated to OAM. In situ OAM complements other OAM mechanisms which use dedicated probe packets to convey OAM information.

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/.

Copyright Notice

Table of Contents

1. Introduction

This document discusses requirements for "in-situ" Operations, Administration, and Maintenance (OAM) mechanisms. In this context, "in-situ OAM" refers to the concept of directly encoding telemetry information within the data packet as it traverses the network or telemetry domain. Mechanisms which add tracing or other types of telemetry information to the regular data traffic, sometimes also referred to as "in-band" OAM can complement active, probe-based mechanisms such as ping or traceroute, which are sometimes considered as "out-of-band", because the messages are transported independently from regular data traffic. In terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, in-situ OAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] in-situ OAM could be portrayed as "hybrid OAM, type 1". "In-situ" mechanisms do not require extra packets to be sent and hence don't change the packet traffic mix within the network. Traceroute and ping for example use ICMP messages: New packets are injected to get tracing information. Those add to the number of messages in a network, which already might be highly loaded or suffering performance issues for a particular path or traffic type.

A number of in-situ as well as in-band OAM mechanisms have been discussed, such as the INT spec for the P4 programming language [P4] or the SPUD prototype [I-D.hildebrand-spud-prototype]. The SPUD prototype uses a similar logic that allows network devices on the path between endpoints to participate explicitly in the tube outside the end-to-end context. Even the IPv4 route-record option defined in [RFC0791] can be considered an in-situ OAM mechanism. Per what was already stated, in-situ OAM complements "out-of-band" mechanisms such as ping or traceroute, or more recent active probing mechanisms, as described in [I-D.lapukhov-dataplane-probe]. In-situ OAM mechanisms can be leveraged where current out-of-band mechanisms do not apply or do not offer the desired characteristics or requirements, such as

proving that a certain set of traffic takes a pre-defined path, strict congruency between overlay and underlay transports is in place, checking service level agreements for the live data traffic, detailed statistics or verification of path selections within a domain, or scenarios where probe traffic is potentially handled differently from regular data traffic by the network devices. [RFC7276] presents an overview of OAM tools.

Compared to probably the most basic example of "in-situ OAM" which is IPv4 route recording [RFC0791], an in-situ OAM approach has the following capabilities:

a. A flexible data format to allow different types of information to be captured as part of an in-situ OAM operation, including but not limited to path tracing information, operational and telemetry information such as timestamps, sequence numbers, or even generic data such as queue size, geo-location of the node that forwarded the packet, etc.

b. A data format to express node as well as link identifiers to record the path a packet takes with a fixed amount of added data.

c. The ability to determine whether any nodes were skipped while recording in-situ OAM information (i.e., in-situ OAM is not supported or not enabled on those nodes).

d. The ability to actively process information in the packet, for example to prove in a cryptographically secure way that a packet really took a pre-defined path using some traffic steering method such as service chaining or traffic engineering.

e. The ability to include OAM data beyond simple path information, such as timestamps or even generic data of a particular use case.

f. The ability to carry in-situ OAM data in various different transport protocols.

2. Conventions

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].

Abbreviations used in this document:

ECMP: Equal Cost Multi-Path

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

MTU: Maximum Transmit Unit

NSH: Network Service Header

NFV: Network Function Virtualization

OAM: Operations, Administration, and Maintenance

PMTU: Path MTU

SFC: Service Function Chain

SLA: Service Level Agreement

SR: Segment Routing

SID: Segment Identifier

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension

This document defines in-situ Operations, Administration, and Maintenance (in-situ OAM), as the subset in which OAM information is carried along with data packets. This is as opposed to "out-of-band OAM", where specific packets are dedicated to carrying OAM information.

3. Motivation for in-situ OAM

In several scenarios it is beneficial to make information about the path a packet took through the network or through a network device as well as associated telemetry information available to the operator. This includes not only tasks like debugging, troubleshooting, as well as network planning and network optimization but also policy or service level agreement compliance checks. This section discusses the motivation to introduce new methods for enhanced in-situ network diagnostics.

3.1. Path Congruency Issues with Dedicated OAM Packets

Packet scheduling algorithms, especially for balancing traffic across equal cost paths or links, often leverage information contained within the packet, such as protocol number, IP-address or MAC-address. Probe packets would thus either need to be sent from the exact same endpoints with the exact same parameters, or probe packets would need to be artificially constructed as "fake" packets and

inserted along the path. Both approaches are often not feasible from an operational perspective, be it that access to the end-system is not feasible, or that the diversity of parameters and associated probe packets to be created is simply too large. An in-situ mechanism is an alternative in those cases.

In-situ mechanisms are not impacted by differences in the handling of probe traffic compared to other data packets, where probe traffic is handled differently (and potentially forwarded differently) by a router than regular data traffic. This obviously assumes that the addition of in-situ information does not change the forwarding behavior of the packet. Note that in certain implementations, the addition information to a transport protocol changes the forwarding behavior. IPv6 extension header processing is one example. Some implementations process IPv6 packets with extension headers in the "slow" path of a router, as opposed to the "fast" path.

3.2. Results Sent to a System Other Than the Sender

Traditional ping and traceroute tools return the OAM results to the sender of the probe. Even when the ICMP messages that are used with these tools are enhanced, and additional telemetry is collected (e.g., ICMP Multi-Part [RFC4884] supporting MPLS information [RFC4950], Interface and Next-Hop Identification [RFC5837], etc.), it would be advantageous to separate the sending of an OAM probe from the receiving of the telemetry data. In this context, it is helpful to eliminate the requirement that there be a working bidirectional path.

3.3. Overlay and Underlay Correlation

Several network deployments leverage tunneling mechanisms to create overlay or service-layer networks. Examples include VXLAN-GPE, GRE, or LISP. One often observed attribute of overlay networks is that they do not offer the user of the overlay any insight into the underlay network. This means that the path that a particular tunneled packet takes, nor other operational details such as the per-hop delay/jitter in the underlay are visible to the user of the overlay network, giving rise to diagnosis and debugging challenges in case of connectivity or performance issues. The scope of OAM tools like ping or traceroute is limited to either the overlay or the underlay which means that the user of the overlay has typically no access to OAM in the underlay, unless specific operational procedures are put in place. With in-situ OAM the operator of the underlay can offer details of the connectivity in the underlay to the user of the overlay. This could include the ability to find out which underlay elements are shared by overlays and ability to know which overlays are mapped to the same underlay elements. Deployment dependent

underlay transit nodes can be configured to update OAM information in the overlay transport encapsulation. The operator of the egress tunnel router could choose to share the recorded information about the path with the user of the overlay.

Coupled with mechanisms such as Segment Routing (SR) [I-D.ietf-spring-segment-routing], overlay network and underlay network can be more tightly coupled: The user of the overlay has detailed diagnostic information available in case of failure conditions. The user of the overlay can also use the path recording information as input to traffic steering or traffic engineering mechanisms, to for example achieve path symmetry for the traffic between two endpoints. [I-D.brockners-lisp-sr] is an example for how these methods can be applied to LISP.

3.4. SLA Verification

In-situ OAM can help users of an overlay-service to verify that negotiated SLAs for the real traffic are met by the underlay network provider. Different from solutions which rely on active probes to test an SLA, in-situ OAM based mechanisms avoid wrong interpretations and "cheating", which can happen if the probe traffic that is used to perform SLA-check is prioritized by the network provider of the underlay. In active/standby deployments in-situ OAM would only allow for SLA verification of the active path.

3.5. Analytics and Diagnostics

Network planners and operators benefit from knowledge of the actual traffic distribution in the network. When deriving an overall network connectivity traffic matrix one typically needs to correlate data gathered from each individual device in the network. If the path of a packet is recorded while the packet is forwarded, the entire path that a packet took through the network is available to the egress system. This obviates the need to retrieve individual traffic statistics from every device in the network and correlate those statistics, or employ other mechanisms such as leveraging traffic engineering with null-bandwidth tunnels just to retrieve the appropriate statistics to generate the traffic matrix.

In addition, with individual path tracing, information is available at packet level granularity, rather than only at aggregate level - as is usually the case with IPFIX-style methods which employ flow-filters at the network elements. Data-center networks which use equal-cost multipath (ECMP) forwarding are one example where detailed statistics on flow distribution in the network are highly desired. If a network supports ECMP, one can create detailed statistics for the different paths packets take through the network at the egress

system, without a need to correlate/aggregate statistics from every router in the system. Transit devices are off-loaded from the task of gathering packet statistics.

In high-speed networks one can leverage and benefit from packet-accurate measurements with for example hardware-accurate timestamping (i.e., nanosecond-level verification) to support optimized packet scheduling and queuing mechanisms.

3.6. Frame Replication/Elimination Decision for Bi-casting/Active-active Networks

Bandwidth- and power-constrained, time-sensitive, or loss-intolerant networks (e.g., networks for industry automation/control, health care) require efficient OAM methods to decide when to replicate packets to a secondary path in order to keep the loss/error-rate for the receiver at a tolerable level - and also when to stop replication and eliminate the redundant flow. Many Internet of Things (IoT) networks are time sensitive and cannot leverage automatic retransmission requests (ARQ) to cope with transmission errors or lost packets. Transmitting the data over multiple disparate paths (often called bi-casting or live-live) is a method used to reduce the error rate observed by the receiver. Time sensitive networks (TSN) receive a lot of attention from the manufacturing industry as shown by a various standardization activities and industry forums being formed (see e.g., IETF 6TiSCH, IEEE P802.1CB, AVnu).

3.7. Proof of Transit

Several deployments use traffic engineering, policy routing, segment routing or Service Function Chaining (SFC) [RFC7665] to steer packets through a specific set of nodes. In certain cases regulatory obligations or a compliance policy require to prove that all packets that are supposed to follow a specific path are indeed being forwarded across the exact set of nodes specified. If a packet flow is supposed to go through a series of service functions or network nodes, it has to be proven that all packets of the flow actually went through the service chain or collection of nodes specified by the policy. In case the packets of a flow weren’t appropriately processed, a verification device would be required to identify the policy violation and take corresponding actions (e.g., drop or redirect the packet, send an alert etc.) corresponding to the policy. In today’s deployments, the proof that a packet traversed a particular service chain is typically delivered in an indirect way: Service appliances and network forwarding are in different trust domains. Physical hand-off-points are defined between these trust domains (i.e., physical interfaces). Or in other terms, in the "network forwarding domain" things are wired up in a way that traffic

is delivered to the ingress interface of a service appliance and received back from an egress interface of a service appliance. This "wiring" is verified and trusted. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using technologies such as Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. Because of that very reason, networks operators require that different trust layers not to be mixed in the same device. For an NFV scenario a different proof is required. Offering a proof that a packet traversed a specific set of service functions would allow network operators to move away from the above described indirect methods of proving that a service chain is in place for a particular application.

Deployed service chains without the presence of a "proof of transit" mechanism are typically operated as fail-open system: The packets that arrive at the end of a service chain are processed. Adding "proof of transit" capabilities to a service chain allows an operator to turn a fail-open system into a fail-close system, i.e. packets that did not properly traverse the service chain can be blocked.

A solution approach could be based on OAM data which is added to every packet for achieving Proof Of Transit (POT).The OAM data is updated at every hop and is used to verify whether a packet traversed all required nodes. When the verifier receives each packet, it can validate whether the packet traversed the service chain correctly. The detailed mechanisms used for path verification along with the procedures applied to the OAM data carried in the packet for path verification are beyond the scope of this document. Details are addressed in [I-D.brockners-proof-of-transit]. In this document the term "proof" refers to a discrete set of bits that represents an integer or string carried as OAM data. The OAM data is used to verify whether a packet traversed the nodes it is supposed to traverse.

3.8. Use Cases

In-situ OAM could be leveraged for several use cases, including:

- Traffic Matrix: Derive the network traffic matrix: Traffic for a given time interval between any two edge nodes of a given domain. Could be performed for all traffic or on a per Quality of Service (QoS) class.

- Flow Debugging: Discover which path(s) a particular set of traffic (identified by an n-tuple) takes in the network. Such a procedure

is particularly useful in case traffic is balanced across multiple paths, like with link aggregation (LACP) or equal cost multi-pathing (ECMP).

- Loss Statistics per Path: Retrieve loss statistics per flow and path in the network.

- Path Heat Maps: Discover highly utilized links in the network.

- Trend Analysis on Traffic Patterns: Analyze if (and if so how) the forwarding path for a specific set of traffic changes over time (can give hints to routing issues, unstable links etc.)

- Network Delay Distribution: Show delay distribution across network by node or links. If enabled per application or for a specific flow then display the path taken along with the delay incurred at every hop.

- SLA Verification: Verify that a negotiated service level agreement (SLA), e.g., for packet drop rates or delay/jitter is conformed to by the actual traffic.

- Low-power Networks: Include application level OAM information (e.g., battery charge level, cache or buffer fill level) into data traffic to avoid sending extra OAM traffic which incur an extra cost on the devices. Using the battery charge level as example, one could avoid sending extra OAM packets just to communicate battery health, and as such would save battery on sensors.

- Path Verification or Service Function Path Verification: Proof and verification of packets traversing check points in the network, where check points can be nodes in the network or service functions.

- Geo-location Policy: Network policy implemented based on which path packets took. Example: Only if packets originated and stayed within the trading-floor department, access to specific applications or servers is granted.

- Device-level Troubleshooting and Optimization: In many cases, network operators could benefit from information specific to a single device. A non-exhaustive list of useful information includes: queue-depths, buffer utilization (either shared or per-port), packet latency measured from a known starting point, packet latency introduced by a single device, and resource utilization (CPU, memory, link bandwidth) of a given device or link. In some cases, this information changes over per-packet timescales (i.e., nanoseconds) and as such it is extremely challenging to collect

and report this info in an accurate and scalable manner. By encoding the information from the forwarding element directly within a data packet (i.e., within the 'fast-path') this information can be added to some or all data packets and then collected and analyzed by human or machine tools. This type of information is particularly valuable for troubleshooting low-level device errors as well as providing a knowledge feedback loop for network and device optimization.

- Custom Network Probing: Active network probing and in-situ OAM can be combined for customized and efficient network probing. This could for example be a customized traceroute.

4. Considerations for In-situ OAM

The implementation of an in-situ OAM mechanism needs to take several considerations into account, including administrative boundaries, how information is recorded, Maximum Transfer Unit (MTU), Path MTU Discovery (PMTUD) and packet size, etc.

4.1. Type of Information to be Recorded

The information gathered for in-situ OAM can be categorized into three main categories: Information with a per-hop scope, such as path tracing; information which applies to a specific set of hops, such as path or service chain verification; information which only applies to the edges of a domain, such as sequence numbers. Note that a single network device could comprise several in-situ OAM hops, for example in case one wants to trace the path of a packet through that device.

- "edge to edge": Information that needs to be shared between network edges (the "edge" of a network could either be a host or a domain edge device): Edge to edge data e.g., packet and octet count of data entering a well-defined domain and leaving it is helpful in building traffic matrix, sequence number (also called "path packet counters") is useful for the flow to detect packet loss.

- "selected hops": Information that applies to a specific set of nodes only. In case of path verification, only the nodes which are "check points" are required to interpret and update the information in the packet.

- "per hop": Information that is gathered at every hop along the path a packet traverses within an administrative domain:

  * Hop by Hop information e.g., Nodes visited for path tracing, Timestamps at each hop to find delays along the path

* Stats collection at each hop to optimize communication in resource constrained networks e.g., battery, CPU, memory status of each node piggy backed in a data packet is useful in low power lossy networks where network nodes are mostly asleep and communication is expensive

4.2. MTU and Packet Size

The recorded data at every hop might lead to packet size exceeding the Maximum Transmit Unit (MTU). A detailed discussion of the implications of oversized IPv6 header chains is found in [RFC7112]. The Path MTU restricts the amount of data that can be recorded for purpose of OAM within a data packet.

If in-situ OAM data is inserted at the edge of the domain (e.g., by intermediate routers) then the MTU on all interfaces with the domain (MTU_INT) MUST be >= the maximum MTU on any "external" facing interfaces (MTU_EXT) and the total size of in-situ OAM data to be recorded MUST be <= (MTU_INT - MTU_EXT).

In-situ OAM comprises two approaches to insert OAM data fields in the packets:

- Pre-allocated: In this case, the encapsulating node inserts empty data fields into the packet to cover the entire domain. The data fields will be incrementally updated/filled as the packet progresses through the network. With pre-allocation the packet size is only changed at the encapsulating node and is kept constant throughout the domain. The pre-allocated approach is beneficial for software data-plane implementations where allocating the required space only once and index into the array to populate the data during transit avoids copy operations at every hop.

- Incremental: Every node that desires to include in-situ OAM information extends the packet as needed. The incremental approach is beneficial for hardware data-plane implementations as it eliminates the need for the transit nodes to read the full array and lookup the pointer in the option prior to updating the data fields contents.

The "incremental" or the "pre-allocated" approaches could even be combined in the same deployment - in which case two in-situ OAM headers would be present in the packet: One for the incremental approach and one for the pre-allocated approach. In such a case one would expect that nodes with a hardware data-plane would update the incremental header, whereas nodes with a software data-plane would process the pre-allocated header.

4.3. Administrative Boundaries

There are several challenges in enabling in-situ OAM in the public Internet as well as in corporate/enterprise networks across administrative domains, which include but are not limited to:

- Deployment dependent, the data fields that in-situ OAM requires as part of a specific transport protocol may not be supported across administrative boundaries.

- Current OAM implementations are often done in the slow path, i.e., OAM packets are punted to router's CPU for processing. This leads to performance and scaling issues and opens up routers for attacks such as Denial of Service (DoS) attacks.

- Discovery of network topology and details of the network devices across administrative boundaries may open up attack vectors compromising network security.

- Specifically on IPv6: At the administrative boundaries IPv6 packets with extension headers are dropped for several reasons described in [RFC7872].

The following considerations will be discussed in a future version of this document: If the packet is dropped due to the presence of the in-situ OAM; If the policy failure is treated as feature disablement and any further recording is stopped but the packet itself is not dropped, it may lead to every node in the path to make this policy decision.

4.3.1. Layered In-Situ OAM Domains

Like any OAM domain, in-situ OAM domains could also be layered/ nested. Layering/nesting of in-situ OAM follows the general approach of OAM layering: An in-situ OAM domain consists of maintenance end-points (MEP) and maintenance intermediate points (MIP). MEP add to or remove the entire set of in-situ OAM data fields from the traffic, while only MIP update or add in-situ OAM data fields. When in-situ OAM layering is employed, a MEP of one layer becomes a MIP in the layer above, while MIP of the lower layer are not visible to the layer above - unless specifically configured otherwise.

Consider the following examples:

- NSH over IPv6: In-situ OAM data fields could be present in both transport protocols: NSH and IPv6, with NSH forming the overlay network and IPv6 forming the underlay network. The network which deploys NSH would form an in-situ OAM domain. In addition each

IPv6 underlay network which connects two NSH nodes forms an in-situ OAM domain. The in-situ OAM domain with NSH as transport could be considered as layered on top of the different in-situ OAM domains which use IPv6 as transport.

- NSH using an in-situ OAM aware transport: Consider a case where the underlay network would not natively support in-situ OAM, still the individual transport nodes would have the capability to "look deep into the packet" and update/add in-situ OAM information in the NSH header. The in-situ OAM domain with NSH as transport could be considered as layered on top of the different in-situ OAM domains which are in-situ OAM aware and connect the individual NSH nodes.

4.4. Selective Enablement

The ability to selectively enable in-situ OAM is valuable. While it may be desirable to enable data collection on all traffic or devices, this may not always be feasible. In-situ OAM collection may also come with a performance impact to forwarding rates or feature capabilities, which may be acceptable in only some locations. For example, the SPUD prototype uses the notion of "pipes" to describe the portion of the traffic that could be subject to in-path inspection. Mechanisms to decide which traffic would be subject to in-situ OAM are outside the scope of this document.

4.5. Forwarding Behavior

In-situ OAM adds additional data fields to live user traffic and as such changes the packet which is also why in-situ OAM is characterized as "hybrid, type 1" OAM. The effectiveness of in-situ OAM as a tool for operations depends on forwarding nodes not altering their forwarding behavior in case of in-situ OAM data fields being present in the packet. As a consequence, an implementation of in-situ OAM should not change the forwarding behavior of the packet, i.e. packets with or without in-situ OAM data fields should be handled the same way by a forwarding node (see also the associated requirement further below). Note that there are implementations where the addition of meta-data to live user traffic might cause the forwarding behavior of the packet to change, e.g. certain implementation handle IPv6 packets with or without extension headers differently (see [RFC7872]).

4.6. Optimization of Node and Interface Identifiers

Since packets have a finite maximum size, the data recording or carrying capacity of one packet in which the in-situ OAM metadata is present is limited. In-situ OAM should use its own dedicated

namespace (confined to the domain in-situ OAM operates in) to represent node and interface IDs to save space in the header. Generic representations of node and interface identifiers which are globally unique (such as a UUID) would consume significantly more bits of in-situ OAM data.

4.7. Loop Communication Path (IPv6-specifics)

When recorded data is required to be analyzed on a source node that issues a packet and inserts in-situ OAM data, the recorded data needs to be carried back to the source node.

One way to carry the in-situ OAM data back to the source is to utilize an ICMP Echo Request/Reply (ping) or ICMPv6 Echo Request/ Reply (ping6) mechanism. In order to run the in-situ OAM mechanism appropriately on the ping/ping6 mechanism, the following two operations should be implemented by the ping/ping6 target node:

1. All of the in-situ OAM fields would be copied from an Echo Request message to an Echo Reply message.

2. The Hop Limit field of the IPv6 header of these messages would be copied as a continuous sequence. Further considerations are addressed in a future version of this document.

5. Requirements for In-situ OAM Data Types

The above discussed use cases require different types of in-situ OAM data. This section details requirements for in-situ OAM derived from the discussion above.

5.1. Generic Requirements

REQ-G1: Classification: It should be possible to enable in-situ OAM on a selected set of traffic (e.g., per interface, based on an access control list specifying a specific set of traffic, etc.) The selected set of traffic can also be all traffic.

REQ-G2: Scope: If in-situ OAM is used only within a specific domain, provisions need to be put in place to ensure that in-situ OAM data stays within the specific domain only.

REQ-G3: Transport independence: Data formats for in-situ OAM shall be defined in a transport independent way. In-situ OAM applies to a variety of transport protocols. Encapsulations should be defined how the generic data formats are carried by a specific protocol.

REQ-G4: Layering: It should be possible to have in-situ OAM information for different transport protocol layers be present in several fields within a single packet. This could for example be the case when tunnels are employed and in-situ OAM information is to be gathered for both the underlay as well as the overlay network. Layering support should not be limited to just underlay and overlay, but include more than two layers.

REQ-G5: MTU size: With in-situ OAM information added, packets MUST NOT become larger than the path MTU.

REQ-G5.1: If due to some reason a packet which contains in situ OAM data fields cannot be forwarded due to the presence of in-situ OAM data fields, the node SHOULD remove the in situ OAM data fields and forward the packet, rather than drop the entire packet.

REQ-G5.2: If the encapsulating router is unable to insert in-situ OAM data fields into a packet, e.g., due to MTU issues, even though it is configured to do so, it should use some operational means to inform the operator (e.g., syslog) about the inability to add in-situ OAM data fields. Even if the in-situ OAM encapsulating node fails to add in-situ OAM data fields, it should forward the packet normally.

REQ-G5.3: MTU size consideration for in-situ OAM MUST take domain specifics into account, e.g., changes of the domain topology due to path protection mechanisms might extend the hop count of a path etc.

REQ-G6: Data structure reuse: The data fields and associated types defined and used for in-situ OAM ought to be reusable for out-of-band OAM telemetry as well.

REQ-G7: Data fields: It is desirable that the format of in-situ OAM data fields leverages already defined data formats for OAM as much as feasible.

REQ-G8: Combination with active OAM mechanisms: In-situ OAM should be usable for active network probing, like for example a customized version of traceroute. Decapsulating in-situ OAM nodes may have an ability to send the in-situ OAM

information retrieved from the packet back to the source address of the packet or to the encapsulating node.

REQ-G9: Unaltered forwarding behavior of in-situ OAM nodes: The addition of in-situ OAM data fields should not change the way packets are forwarded within the in-situ OAM domain.

REQ-G10: Layering of in-situ OAM domains: It should be possible to layer in-situ OAM domains on each other. Layering should be supported within the same, as well as with different transport protocols which carry in-situ OAM data fields.

5.2. In-situ OAM Data with Per-hop Scope

REQ-H1: Missing nodes detection: Data shall be present that allows a node to detect whether all nodes that might participate in in-situ OAM operations have indeed participated.

REQ-H2: Node, instance or device identifier: Data shall be present that allows to retrieve the identity of the entity reporting telemetry information. The entity can be a device, or a subsystem/component within a device. The latter will allow for packet tracing within a device in much the same way as between devices.

REQ-H3: Ingress interface identifier: Data shall be present that allows the identification of the interface a particular packet was received from. The interface can be a logical and/or physical entity.

REQ-H4: Egress interface identifier: Data shall be present that allows the identification of the interface a particular packet was forwarded to. Interface can be a logical or physical entity.

REQ-H5: Time-related requirements

REQ-H5.1: Delay: Data shall be present that allows to retrieve the delay between two or more points of interest within the system. Those points can be within the same device or on different devices.

REQ-H5.2: Jitter: Data shall be present that allows to retrieve the jitter between two or more points of interest within the system. Those points can be within the same device or on different devices. Jitter can be derived from the different

timestamps gathered and does not necessarily need to be an explicit data field.

REQ-H5.3: Wall-clock time: Data shall be present that allows to retrieve the wall-clock time visited a particular point of interest in the system.

REQ-H5.4: Time precision: Time with different precision should be supported. Use-case dependent, the required precision could e.g., be nanoseconds, microseconds, milliseconds, or seconds.

REQ-H6: Generic data fields (like e.g., GPS/Geo-location information): It should be possible to add user-defined OAM data at select hops to the packet. The semantics of the data are defined by the user.

5.3. In-situ OAM with Selected Hop Scope

REQ-S1: Proof of transit: Data shall be present which allows to securely prove that a packet has visited or ore several particular points of interest (i.e., a particular set of nodes).

REQ-S1.1: In case "Shamir’s secret sharing scheme" is used for proof of transit, two data fields, "random" and "cumulative" shall be present. The number of bits used for "random" and "cumulative" data fields can vary between deployments and should thus be configurable.

REQ-S1.2: Enable a fail-open service chaining system to be converted into a fail-closed service chaining system.

5.4. In-situ OAM with End-to-end Scope

REQ-E1: Sequence numbering:

REQ-E1.1: Reordering detection: It should be possible to detect whether packets have been reordered while traversing an in situ OAM domain.

REQ-E1.2: Duplicates detection: It should be possible to detect whether packets have been duplicated while traversing an in situ OAM domain.

REQ-E1.3: Detection of packet drops: It should be possible to detect whether packets have been dropped while traversing an in-situ OAM domain.

6. Security Considerations and Requirements

6.1. General considerations

General Security considerations will be expanded on in a later version of this document.

In-situ OAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring in-situ OAM according to their needs. Still operators need to properly secure the in-situ OAM domain to avoid malicious configuration and use, which could include injecting malicious in-situ OAM packets into a domain.

6.2. Proof of Transit

Threat Model: Attacks on the deployments could be due to malicious administrators or accidental misconfiguration resulting in bypassing of certain nodes. The solution approach should meet the following requirements:

REQ-SEC1: Sound Proof of Transit: A valid and verifiable proof that the packet definitively traversed through all the nodes as expected. Probabilistic methods to achieve this should be avoided, as the same could be exploited by an attacker.

REQ-SEC2: Tampering of meta data: An active attacker should not be able to insert or modify or delete meta data in whole or in parts and bypass few (or all) nodes. Any deviation from the expected path should be accurately determined.

REQ-SEC3: Replay Attacks: A attacker (active/passive) should not be able to reuse the POT bits in the packet by observing the OAM data in the packet, packet characteristics (like IP addresses, octets transferred, timestamps) or even the proof bits themselves. The solution approach should consider usage of these parameters for deriving any secrets cautiously. Mitigating replay attacks beyond a window of longer duration could be intractable to achieve with fixed number of bits allocated for proof.

REQ-SEC4: Pre-play Attacks: A active attacker should not be able to generate or reuse valid POT bits from legitimate packets, in order to prove to the verifier as valid packets. This

slight variant of replay attacks. The attacker extracts POT bits from legitimate packets and ensure they do not reach the verifier. Subsequently reuse those POT bits in crafted packets.

REQ-SEC5: Recycle Secrets: Any configuration of the secrets (like cryptographic keys, initialization vectors etc.) either in the controller or service functions should be re-configurable. Solution approach should enable controls, API calls etc. needed in order to perform such recycling. It is desirable to provide recommendations on the duration of rotation cycles needed for the secure functioning of the overall system.

REQ-SEC6: Secret storage and distribution: Secrets should be shared with the devices over secure channels. Methods should be put in place so that secrets cannot be retrieved by non-authorized personnel from the devices.

7. IANA Considerations

[RFC Editor: please remove this section prior to publication.]

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Jen Linkova, LJ Wobker, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Ignas Bagdonas, LJ Wobker, Erik Nordmark, Vengada Prasad Govindan, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

9. References

9.1. Normative References

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

9.2. Informative References

[I-D.brockners-lisp-sr]
Brockners, F., Bhandari, S., Maino, F., and D. Lewis, "LISP Extensions for Segment Routing", draft-brockners-lisp-sr-01 (work in progress), February 2014.

[I-D.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof of Transit", draft-brockners-proof-of-transit-02 (work in progress), October 2016.

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-10 (work in progress), November 2016.

[I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.

[I-D.lapukhov-dataplane-probe]
Lapukhov, P. and r. remy@barefootnetworks.com, "Data-plane probe for in-band telemetry collection", draft-lapukhov-dataplane-probe-01 (work in progress), June 2016.

[P4] Kim, , "P4: In-band Network Telemetry (INT)", September 2015.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <http://www.rfc-editor.org/info/rfc4884>.

[RFC4950] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "ICMP Extensions for Multiprotocol Label Switching", RFC 4950, DOI 10.17487/RFC4950, August 2007, <http://www.rfc-editor.org/info/rfc4950>.

[RFC5837] Atlas, A., Ed., Bonica, R., Ed., Pignataro, C., Ed., Shen, N., and JR. Rivers, "Extending ICMP for Interface and Next-Hop Identification", RFC 5837, DOI 10.17487/RFC5837, April 2010, <http://www.rfc-editor.org/info/rfc5837>.

[RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <http://www.rfc-editor.org/info/rfc7112>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <http://www.rfc-editor.org/info/rfc7276>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

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

[RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <http://www.rfc-editor.org/info/rfc7872>.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Sashank Dara
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: sadara@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

Petr Lapukhov
Facebook
1 Hacker Way
Menlo Park, CA 94025
USA

URI: petr@fb.com

Remy Chang
Barefoot Networks

Email: remy@barefootnetworks.com

ippm
Internet-Draft
Intended status: Informational
Expires: January 3, 2018

F. Brockners
S. Bhandari
V. Govindan
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
July 02, 2017

Encapsulations for In-situ OAM Data
draft-brockners-inband-oam-transport-05

Abstract

In-situ Operations, Administration, and Maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. In-situ OAM is to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets. This document outlines how in-situ OAM data fields can be transported in protocols such as NSH, Segment Routing, VXLAN-GPE, native IPv6 (via extension headers), and IPv4. Transport options are currently investigated as part of an implementation study. This document is intended to only serve informational purposes.

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/.

Copyright Notice

Table of Contents

1. Introduction

This document discusses transport mechanisms for "in-situ" Operations, Administration, and Maintenance (OAM) data fields. In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. Data types and data formats for in-situ OAM are defined in [I-D.brockners-inband-oam-data].

This document outlines transport encapsulations for the in-situ OAM data defined in [I-D.brockners-inband-oam-data]. This document is to serve informational purposes only. As part of an in-situ OAM implementation study different protocol encapsulations for in-situ OAM data are being explored. Once data formats and encapsulation approaches are settled, protocol specific specifications for in-situ OAM data transport will address the standardization aspect.

The data for in-situ OAM defined in [I-D.brockners-inband-oam-data] can be carried in a variety of protocols based on the deployment needs. This document discusses transport of in-situ OAM data for the following protocols:

- IPv6
- IPv4
- VXLAN-GPE
- NSH
- Segment Routing (IPv6 and MPLS)

This list is non-exhaustive, as it is possible to carry the in-situ OAM data in several other protocols and transports.

A feasibility study of in-situ OAM is currently underway as part of the FD.io project [FD.io]. The in-situ OAM implementation study should be considered as a "tool box" to showcase how "in-situ" OAM can complement probe-packet based OAM mechanisms for different

deployments and packet transport formats. For details, see the open source code in the FD.io [FD.io].

2. Conventions

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].

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

MTU: Maximum Transmit Unit

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

SID: Segment Identifier

SR: Segment Routing

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension

3. In-Situ OAM Metadata Transport in IPv6

This mechanisms of in-situ OAM in IPv6 complement others proposed to enhance diagnostics of IPv6 networks, such as the IPv6 Performance and Diagnostic Metrics Destination Option described in [I-D.ietf-ippm-6man-pdm-option]. The IP Performance and Diagnostic Metrics Destination Option is destination focused and specific to IPv6, whereas in-situ OAM is performed between end-points of the network or a network domain where it is enabled and used.

A historical note: The idea of IPv6 route recording was originally introduced by [I-D.kitamura-ipv6-record-route] back in year 2000. With IPv6 now being generally deployed and new concepts such as Segment Routing [I-D.ietf-spring-segment-routing] being introduced, it is imperative to further mature the Operations, Administration, and Maintenance mechanisms available to IPv6 networks.

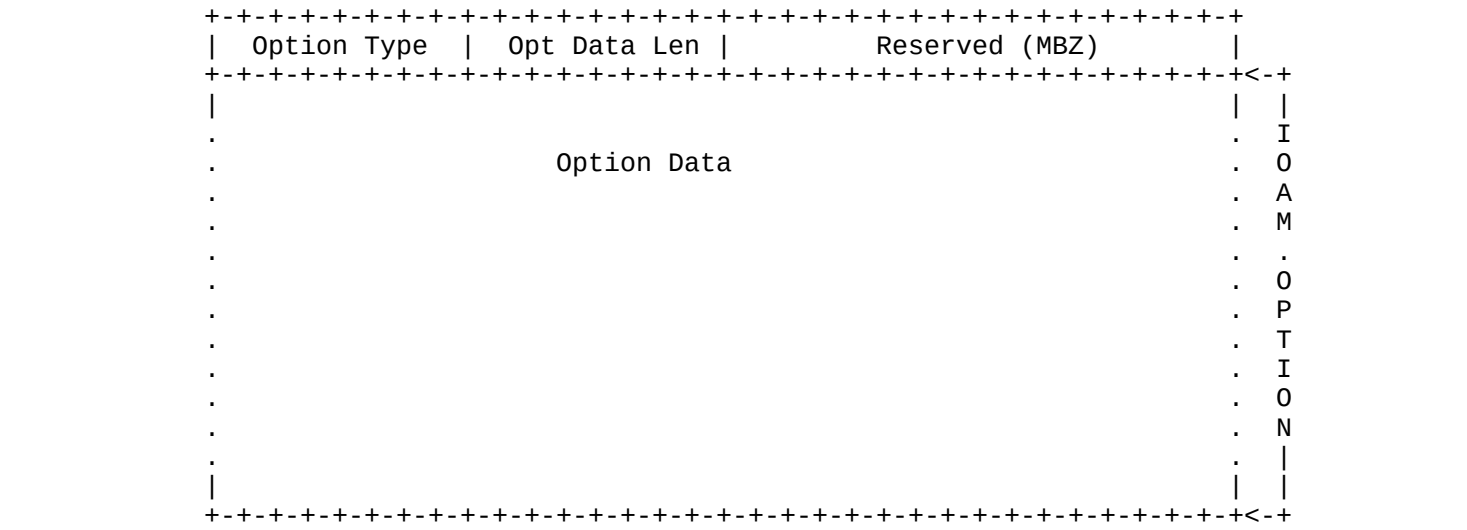
The in-situ OAM options translate into options for an IPv6 hop by hop extension header. The extension header would be inserted by either a host source of the packet, or by a transit/domain-edge node. If the addition of the in-situ OAM Hop-by-Hop Option header would lead to the packet exceeding the MTU of the domain an error should be reported. The methods and procedures of how the error is reported are outside the scope of this document. Likewise if an ICMPv6 forwarding error occurs between encapsulating and decapsulating nodes, the node generating the ICMPv6 error should strip the in-situ OAM Hop-by-Hop Option header before sending the ICMPv6 message to the source.

3.1. In-situ OAM in IPv6 Hop by Hop Extension Header

This section defines in-situ OAM for IPv6 transport. In-situ OAM Options are transported in IPv6 hop-by-hop extension header.

3.1.1. In-situ OAM Hop by Hop Options

IPv6 hop-by-hop option format for carrying in-situ OAM data fields:



Option Type 8-bit identifier of the type of option.

Opt Data Len 8-bit unsigned integer. Length of the Reserved and Option Data field of this option, in octets.

Reserved (MBZ) 16-bit field MUST be filled with zeroes.

Option Data Variable-length field. Option-Type-specific data.

In-situ OAM Options are inserted as Option data as follows:

1. Pre-allocated Tracing Option: The in-situ OAM Preallocated Tracing option defined in [I-D.brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_PRE_TRACE_OPTION_IPV6.

2. Incremental Tracing Option: The in-situ OAM Incremental Tracing option defined in [I-D.brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_INCR_TRACE_OPTION_IPV6.

3. Proof of Transit Option: The in-situ OAM POT option defined in [I-D.brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxxx 8-bit identifier of the type of option. xxxxxx=TBD_IANA_POT_OPTION_IPV6.

4. Edge to Edge Option: The in-situ OAM E2E option defined in [I-D.brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 000xxxxxx 8-bit identifier of the type of option. xxxxxx=TBD_IANA_E2E_OPTION_IPV6.

4. In-situ OAM Metadata Transport in IPv4

Transport of in-situ OAM data in IPv4 will use GRE encapsulation.

GRE encapsulation is defined in [RFC2784]. IOAM is defined as a "set of Protocol Types" TBD_IANA_ETHERNET_NUMBER_IOAM_* and follows GRE header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES].

The different IOAM data fields defined in [I-D.brockners-inband-oam-data] are added as TLVs following the GRE header. In an administrative domain where IOAM is used, insertion of the IOAM protocol header in GRE is enabled at the GRE tunnel endpoints which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.

For IOAM the following new GRE protocol types are requested:

1. IOAM_Trace_Preallocated: TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE_PREALLOCATED

2. IOAM_Trace_Incremental: TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE_INCREMENTAL

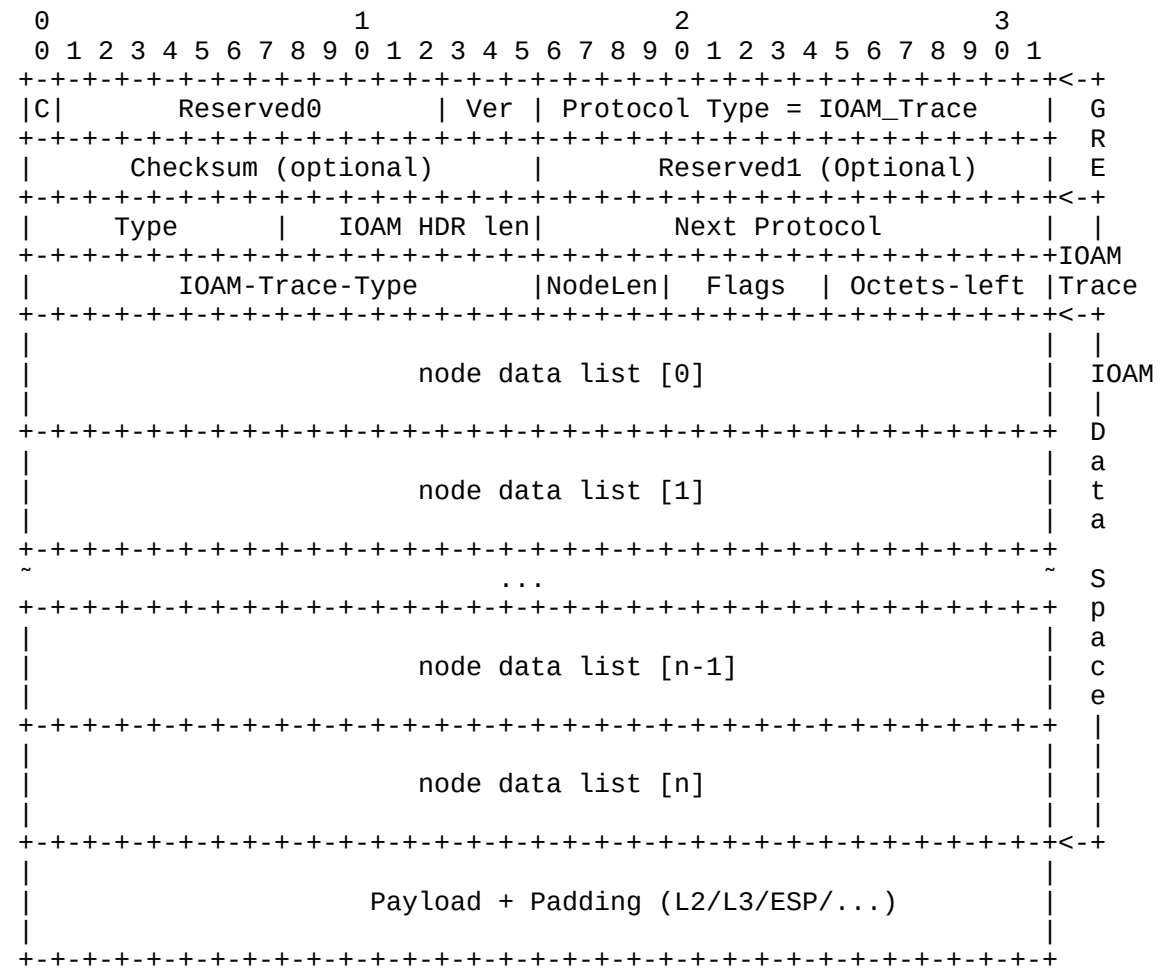
3. IOAM_POT: TBD_IANA_ETHERNET_NUMBER_IOAM_POT

4. IOAM_End-to_End: TBD_IANA_ETHERNET_NUMBER_IOAM_E2E

4.1. In-situ OAM Tracing in GRE

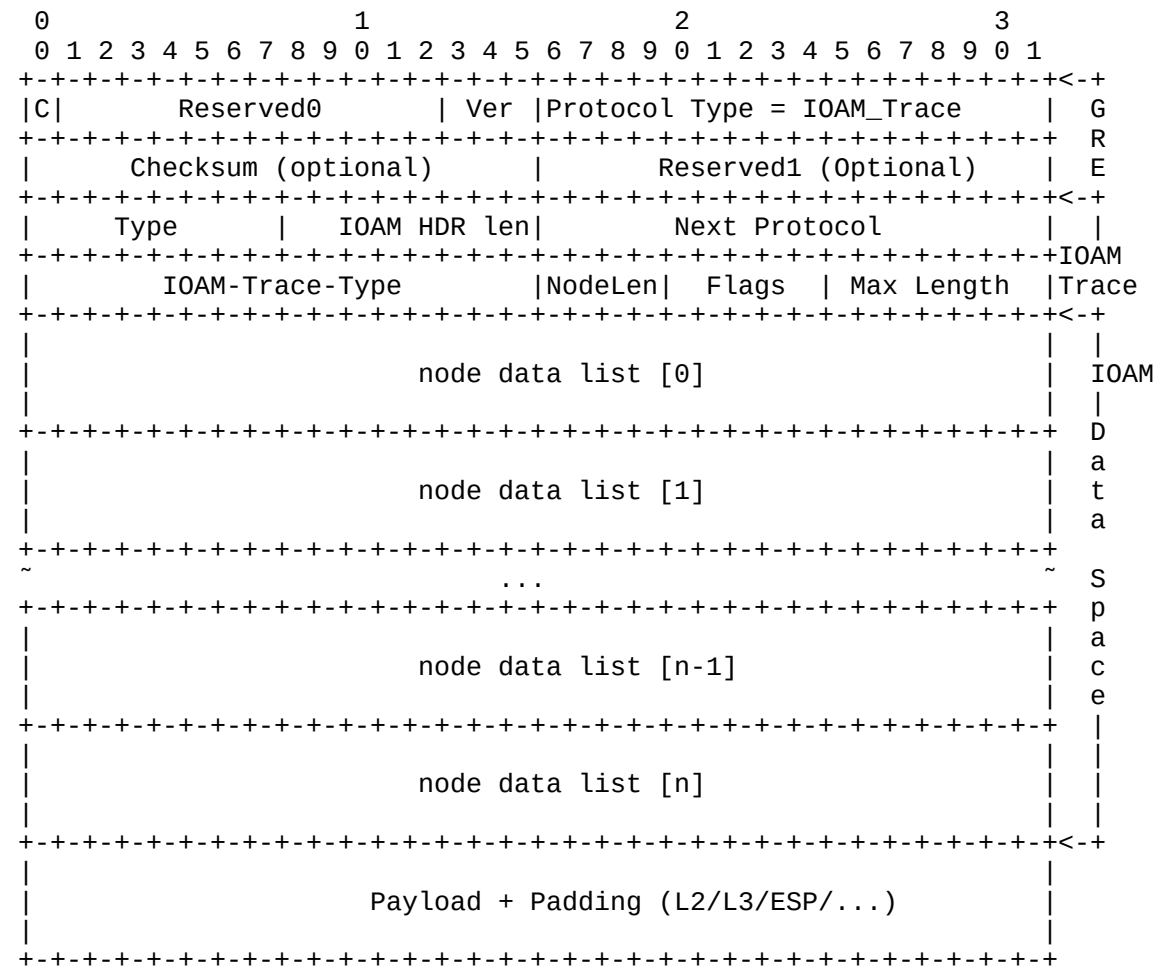
The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported using GRE are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

In-situ OAM Trace header following GRE header(Preallocated IOAM trace):



Pre-allocated Trace Option Data MUST be 4-octet aligned:

In-situ OAM Trace header following GRE header(Incremental IOAM trace):



In-situ OAM Incremental Trace Option Data MUST be 4-octet aligned:

The GRE header and fields are defined in [RFC2784] with Protocol Type set to TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE. IOAM specific fields and header are defined here:

Type: 8-bit unsigned integer defining IOAM header type IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined here.

IOAM HDR Len: 8 bits Length field contains the length of the variable metadata octets.

Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or [ETYPES] SHOULD discard the packet.

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.brockners-inband-oam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Flags: 5-bit field as defined in [I-D.brockners-inband-oam-data].

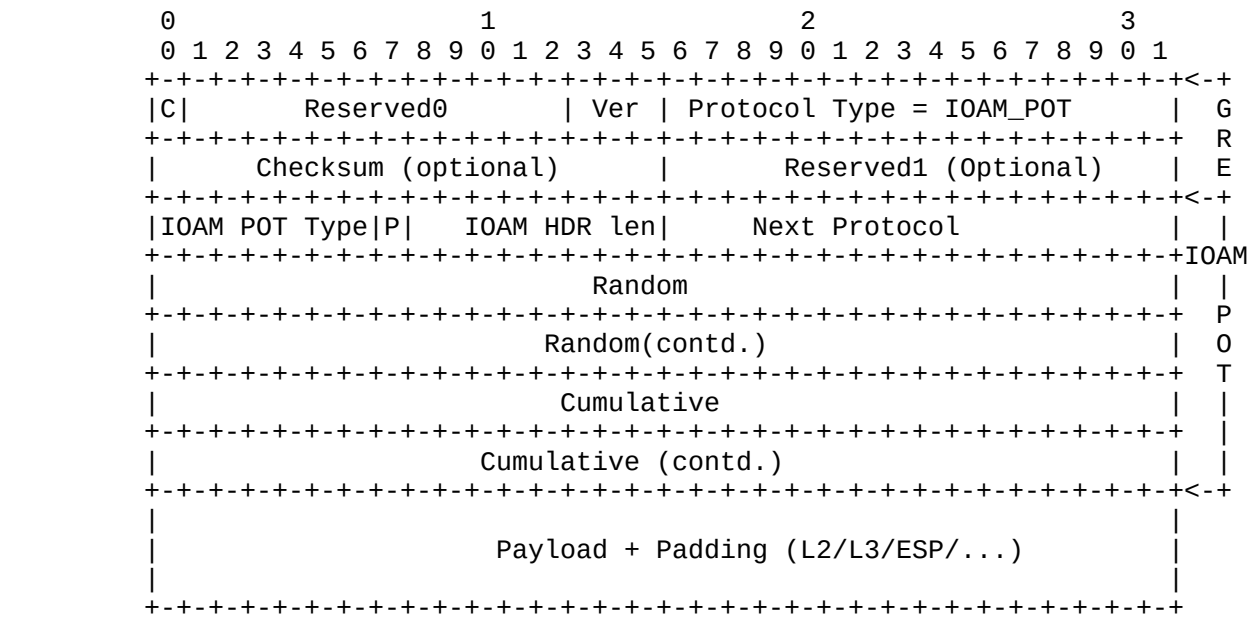
Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

4.2. In-situ OAM POT in GRE

In-situ OAM POT header following GRE header:



The GRE header and fields are defined in [RFC2784] with Protocol Type set to TBD_IANA_ETHERNET_NUMBER_IOAM_POT. IOAM specific fields and header are defined here:

IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

IOAM HDR Len: 8 bits Length field contains the length of the variable metadata octets.

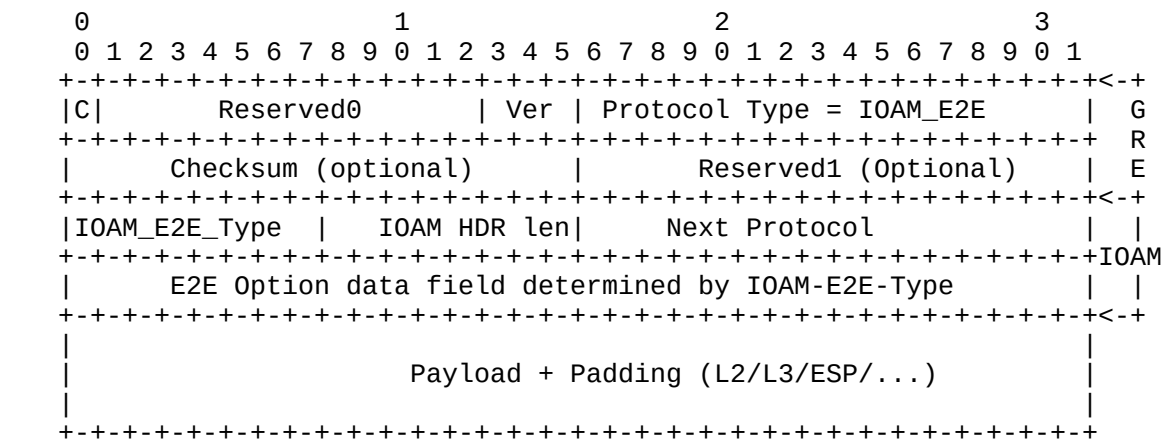
Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or [ETYPES] SHOULD discard the packet.

Random: 64-bit Per-packet random number.

Cumulative: 64-bit Cumulative value that is updated by the Service Functions.

4.3. In-situ OAM End-to-End in GRE

In-situ OAM End-to-End header following GRE header:



IOAM E2E Type: 8-bit identifier of a particular E2E variant that dictates the E2E data that is included as defined in [I-D.brockners-inband-oam-data].

IOAM HDR Len: 8 bits Length field contains the length of the variable metadata octets.

Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or [ETYPES] SHOULD discard the packet.

E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

5. In-situ OAM Metadata Transport in VXLAN-GPE

VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] encapsulation is somewhat similar to IPv6 extension headers in that a series of headers can be contained in the header as a linked list. The different iIOAM types are added as options within a new IOAM protocol header in VXLAN GPE. In an administrative domain where IOAM is used, insertion of the IOAM

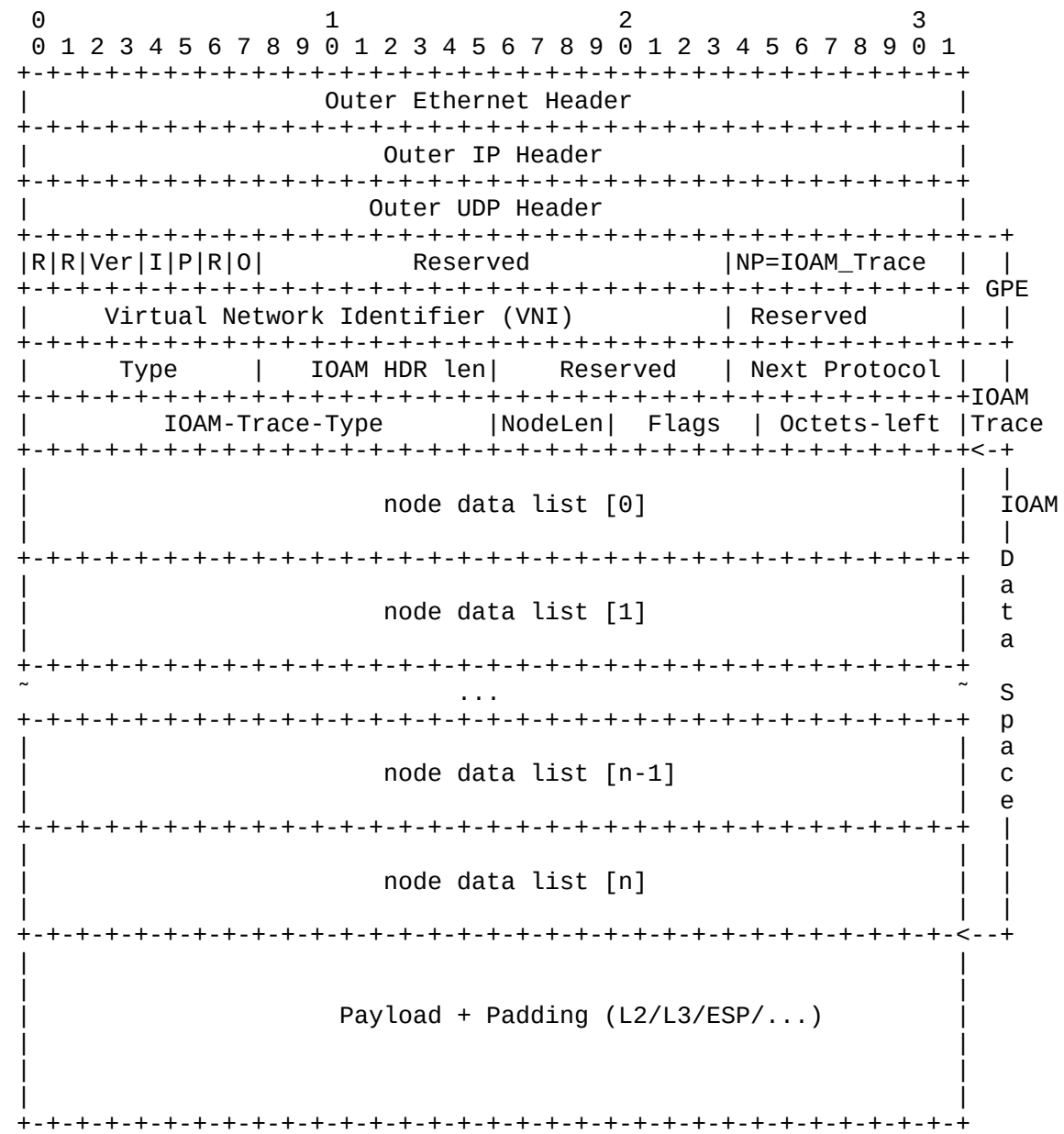
protocol header in VXLAN GPE is enabled at the VXLAN GPE tunnel endpoint which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.

5.1. In-situ OAM Tracing in VXLAN-GPE

The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported in VXLAN-GPE are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

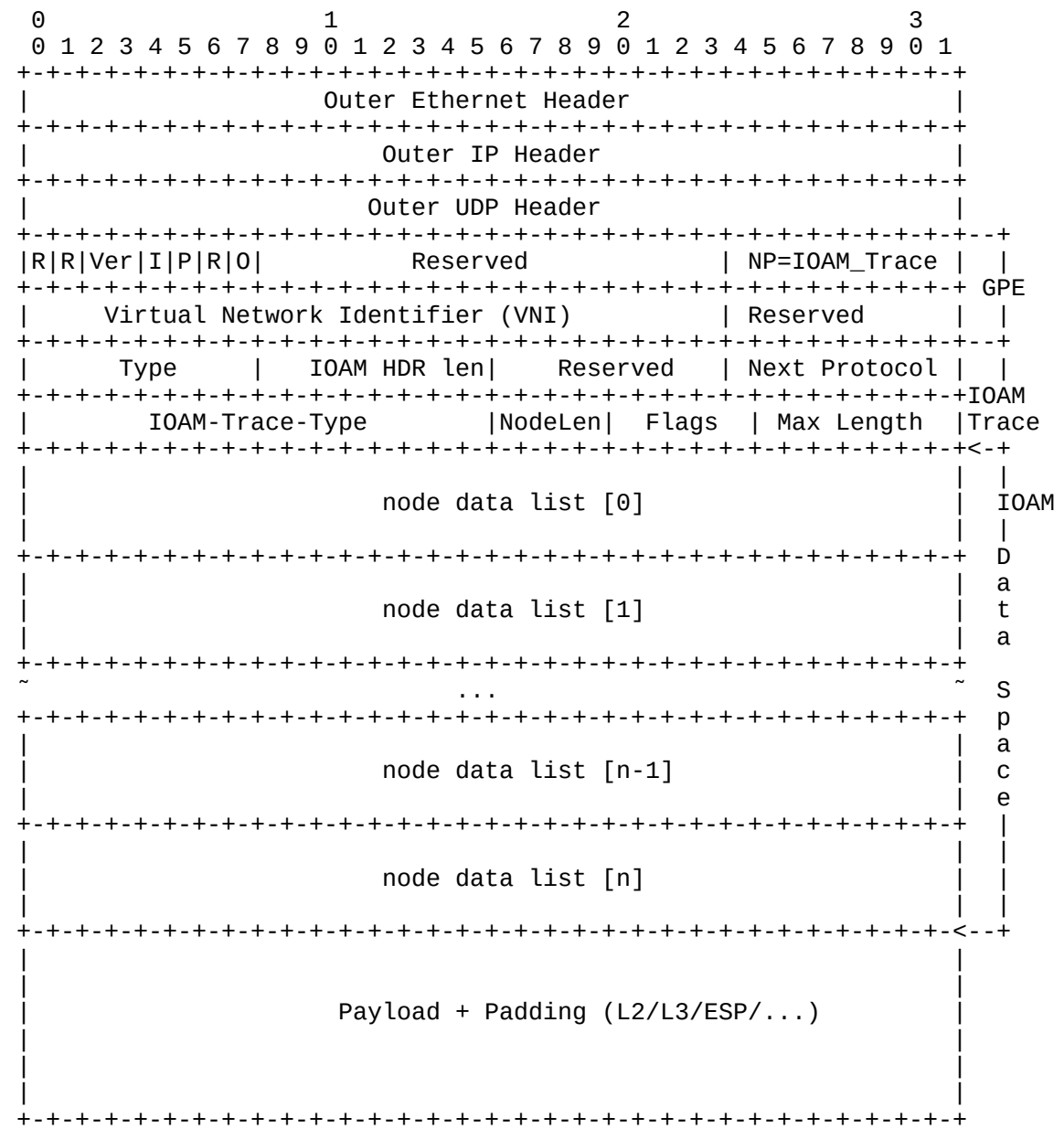
The VXLAN-GPE header and fields are defined in [I-D.ietf-nvo3-vxlan-gpe]. IOAM specific fields and header are defined here:

In-situ OAM Trace header following VXLAN GPE header
(Pre-allocated trace):



Pre-allocated Trace Option Data MUST be 4-octet aligned:

In-situ OAM Trace header following VXLAN GPE header
(Incremental IOAM trace):



In-situ OAM Incremental Trace Option Data MUST be 4-octet aligned:

Type: 8-bit unsigned integer defining IOAM header type IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined here.

IOAM HDR len: 8-bit unsigned integer. Length of the in-situ OAM HDR in 8-octet units.

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.brockners-inband-oam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Flags: 5-bit field as defined in [I-D.brockners-inband-oam-data].

Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

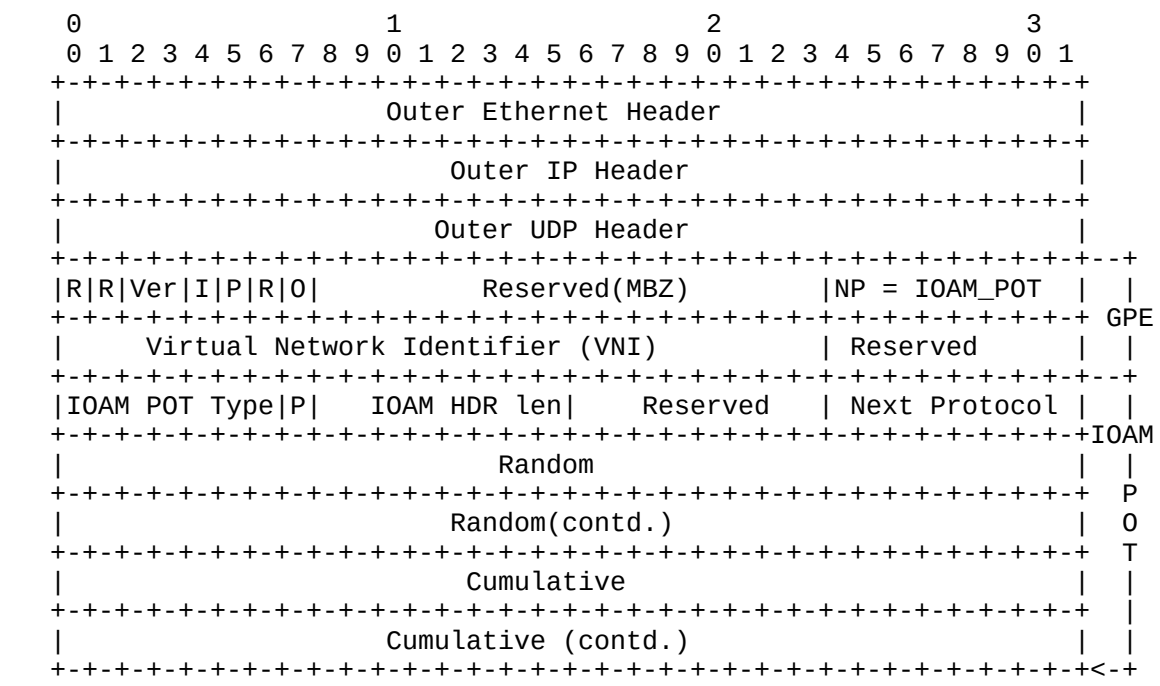
Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

5.2. In-situ OAM POT in VXLAN-GPE

The VXLAN-GPE header and fields are defined in [I-D.ietf-nvo3-vxlan-gpe]. IOAM specific fields and header are defined here:

In-situ OAM POT header following VXLAN GPE header:



IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

IOAM HDR len: 8-bit unsigned integer. Length of the in-situ OAM HDR in 8-octet units

Reserved: 8-bit reserved field MUST be set to zero.

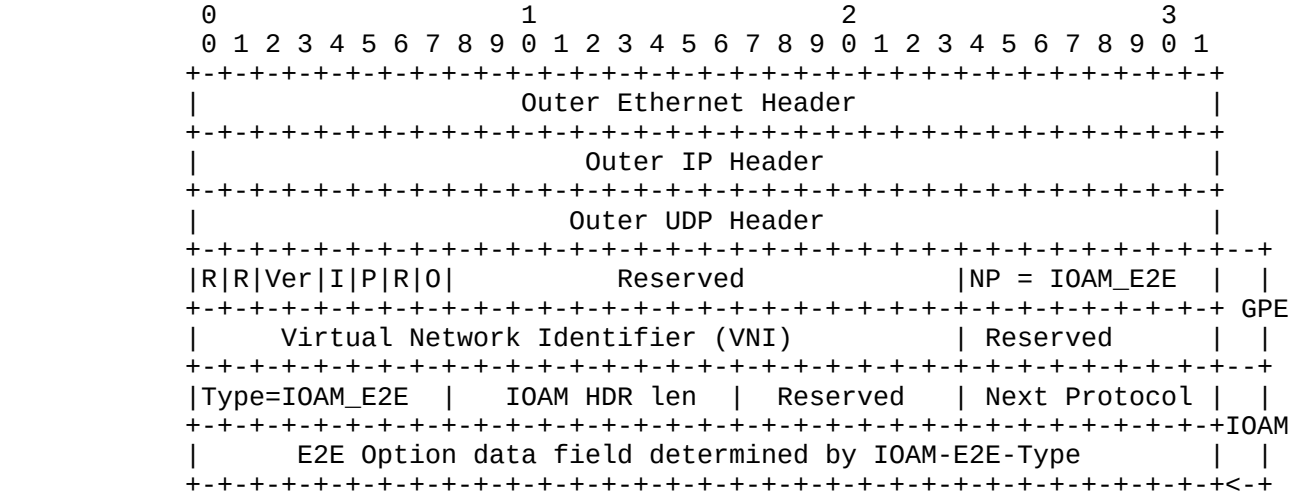
Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

Random: 64-bit Per-packet random number.

Cumulative: 64-bit Cumulative value that is updated by the Service Functions.

5.3. In-situ OAM Edge-to-Edge in VXLAN-GPE

In-situ OAM Edge-to-Edge in VXLAN GPE header:



Type: 8-bit identifier of a particular E2E variant that dictates the E2E data that is included as defined in [I-D.brockners-inband-oam-data].

IOAM HDR len: 8-bit unsigned integer. Length of the in-situ OAM HDR in 8-octet units

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

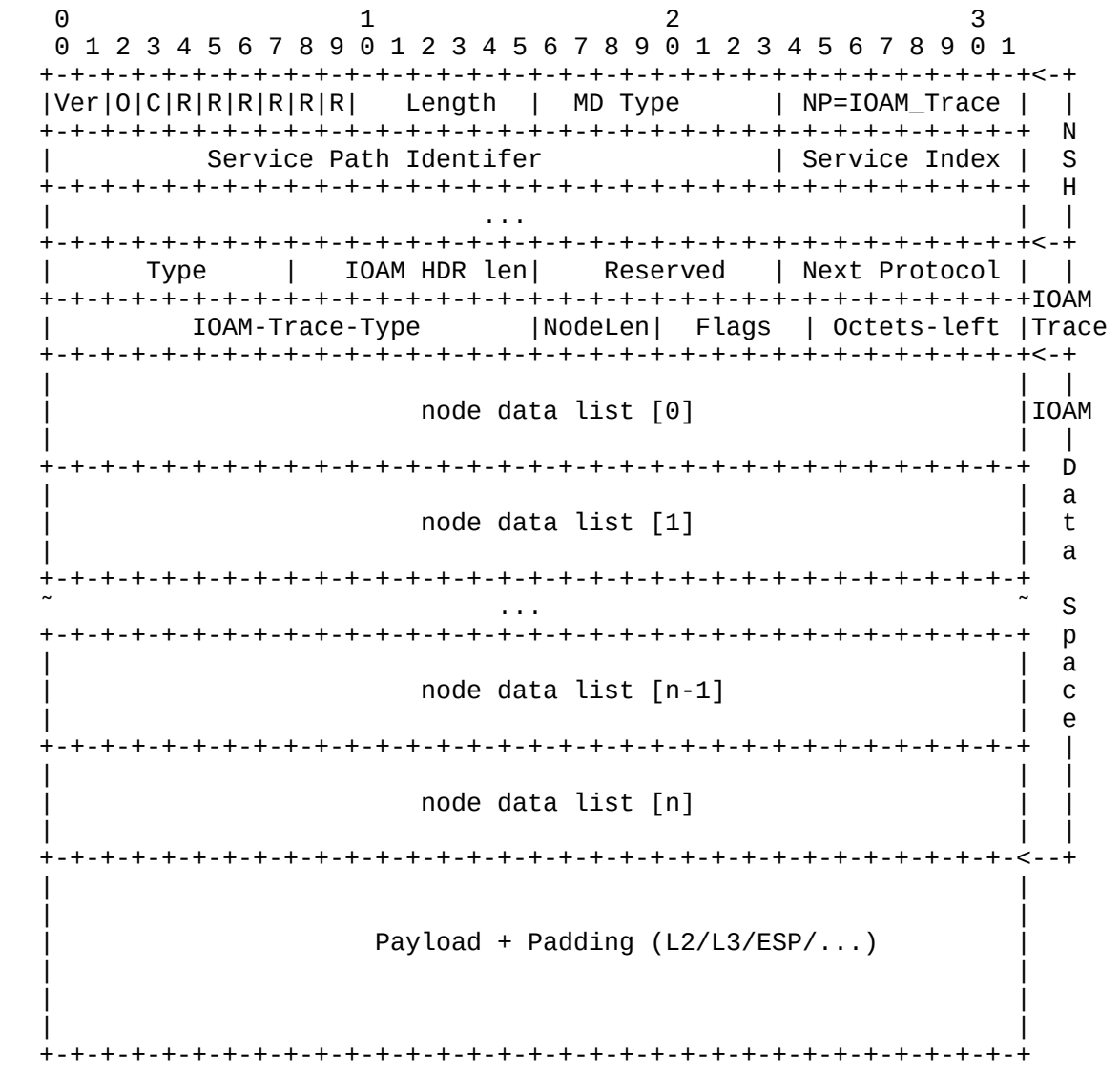
6. In-situ OAM Metadata Transport in NSH

6.1. In-situ OAM Tracing in NSH

The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported in NSH are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

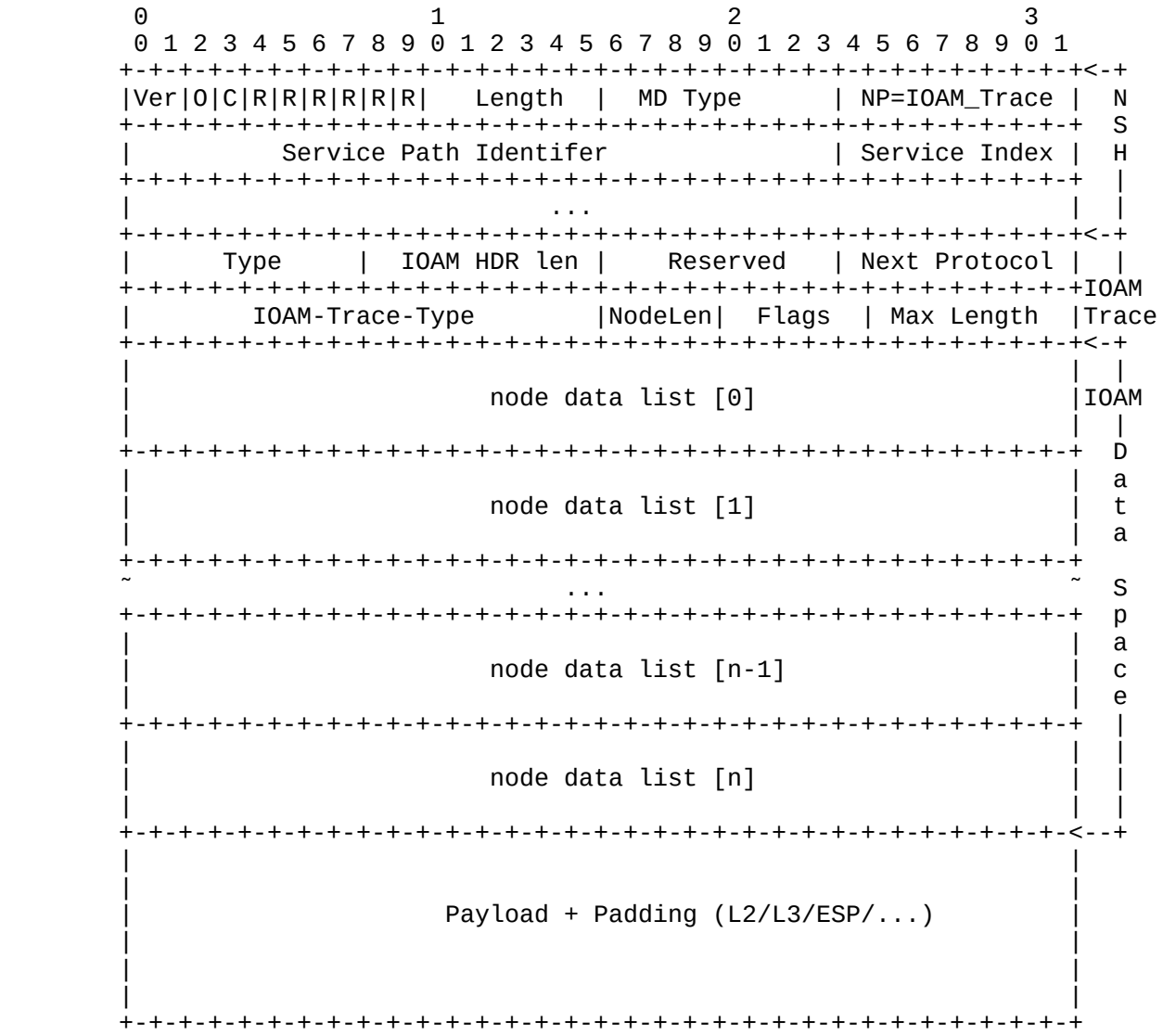
In Service Function Chaining (SFC) [RFC7665], the Network Service Header (NSH) [I-D.ietf-sfc-nsh] already includes path tracing capabilities [I-D.penno-sfc-trace]. Tracing information can be carried in-situ as IOAM data fields following NSH MDx metadata TLVs.

In-situ OAM Trace header following NSH MDx header
(Pre-allocated IOAM trace):



In-situ OAM Pre-allocated Trace Option Data MUST be 4-octet aligned:

In-situ OAM Trace header following NSH MDx header
(Incremental IOAM trace):



In-situ OAM Incremental Trace Option Data MUST be 4-octet aligned:

Next Protocol of NSH: TBD value for IOAM_Trace.

Type: 8-bit unsigned integer defining IOAM header type IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined here.

IOAM HDR len: 8-bit unsigned integer. Length of the in-situ OAM HDR in 8-octet units.

Reserved bits and R bits: Reserved bits are present for future use. The reserved bits MUST be set to 0x0.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol.

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.brockners-inband-oam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Flags: 5-bit field as defined in [I-D.brockners-inband-oam-data].

Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

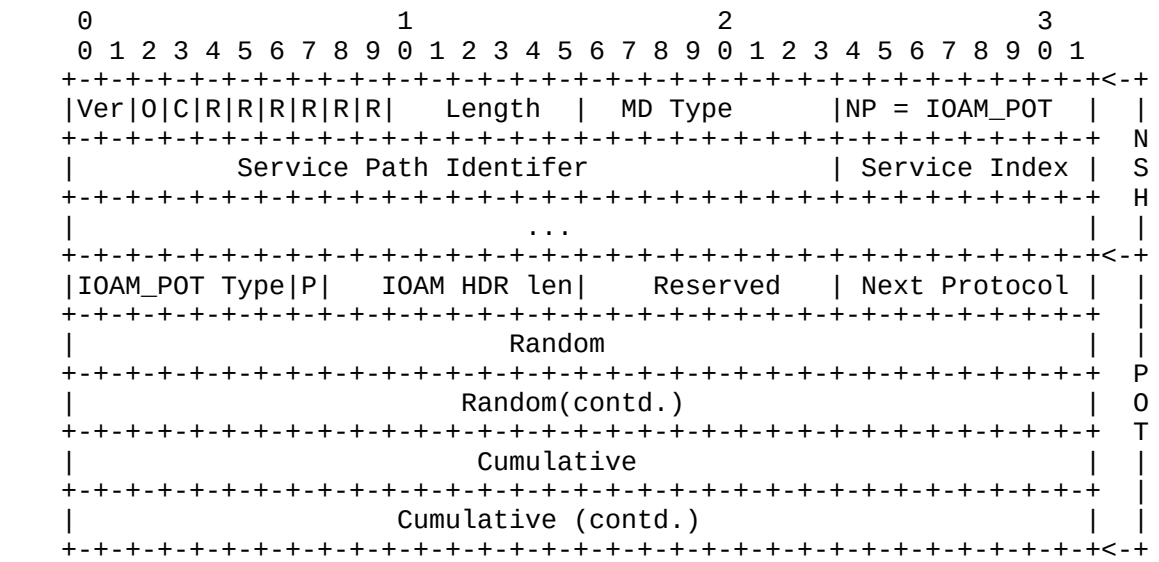
Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

6.2. In-situ OAM POT in NSH

The "Proof of Transit" capabilities (see [I-D.brockners-inband-oam-requirements] and [I-D.brockners-proof-of-transit]) of in-situ OAM can be leveraged within NSH. In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM data into the NSH header is enabled at the required nodes (i.e. at the in-situ OAM encapsulating/ decapsulating nodes) by means of configuration.

Proof of transit in-situ OAM data is added as NSH Type 2 metadata:

In-situ OAM POT header following NSH MDx header:



Next Protocol of NSH: TBD value for IOAM_POT.

IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

IOAM HDR len: 8-bit unsigned integer. Length of the in-situ OAM HDR in 8-octet units

Reserved bits and R bits: Reserved bits are present for future use. The reserved bits MUST be set to 0x0.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol.

Random: 64-bit Per-packet random number.

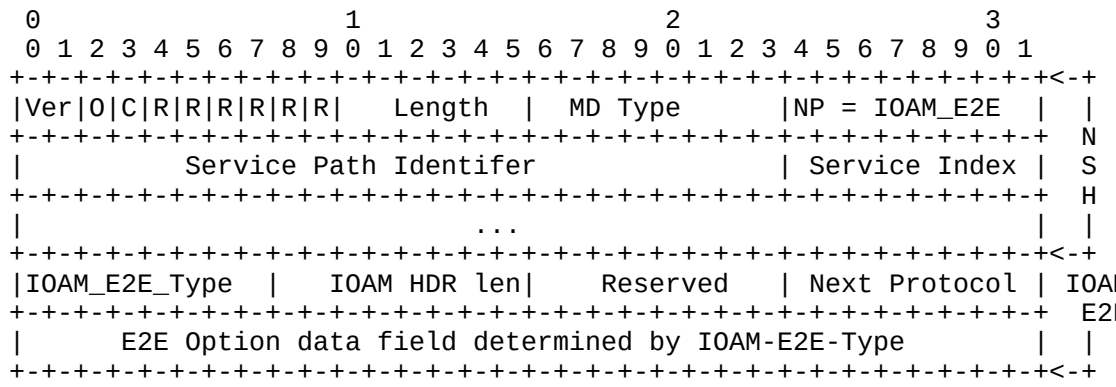
Cumulative: 64-bit Cumulative value that is updated by the Service Functions.

6.3. In-situ OAM Edge-to-Edge in NSH

The "Edge-to-Edge" capabilities (see [I-D.brockners-inband-oam-requirements]) of in-situ OAM can be leveraged within NSH. In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM data into the NSH header is enabled at the required nodes (i.e. at the in-situ OAM encapsulating/ decapsulating nodes) by means of configuration.

Edge-to-Edge in-situ OAM data is added as a TLV following NSH MDx metadata:

In-situ OAM E2E header following NSH MDx header:



Next Protocol of NSH: TBD value for IOAM_E2E.

IOAM E2E Type: 8-bit identifier of a particular E2E variant that dictates the IOAM E2E data that is included as defined in [I-D.brockners-inband-oam-data].

IOAM HDR len: 8-bit unsigned integer. Length of the in-situ OAM HDR in 8-octet units

Reserved bits and R bits: Reserved bits are present for future use. The reserved bits MUST be set to 0x0.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol.

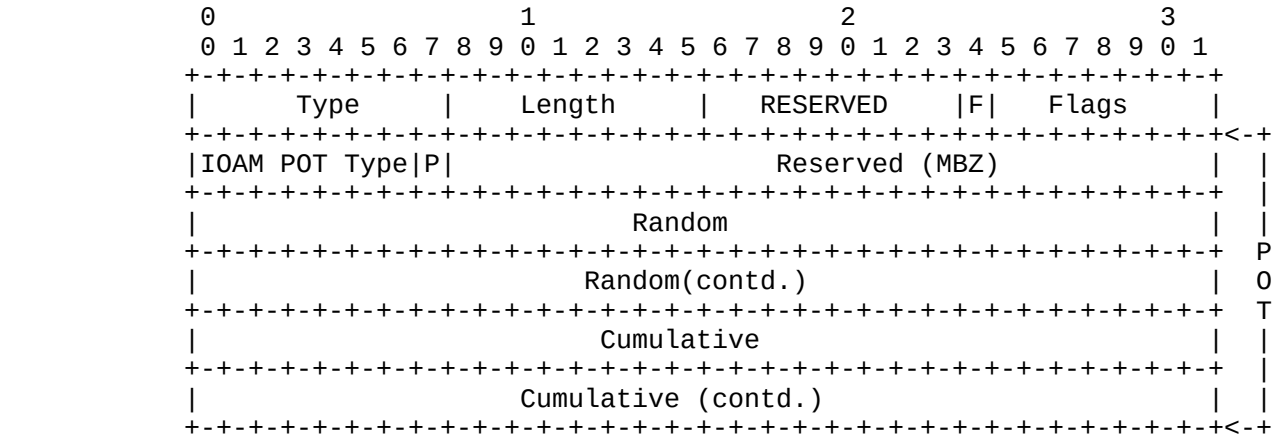
E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

7. In-situ OAM Metadata Transport in Segment Routing

7.1. In-situ OAM in SR with IPv6 Transport

Similar to NSH, a policy defined using Segment Routing for IPv6 can be verified using the in-situ OAM "Proof of Transit" approach. The Segment Routing Header (SRH) for IPv6 offers the ability to transport TLV structured data, similar to what NSH does (see [I-D.ietf-6man-segment-routing-header]). In an domain where in-situ OAM is used, insertion of the in-situ OAM data is enabled at the required edge nodes (i.e. at the in-situ OAM encapsulating/ decapsulating nodes) by means of configuration.

A new "POT TLV" is defined for the SRH which is to carry proof of transit in situ OAM data.



Type: To be assigned by IANA.

Length: 20.

RESERVED: 8 bits. SHOULD be unset on transmission and MUST be ignored on receipt.

F: 1 bit. Indicates which POT-profile is active. 0 means the even POT-profile is active, 1 means the odd POT-profile is active.

Flags: 8 bits. No flags are defined in this document.

IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

Reserved (MBZ): 24-bit field MUST be filled with zeroes.

Random: 64-bit per-packet random number.

Cumulative: 64-bit cumulative value that is updated at specific nodes that form the service path to be verified.

7.2. In-situ OAM in SR with MPLS Transport

In-situ OAM "Proof of Transit" data can also be carried as part of the MPLS label stack. Details will be addressed in a future version of this document.

8. IANA Considerations

IANA considerations will be added in a future version of this document.

9. Manageability Considerations

Manageability considerations will be addressed in a later version of this document..

10. Security Considerations

Security considerations will be addressed in a later version of this document. For a discussion of security requirements of in-situ OAM, please refer to [I-D.brockners-inband-oam-requirements].

11. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, and Andrew Yourtchenko for the comments and advice. The authors would like to acknowledge Craig Hill for contributing GRE IOAM encapsulation. For the IPv6 encapsulation, this document leverages and builds on top of several concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

12. References

12.1. Normative References

[ETYPES] "IANA Ethernet Numbers", <https://www.iana.org/assignments/ethernet-numbers/ethernet-numbers.xhtml>.

[I-D.brockners-inband-oam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., <>, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-brockners-inband-oam-data-05 (work in progress), May 2017.

[I-D.brockners-inband-oam-requirements] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.ietf-6man-segment-routing-header] Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-06 (work in progress), March 2017.

[I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.

[I-D.ietf-sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-13 (work in progress), June 2017.

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

[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>.

[RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <http://www.rfc-editor.org/info/rfc3232>.

12.2. Informative References

[FD.io] "Fast Data Project: FD.io", <https://fd.io/>.

[I-D.brockners-proof-of-transit] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof of Transit", draft-brockners-proof-of-transit-03 (work in progress), March 2017.

[I-D.ietf-ippm-6man-pdm-option] Elkins, N., Hamilton, R., and m. mackermann@bcbsm.com, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", draft-ietf-ippm-6man-pdm-option-13 (work in progress), June 2017.

[I-D.ietf-spring-segment-routing] Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

[I-D.kitamura-ipv6-record-route] Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.

[I-D.penno-sfc-trace] Penno, R., Quinn, P., Pignataro, C., and D. Zhou, "Services Function Chaining Traceroute", draft-penno-sfc-trace-03 (work in progress), September 2015.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Vengada Prasad Govindan
Cisco Systems, Inc.

Email: venggovi@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Petr Lapukhov
Facebook
1 Hacker Way
Menlo Park, CA 94025
US

Email: petr@fb.com

Remy Chang
Barefoot Networks
2185 Park Boulevard
Palo Alto, CA 94306
US

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 8, 2018

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
J. Leddy
Comcast
S. Youell
JPMC
D. Mozes

T. Mizrahi
Marvell

May 7, 2018

Proof of Transit
draft-brockners-proof-of-transit-05

Abstract

Several technologies such as Traffic Engineering (TE), Service Function Chaining (SFC), and policy based routing are used to steer traffic through a specific, user-defined path. This document defines mechanisms to securely prove that traffic transited said defined path. These mechanisms allow to securely verify whether, within a given path, all packets traversed all the nodes that they are supposed to visit.

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 November 8, 2018.

Table of Contents

1. Introduction

Several deployments use Traffic Engineering, policy routing, Segment Routing (SR), and Service Function Chaining (SFC) [RFC7665] to steer packets through a specific set of nodes. In certain cases, regulatory obligations or a compliance policy require operators to prove that all packets that are supposed to follow a specific path are indeed being forwarded across and exact set of pre-determined nodes.

If a packet flow is supposed to go through a series of service functions or network nodes, it has to be proven that indeed all packets of the flow followed the path or service chain or collection of nodes specified by the policy. In case some packets of a flow weren't appropriately processed, a verification device should determine the policy violation and take corresponding actions corresponding to the policy (e.g., drop or redirect the packet, send an alert etc.) In today's deployments, the proof that a packet traversed a particular path or service chain is typically delivered in an indirect way: Service appliances and network forwarding are in different trust domains. Physical hand-off-points are defined between these trust domains (i.e. physical interfaces). Or in other terms, in the "network forwarding domain" things are wired up in a way that traffic is delivered to the ingress interface of a service appliance and received back from an egress interface of a service appliance. This "wiring" is verified and then trusted upon. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using technologies such as Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the

hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. As a consequence, different trust layers should not to be mixed in the same device. For an NFV scenario a different type of proof is required. Offering a proof that a packet indeed traversed a specific set of service functions or nodes allows operators to evolve from the above described indirect methods of proving that packets visit a predetermined set of nodes.

The solution approach presented in this document is based on a small portion of operational data added to every packet. This "in-situ" operational data is also referred to as "proof of transit data", or POT data. The POT data is updated at every required node and is used to verify whether a packet traversed all required nodes. A particular set of nodes "to be verified" is either described by a set of secret keys, or a set of shares of a single secret. Nodes on the path retrieve their individual keys or shares of a key (using for e.g., Shamir's Secret Sharing scheme) from a central controller. The complete key set is only known to the controller and a verifier node, which is typically the ultimate node on a path that performs verification. Each node in the path uses its secret or share of the secret to update the POT data of the packets as the packets pass through the node. When the verifier receives a packet, it uses its key(s) along with data found in the packet to validate whether the packet traversed the path correctly.

2. Conventions

Abbreviations used in this document:

HMAC: Hash based Message Authentication Code. For example, HMAC-SHA256 generates 256 bits of MAC

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

POT-profile: Proof of Transit Profile that has the necessary data for nodes to participate in proof of transit

RND: Random Bits generated per packet. Packet fields that donot change during the traversal are given as input to HMAC-256 algorithm. A minimum of 32 bits (left most) need to be used from the output if RND is used to verify the packet integrity. This is a standard recommendation by NIST.

SEQ_NO: Sequence number initialized to a predefined constant. This is used in concatenation with RND bits to mitigate different attacks discussed later.

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

This section discusses methods and algorithms to provide for a "proof of transit" for packets traversing a specific path. A path which is to be verified consists of a set of nodes. Transit of the data packets through those nodes is to be proven. Besides the nodes, the setup also includes a Controller that creates secrets and secrets shares and configures the nodes for POT operations.

The methods how traffic is identified and associated to a specific path is outside the scope of this document. Identification could be done using a filter (e.g., 5-tuple classifier), or an identifier which is already present in the packet (e.g., path or service identifier, NSH Service Path Identifier (SPI), flow-label, etc.)

The solution approach is detailed in two steps. Initially the concept of the approach is explained. This concept is then further refined to make it operationally feasible.

3.1. Basic Idea

The method relies on adding POT data to all packets that traverse a path. The added POT data allows a verifying node (egress node) to check whether a packet traversed the identified set of nodes on a path correctly or not. Security mechanisms are natively built into the generation of the POT data to protect against misuse (i.e. configuration mistakes, malicious administrators playing tricks with routing, capturing, spoofing and replaying packets). The mechanism for POT leverages "Shamir’s Secret Sharing" scheme [SSS].

Shamir’s secret sharing base idea: A polynomial (represented by its coefficients) is chosen as a secret by the controller. A polynomial represents a curve. A set of well-defined points on the curve are

needed to construct the polynomial. Each point of the polynomial is called "share" of the secret. A single secret is associated with a particular set of nodes, which typically represent the path, to be verified. Shares of the single secret (i.e., points on the curve) are securely distributed from a Controller to the network nodes. Nodes use their respective share to update a cumulative value in the POT data of each packet. Only a verifying node has access to the complete secret. The verifying node validates the correctness of the received POT data by reconstructing the curve.

The polynomial cannot be constructed if any of the points are missed or tampered. Per Shamir's Secret Sharing Scheme, any lesser points means one or more nodes are missed. Details of the precise configuration needed for achieving security are discussed further below.

While applicable in theory, a vanilla approach based on Shamir's secret sharing could be easily attacked. If the same polynomial is reused for every packet for a path a passive attacker could reuse the value. As a consequence, one could consider creating a different polynomial per packet. Such an approach would be operationally complex. It would be complex to configure and recycle so many curves and their respective points for each node. Rather than using a single polynomial, two polynomials are used for the solution approach: A secret polynomial which is kept constant, and a per-packet polynomial which is public. Operations are performed on the sum of those two polynomials - creating a third polynomial which is secret and per packet.

3.2. Solution Approach

Solution approach: The overall algorithm uses two polynomials: POLY-1 and POLY-2. POLY-1 is secret and constant. Each node gets a point on POLY-1 at setup-time and keeps it secret. POLY-2 is public, random and per packet. Each node generates a point on POLY-2 each time a packet crosses it. Each node then calculates (point on POLY-1 + point on POLY-2) to get a (point on POLY-3) and passes it to verifier by adding it to each packet. The verifier constructs POLY-3 from the points given by all the nodes and cross checks whether POLY-3 = POLY-1 + POLY-2. Only the verifier knows POLY-1. The solution leverages finite field arithmetic in a field of size "prime number".

Detailed algorithms are discussed next. A simple example is discussed in Section 3.3.

3.2.1. Setup

A controller generates a first polynomial (POLY-1) of degree k and k+1 points on the polynomial. The constant coefficient of POLY-1 is considered the SECRET. The non-constant coefficients are used to generate the Lagrange Polynomial Constants (LPC). Each of the k nodes (including verifier) are assigned a point on the polynomial i.e., shares of the SECRET. The verifier is configured with the SECRET. The Controller also generates coefficients (except the constant coefficient, called "RND", which is changed on a per packet basis) of a second polynomial POLY-2 of the same degree. Each node is configured with the LPC of POLY-2. Note that POLY-2 is public.

3.2.2. In Transit

For each packet, the ingress node generates a random number (RND). It is considered as the constant coefficient for POLY-2. A cumulative value (CML) is initialized to 0. Both RND, CML are carried as within the packet POT data. As the packet visits each node, the RND is retrieved from the packet and the respective share of POLY-2 is calculated. Each node calculates (Share(POLY-1) + Share(POLY-2)) and CML is updated with this sum. This step is performed by each node until the packet completes the path. The verifier also performs the step with its respective share.

3.2.3. Verification

The verifier cross checks whether CML = SECRET + RND. If this matches then the packet traversed the specified set of nodes in the path. This is due to the additive homomorphic property of Shamir’s Secret Sharing scheme.

3.3. Illustrative Example

This section shows a simple example to illustrate step by step the approach described above.

3.3.1. Basic Version

Assumption: It is to be verified whether packets passed through 3 nodes. A polynomial of degree 2 is chosen for verification.

Choices: Prime = 53. POLY-1(x) = (3x^2 + 3x + 10) mod 53. The secret to be re-constructed is the constant coefficient of POLY-1, i.e., SECRET=10. It is important to note that all operations are done over a finite field (i.e., modulo prime).

3.3.1.1. Secret Shares

The shares of the secret are the points on POLY-1 chosen for the 3 nodes. For example, let x0=2, x1=4, x2=5.

POLY-1(2) = 28 => (x0, y0) = (2, 28)

POLY-1(4) = 17 => (x1, y1) = (4, 17)

POLY-1(5) = 47 => (x2, y2) = (5, 47)

The three points above are the points on the curve which are considered the shares of the secret. They are assigned to three nodes respectively and are kept secret.

3.3.1.2. Lagrange Polynomials

Lagrange basis polynomials (or Lagrange polynomials) are used for polynomial interpolation. For a given set of points on the curve Lagrange polynomials (as defined below) are used to reconstruct the curve and thus reconstruct the complete secret.

l0(x) = (((x-x1) / (x0-x1)) * ((x-x2)/x0-x2))) mod 53 =
(((x-4) / (2-4)) * ((x-5)/2-5))) mod 53 =
(10/3 - 3x/2 + (1/6)x^2) mod 53

l1(x) = (((x-x0) / (x1-x0)) * ((x-x2)/x1-x2))) mod 53 =
(-5 + 7x/2 - (1/2)x^2) mod 53

l2(x) = (((x-x0) / (x2-x0)) * ((x-x1)/x2-x1))) mod 53 =
(8/3 - 2 + (1/3)x^2) mod 53

3.3.1.3. LPC Computation

Since x0=2, x1=4, x2=5 are chosen points. Given that computations are done over a finite arithmetic field ("modulo a prime number"), the Lagrange basis polynomial constants are computed modulo 53. The Lagrange Polynomial Constant (LPC) would be 10/3 , -5 , 8/3.

LPC(x0) = (10/3) mod 53 = 21

LPC(x1) = (-5) mod 53 = 48

LPC(x2) = (8/3) mod 53 = 38

For a general way to compute the modular multiplicative inverse, see e.g., the Euclidean algorithm.

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

POLY1(x) = l0(x) * y0 + l1(x) * y1 + l2(x) * y2

Subsequently, the SECRET, which is the constant coefficient of POLY1(x) can be computed as below

SECRET = (y0*LPC(l0)+y1*LPC(l1)+y2*LPC(l2)) mod 53

The secret can be easily reconstructed using the y-values and the LPC:

SECRET = (y0*LPC(l0) + y1*LPC(l1) + y2*LPC(l2)) mod 53 = mod (28 * 21 + 17 * 48 + 47 * 38) mod 53 = 3190 mod 53 = 10

One observes that the secret reconstruction can easily be performed cumulatively hop by hop. CML represents the cumulative value. It is the POT data in the packet that is updated at each hop with the node's respective (yi*LPC(i)), where i is their respective value.

3.3.1.5. Verification

Upon completion of the path, the resulting CML is retrieved by the verifier from the packet POT data. Recall that verifier is preconfigured with the original SECRET. It is cross checked with the CML by the verifier. Subsequent actions based on the verification failing or succeeding could be taken as per the configured policies.

3.3.2. Enhanced Version

As observed previously, the vanilla algorithm that involves a single secret polynomial is not secure. Therefore, the solution is further enhanced with usage of a random second polynomial chosen per packet.

3.3.2.1. Random Polynomial

Let the second polynomial POLY-2 be (RND + 7x + 10 x^2). RND is a random number and is generated for each packet. Note that POLY-2 is public and need not be kept secret. The nodes can be pre-configured with the non-constant coefficients (for example, 7 and 10 in this case could be configured through the Controller on each node). So precisely only RND value changes per packet and is public and the rest of the non-constant coefficients of POLY-2 kept secret.

3.3.2.2. Reconstruction

Recall that each node is preconfigured with their respective Share(POLY-1). Each node calculates its respective Share(POLY-2) using the RND value retrieved from the packet. The CML reconstruction is enhanced as below. At every node, CML is updated as

CML = CML+(((Share(POLY-1)+ Share(POLY-2)) * LPC) mod Prime

Let us observe the packet level transformations in detail. For the example packet here, let the value RND be 45. Thus POLY-2 would be (45 + 7x + 10x^2).

The shares that could be generated are (2, 46), (4, 21), (5, 12).

At ingress: The fields RND = 45. CML = 0.

At node-1 (x0): Respective share of POLY-2 is generated i.e., (2, 46) because share index of node-1 is 2.

CML = 0 + ((28 + 46)* 21) mod 53 = 17

At node-2 (x1): Respective share of POLY-2 is generated i.e., (4, 21) because share index of node-2 is 4.

CML = 17 + ((17 + 21)*48) mod 53 = 17 + 22 = 39

At node-3 (x2), which is also the verifier: The respective share of POLY-2 is generated i.e., (5, 12) because the share index of the verifier is 12.

CML = 39 + ((47 + 12)*38) mod 53 = 39 + 16 = 55 mod 53 = 2

The verification using CML is discussed in next section.

3.3.2.3. Verification

As shown in the above example, for final verification, the verifier compares:

VERIFY = (SECRET + RND) mod Prime, with Prime = 53 here

VERIFY = (RND-1 + RND-2) mod Prime = (10 + 45) mod 53 = 2

Since VERIFY = CML the packet is proven to have gone through nodes 1, 2, and 3.

3.3.3. Final Version

The enhanced version of the protocol is still prone to replay and preplay attacks. An attacker could reuse the POT metadata for bypassing the verification. So additional measures using packet integrity checks (HMAC) and sequence numbers (SEQ_NO) are discussed later "Security Considerations" section.

3.4. Operational Aspects

To operationalize this scheme, a central controller is used to generate the necessary polynomials, the secret share per node, the prime number, etc. and distributing the data to the nodes participating in proof of transit. The identified node that performs the verification is provided with the verification key. The information provided from the Controller to each of the nodes participating in proof of transit is referred to as a proof of transit profile (POT-profile). Also note that the set of nodes for which the transit has to be proven are typically associated to a different trust domain than the verifier. Note that building the trust relationship between the Controller and the nodes is outside the scope of this document. Techniques such as those described in [I-D.ietf-anima-autonomic-control-plane] might be applied.

To optimize the overall data amount of exchanged and the processing at the nodes the following optimizations are performed:

1. The points (x, y) for each of the nodes on the public and private polynomials are picked such that the x component of the points match. This lends to the LPC values which are used to calculate the cumulative value CML to be constant. Note that the LPC are only depending on the x components. They can be computed at the controller and communicated to the nodes. Otherwise, one would need to distributed the x components to all the nodes.

2. A pre-evaluated portion of the public polynomial for each of the nodes is calculated and added to the POT-profile. Without this all the coefficients of the public polynomial had to be added to the POT profile and each node had to evaluate them. As stated before, the public portion is only the constant coefficient RND value, the pre-evaluated portion for each node should be kept secret as well.

3. To provide flexibility on the size of the cumulative and random numbers carried in the POT data a field to indicate this is shared and interpreted at the nodes.

3.5. Alternative Approach

In certain scenarios preserving the order of the nodes traversed by the packet may be needed. An alternative, "nested encryption" based approach is described here for preserving the order

3.5.1. Basic Idea

1. The controller provisions all the nodes with their respective secret keys.

2. The controller provisions the verifier with all the secret keys of the nodes.

3. For each packet, the ingress node generates a random number RND and encrypts it with its secret key to generate CML value

4. Each subsequent node on the path encrypts CML with their respective secret key and passes it along

5. The verifier is also provisioned with the expected sequence of nodes in order to verify the order

6. The verifier receives the CML, RND values, re-encrypts the RND with keys in the same order as expected sequence to verify.

3.5.2. Pros

Nested encryption approach retains the order in which the nodes are traversed.

3.5.3. Cons

1. Standard AES encryption would need 128 bits of RND, CML. This results in a 256 bits of additional overhead is added per packet

2. In hardware platforms that do not support native encryption capabilities like (AES-NI). This approach would have considerable impact on the computational latency

4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data fields in every packet that should be verified:

1. RND: Random number (the constant coefficient of public polynomial)

2. CML: Cumulative

The size of the data fields determines how often a new set of polynomials would need to be created. At maximum, the largest RND number that can be represented with a given number of bits determines the number of unique polynomials POLY-2 that can be created. The table below shows the maximum interval for how long a single set of polynomials could last for a variety of bit rates and RND sizes: When choosing 64 bits for RND and CML data fields, the time between a renewal of secrets could be as long as 3,100 years, even when running at 100 Gbps.

Transfer rate	Secret/RND size	Max # of packets	Time RND lasts
1 Gbps	64	2^64 = approx. 2*10^19	approx. 310,000 years
10 Gbps	64	2^64 = approx. 2*10^19	approx. 31,000 years
100 Gbps	64	2^64 = approx. 2*10^19	approx. 3,100 years
1 Gbps	32	2^32 = approx. 4*10^9	2,200 seconds
10 Gbps	32	2^32 = approx. 4*10^9	220 seconds
100 Gbps	32	2^32 = approx. 4*10^9	22 seconds

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

A POT system consists of a number of nodes that participate in POT and a Controller, which serves as a control and configuration entity. The Controller is to create the required parameters (polynomials, prime number, etc.) and communicate those to the nodes. The sum of all parameters for a specific node is referred to as "POT-profile". This document does not define a specific protocol to be used between Controller and nodes. It only defines the procedures and the associated YANG data model.

5.1. Procedure

The Controller creates new POT-profiles at a constant rate and communicates the POT-profile to the nodes. The controller labels a POT-profile "even" or "odd" and the Controller cycles between "even" and "odd" labeled profiles. The rate at which the POT-profiles are communicated to the nodes is configurable and is more frequent than the speed at which a POT-profile is "used up" (see table above). Once the POT-profile has been successfully communicated to all nodes (e.g., all NETCONF transactions completed, in case NETCONF is used as a protocol), the controller sends an "enable POT-profile" request to the ingress node.

All nodes maintain two POT-profiles (an even and an odd POT-profile): One POT-profile is currently active and in use; one profile is standby and about to get used. A flag in the packet is indicating whether the odd or even POT-profile is to be used by a node. This is to ensure that during profile change the service is not disrupted. If the "odd" profile is active, the Controller can communicate the "even" profile to all nodes. Only if all the nodes have received the POT-profile, the Controller will tell the ingress node to switch to the "even" profile. Given that the indicator travels within the packet, all nodes will switch to the "even" profile. The "even" profile gets active on all nodes and nodes are ready to receive a new "odd" profile.

Unless the ingress node receives a request to switch profiles, it’ll continue to use the active profile. If a profile is "used up" the ingress node will recycle the active profile and start over (this could give rise to replay attacks in theory - but with 2^32 or 2^64 packets this isn’t really likely in reality).

5.2. YANG Model

This section defines that YANG data model for the information exchange between the Controller and the nodes.

<CODE BEGINS> file "ietf-pot-profile@2016-06-15.yang"

```
module ietf-pot-profile {

  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-pot-profile";

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";
```

```
     contact "";

     description "This module contains a collection of YANG
                  definitions for proof of transit configuration
                  parameters. The model is meant for proof of
                  transit and is targeted for communicating the
                  POT-profile between a controller and nodes
                  participating in proof of transit.";

     revision 2016-06-15 {
       description
         "Initial revision.";
       reference
         "";
     }

     typedef profile-index-range {
       type int32 {
         range "0 .. 1";
       }
       description
         "Range used for the profile index. Currently restricted to
          0 or 1 to identify the odd or even profiles.";
     }


     grouping pot-profile {
       description "A grouping for proof of transit profiles.";
       list pot-profile-list {
         key "pot-profile-index";
         ordered-by user;
         description "A set of pot profiles.";

         leaf pot-profile-index {
           type profile-index-range;
           mandatory true;
           description
             "Proof of transit profile index.";
         }

         leaf prime-number {
           type uint64;
           mandatory true;
           description
             "Prime number used for module math computation";
         }

         leaf secret-share {
```

```
          type uint64;
          mandatory true;
          description
            "Share of the secret of polynomial 1 used in computation";
        }

        leaf public-polynomial {
          type uint64;
          mandatory true;
          description
            "Pre evaluated Public polynomial";
        }

        leaf lpc {
          type uint64;
          mandatory true;
          description
            "Lagrange Polynomial Coefficient";
        }

        leaf validator {
          type boolean;
          default "false";
          description
            "True if the node is a verifier node";
        }

        leaf validator-key {
          type uint64;
          description
            "Secret key for validating the path, constant of poly 1";
        }

        leaf bitmask {
          type uint64;
          default 4294967295;
          description
            "Number of bits as mask used in controlling the size of the
             random value generation. 32-bits of mask is default.";
        }
      }
    }

    container pot-profiles {
      description "A group of proof of transit profiles.";

      list pot-profile-set {
        key "pot-profile-name";
```

```
        ordered-by user;
        description
          "Set of proof of transit profiles that group parameters
           required to classify and compute proof of transit
           metadata at a node";

        leaf pot-profile-name {
          type string;
          mandatory true;
          description
            "Unique identifier for each proof of transit profile";
        }

        leaf active-profile-index {
          type profile-index-range;
          description
            "Proof of transit profile index that is currently active.
             Will be set in the first hop of the path or chain.
             Other nodes will not use this field.";
        }

        uses pot-profile;
      }
    /*** Container: end ***/
    }
  /*** module: end ***/
  }
  <CODE ENDS>
```

6. IANA Considerations

IANA considerations will be added in a future version of this document.

7. Manageability Considerations

Manageability considerations will be addressed in a later version of this document.

8. Security Considerations

Different security requirements achieved by the solution approach are discussed here.

8.1. Proof of Transit

Proof of correctness and security of the solution approach is per Shamir's Secret Sharing Scheme [SSS]. Cryptographically speaking it achieves information-theoretic security i.e., it cannot be broken by an attacker even with unlimited computing power. As long as the below conditions are met it is impossible for an attacker to bypass one or multiple nodes without getting caught.

- If there are k+1 nodes in the path, the polynomials (POLY-1, POLY-2) should be of degree k. Also k+1 points of POLY-1 are chosen and assigned to each node respectively. The verifier can re-construct the k degree polynomial (POLY-3) only when all the points are correctly retrieved.

- Precisely three values are kept secret by individual nodes. Share of SECRET (i.e. points on POLY-1), Share of POLY-2, LPC, P. Note that only constant coefficient, RND, of POLY-2 is public. x values and non-constant coefficient of POLY-2 are secret

An attacker bypassing a few nodes will miss adding a respective point on POLY-1 to corresponding point on POLY-2 , thus the verifier cannot construct POLY-3 for cross verification.

Also it is highly recommended that different polynomials should be used as POLY-1 across different paths, traffic profiles or service chains.

8.2. Cryptanalysis

A passive attacker could try to harvest the POT data (i.e., CML, RND values) in order to determine the configured secrets. Subsequently two types of differential analysis for guessing the secrets could be done.

- Inter-Node: A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1. This is because at each point there are four unknowns (i.e. Share(POLY-1), Share(Poly-2) LPC and prime number P) and three known values (i.e. RND, CML-before, CML-after).

- Inter-Packets: A passive attacker could observe CML values across packets (i.e., values of PKT-1 and subsequent PKT-2), in order to predict the secrets. Differential analysis across packets could be mitigated using a good PRNG for generating RND. Note that if constant coefficient is a sequence number than CML values become quite predictable and the scheme would be broken.

8.3. Anti-Replay

A passive attacker could reuse a set of older RND and the intermediate CML values to bypass certain nodes in later packets. Such attacks could be avoided by carefully choosing POLY-2 as a (SEQ_NO + RND). For example, if 64 bits are being used for POLY-2 then first 16 bits could be a sequence number SEQ_NO and next 48 bits could be a random number.

Subsequently, the verifier could use the SEQ_NO bits to run classic anti-replay techniques like sliding window used in IPSEC. The verifier could buffer up to 2^16 packets as a sliding window. Packets arriving with a higher SEQ_NO than current buffer could be flagged legitimate. Packets arriving with a lower SEQ_NO than current buffer could be flagged as suspicious.

For all practical purposes in the rest of the document RND means SEQ_NO + RND to keep it simple.

The solution discussed in this memo does not currently mitigate replay attacks. An anti-replay mechanism may be included in future versions of the solution.

8.4. Anti-Preplay

An active attacker could try to perform a man-in-the-middle (MITM) attack by extracting the POT of PKT-1 and using it in PKT-2. Subsequently attacker drops the PKT-1 in order to avoid duplicate POT values reaching the verifier. If the PKT-1 reaches the verifier, then this attack is same as Replay attacks discussed before.

Preplay attacks are possible since the POT metadata is not dependent on the packet fields. Below steps are recommended for remediation:

- Ingress node and Verifier are configured with common pre shared key
- Ingress node generates a Message Authentication Code (MAC) from packet fields using standard HMAC algorithm.
- The left most bits of the output are truncated to desired length to generate RND. It is recommended to use a minimum of 32 bits.
- The verifier regenerates the HMAC from the packet fields and compares with RND. To ensure the POT data is in fact that of the packet.

If an HMAC is used, an active attacker lacks the knowledge of the pre-shared key, and thus cannot launch preplay attacks.

The solution discussed in this memo does not currently mitigate prereplay attacks. A mitigation mechanism may be included in future versions of the solution.

8.5. Anti-Tampering

An active attacker could not insert any arbitrary value for CML. This would subsequently fail the reconstruction of the POLY-3. Also an attacker could not update the CML with a previously observed value. This could subsequently be detected by using timestamps within the RND value as discussed above.

8.6. Recycling

The solution approach is flexible for recycling long term secrets like POLY-1. All the nodes could be periodically updated with shares of new SECRET as best practice. The table above could be consulted for refresh cycles (see Section 4).

8.7. Redundant Nodes and Failover

A "node" or "service" in terms of POT can be implemented by one or multiple physical entities. In case of multiple physical entities (e.g., for load-balancing, or business continuity situations - consider for example a set of firewalls), all physical entities which are implementing the same POT node are given that same share of the secret. This makes multiple physical entities represent the same POT node from an algorithm perspective.

8.8. Controller Operation

The Controller needs to be secured given that it creates and holds the secrets, as need to be the nodes. The communication between Controller and the nodes also needs to be secured. As secure communication protocol such as for example NETCONF over SSH should be chosen for Controller to node communication.

The Controller only interacts with the nodes during the initial configuration and thereafter at regular intervals at which the operator chooses to switch to a new set of secrets. In case 64 bits are used for the data fields "CML" and "RND" which are carried within the data packet, the regular intervals are expected to be quite long (e.g., at 100 Gbps, a profile would only be used up after 3100 years) - see Section 4 above, thus even a "headless" operation without a Controller can be considered feasible. In such a case, the

Controller would only be used for the initial configuration of the POT-profiles.

8.9. Verification Scope

The POT solution defined in this document verifies that a data-packet traversed or transited a specific set of nodes. From an algorithm perspective, a "node" is an abstract entity. It could be represented by one or multiple physical or virtual network devices, or is could be a component within a networking device or system. The latter would be the case if a forwarding path within a device would need to be securely verified.

8.9.1. Node Ordering

POT using Shamir’s secret sharing scheme as discussed in this document provides for a means to verify that a set of nodes has been visited by a data packet. It does not verify the order in which the data packet visited the nodes. In case the order in which a data packet traversed a particular set of nodes needs to be verified as well, alternate schemes that e.g., rely on "nested encryption" could to be considered.

8.9.2. Stealth Nodes

The POT approach discussed in this document is to prove that a data packet traversed a specific set of "nodes". This set could be all nodes within a path, but could also be a subset of nodes in a path. Consequently, the POT approach isn’t suited to detect whether "stealth" nodes which do not participate in proof-of-transit have been inserted into a path.

9. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Erik Nordmark, and Andrew Yourtchenko for the comments and advice.

10. References

10.1. Normative References

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[SSS] "Shamir's Secret Sharing", <https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing>.

10.2. Informative References

[I-D.ietf-anima-autonomic-control-plane] Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-03 (work in progress), July 2016.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Sashank Dara
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
BANGALORE, Bangalore, KARNATAKA 560 087
INDIA

Email: sadara@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

David Mozes

Email: mosesster@gmail.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: June 2017

Pierre Francois
Individual Contributor
A. Bashandy
C. Filsfils
Cisco Systems
Bruno Decraene
Stephane Litkowski
Orange

December 8, 2016

Topology Independent Fast Reroute using Segment Routing
draft-francois-rtgwg-segment-routing-ti-lfa-04

Abstract

This document presents Topology Independent Loop-free Alternate Fast Re-route (TI-LFA), aimed at providing protection of node and adjacency segments within the Segment Routing (SR) framework. This Fast Re-route (FRR) behavior builds on proven IP-FRR concepts being LFAs, remote LFAs (RLFA), and remote LFAs with directed forwarding (DLFA). It extends these concepts to provide guaranteed coverage in any IGP network. A key aspect of TI-LFA is the FRR path selection approach establishing protection over post-convergence paths from the point of local repair, dramatically reducing the operational need to control the tie-breaks among various FRR options.

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/ietf/1id-abstracts.txt

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

This Internet-Draft will expire on May 8, 2016.

Table of Contents

1. Introduction

Segment Routing aims at supporting services with tight SLA guarantees [1]. This document provides a local repair mechanism relying on SR-capable of restoring end-to-end connectivity in the case of a sudden failure of a network component.

For each destination in the network, TI-LFA prepares a data-plane switch-over to be activated upon detection of the failure of a link used to reach the destination. TI-LFA provides protection against link failure, node failure, and local SRLG failures. In link failure mode, the destination is protected assuming the failure of the link. In node protection mode, the destination is protected assuming that the neighbor connected to the primary link has failed. In local SRLG protecting mode, the destination is protected assuming that a configured set of links sharing fate with the primary link has failed (e.g. a linecard).

Using segment routing, there is no need to establish TLDP sessions with remote nodes in order to take advantage of the applicability of remote LFAs (RLFA) or remote LFAs with directed forwarding (DLFA)[2]. As a result, preferring LFAs over RLFAs or DLFAs, as well as minimizing the number of RLFA or DLFA repair nodes is not required. This allows for a protection path selection approach meeting operational needs rather than a topologically constrained one.

Using SR, there is no need to create state in the network in order to enforce an explicit FRR path. As a result, we can use optimized detour paths for each specific destination and for each type of failure without creating additional forwarding state. Also, the mode of protection (link, node, SRLG) is not constrained to be network wide or node wide, but can be managed on a per interface basis.

Building on such an easier forwarding environment, the FRR behavior suggested in this document tailors the repair paths over the post-convergence path from the PLR to the protected destination, given the enabled protection mode for the interface.

As the capacity of the post-convergence path is typically planned by the operator to support the post-convergence routing of the traffic for any expected failure, there is much less need for the operator to tune the decision among which protection path to choose. The protection path will automatically follow the natural backup path that would be used after local convergence. This also helps to reduce the amount of path changes and hence service transients: one transition (pre-convergence to post-convergence) instead of two (pre-convergence to FRR and then post-convergence).

We provide the TI-LFA approach that achieves guaranteed coverage against link, node, and local SRLG failure, in any IGP network, relying on the flexibility of SR.

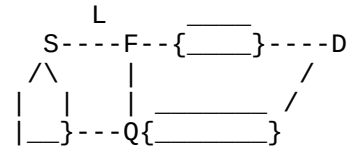


Figure 1 TI-LFA Protection

We use Figure 1 to illustrate the TI-LFA approach.

The Point of Local Repair (PLR), S, needs to find a node Q (a repair node) that is capable of safely forwarding the traffic to a destination D affected by the failure of the protected link L, a set of adjacent links including L (local SRLG), or the node F itself. The PLR also needs to find a way to reach Q without being affected by the convergence state of the nodes over the paths it wants to use to reach Q.

In Section 2 we define the main notations used in the document. They are in line with [2].

In Section 3, we suggest to compute the P-Space and Q-Space properties defined in Section 2, for the specific case of nodes lying over the post-convergence paths towards the protected destinations.

Using the properties defined in Section 3, we describe how to compute protection lists that encode a loopfree post-convergence towards the destination, in Section 4.

Finally, we define the segment operations to be applied by the PLR to ensure consistency with the forwarding state of the repair node, in Section 5.

1.1. 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 RFC-2119

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

2. Terminology

We define the main notations used in this document as the following.

We refer to "old" and "new" topologies as the LSDB state before and after the considered failure.

SPT_old(R) is the Shortest Path Tree rooted at node R in the initial state of the network.

SPT_new(R, X) is the Shortest Path Tree rooted at node R in the state of the network after the resource X has failed.

Dist_old(A,B) is the distance from node A to node B in SPT_old(A).

Dist_new(A,B, X) is the distance from node A to node B in SPT_new(A,X).

Similarly to [4], we rely on the concept of P-Space and Q-Space for TI-LFA.

The P-Space P(R,X) of a node R w.r.t. a resource X (e.g. a link S-F, a node F, or a local SRLG) is the set of nodes that are reachable from R without passing through X. It is the set of nodes that are not downstream of X in SPT_old(R).

The Extended P-Space P'(R,X) of a node R w.r.t. a resource X is the set of nodes that are reachable from R or a neighbor of R, without passing through X.

The Q-Space Q(D,X) of a destination node D w.r.t. a resource X is the set of nodes which do not use X to reach D in the initial state of the network. In other words, it is the set of nodes which have D in their P-Space w.r.t. S-F, F, or a set of links adjacent to S).

A symmetric network is a network such that the IGP metric of each link is the same in both directions of the link.

3. Intersecting P-Space and Q-Space with post-convergence paths

In this section, we suggest to determine the P-Space and Q-Space properties of the nodes along the post-convergence paths from the PLR to the protected destination and compute an SR-based explicit path from P to Q when they are not adjacent. Such properties will be used in Section 4 to compute the TI-LFA repair list.

3.1. P-Space property computation for a resource X

A node N is in P(R, X) if it is not downstream of X in SPT_old(R). X can be a link, a node, or a set of links adjacent to the PLR. A node N is in P'(R,X) if it is not downstream of X in SPT_old(N), for at least one neighbor N of R.

3.2. Q-Space property computation for a link S-F, over post-convergence paths

We want to determine which nodes on the post-convergence path from the PLR to the destination D are in the Q-Space of destination D w.r.t. link S-F.

This can be found by intersecting the post-convergence path to D, assuming the failure of S-F, with Q(D, S-F).

3.3. Q-Space property computation for a set of links adjacent to S, over post-convergence paths

We want to determine which nodes on the post-convergence path from the PLR to the destination D are in the Q-Space of destination D w.r.t. a set of links adjacent to S (S being the PLR). That is, we aim to find the set of nodes on the post-convergence path that use none of the members of the protected set of links, to reach D.

This can be found by intersecting the post-convergence path to D, assuming the failure of the set of links, with the intersection among Q(D, S->X) for all S->X belonging to the set of links.

3.4. Q-Space property computation for a node F, over post-convergence paths

We want to determine which nodes on the post-convergence from the PLR to the destination D are in the Q-Space of destination D w.r.t. node F.

This can be found by intersecting the post-convergence path to D, assuming the failure of F, with Q(D, F).

4. TI-LFA Repair Tunnel

The TI-LFA repair tunnel consists of an outgoing interface and a list of segments (repair list) to insert on the SR header. The repair list encodes the explicit post-convergence path to the destination, which avoids the protected resource X.

The TI-LFA repair tunnel is found by intersecting P(S,X) and Q(D,X) with the post-convergence path to D and computing the explicit SR-based path EP(P, Q) from P to Q when these nodes are not adjacent along the post convergence path. The TI-LFA repair list is expressed generally as (Node_SID(P), EP(P, Q)).

Most often, the TI-LFA repair list has a simpler form, as described in the following sections.

4.1. The repair node is a direct neighbor

When the repair node is a direct neighbor, the outgoing interface is set to that neighbor and the repair segment list is empty.

This is comparable to a post-convergence LFA FRR repair.

4.2. The repair node is a PQ node

When the repair node is in P(S,X), the repair list is made of a single node segment to the repair node.

This is comparable to a post-convergence RLFA repair tunnel.

4.3. The repair is a Q node, neighbor of the last P node

When the repair node is adjacent to P(S,X), the repair list is made of two segments: A node segment to the adjacent P node, and an adjacency segment from that node to the repair node.

This is comparable to a post-convergence DLFA repair tunnel.

4.4. Connecting distant P and Q nodes along post-convergence paths

In some cases, there is no adjacent P and Q node along the post-convergence path. However, the PLR can perform additional computations to compute a list of segments that represent a loopfree path from P to Q.

5. Protecting segments

In this section, we explain how a protecting router S processes the active segment of a packet upon the failure of its primary outgoing interface for the packet, S-F.

The behavior depends on the type of active segment to be protected.

5.1. The active segment is a node segment

The active segment is kept on the SR header, unchanged (1). The repair list is inserted at the head of the list. The active segment becomes the first segment of the inserted repair list.

Note (1): If the SRGB at the repair node is different from the SRGB at the PLR, then the active segment must be updated to fit the SRGB of the repair node.

In Section 5.3, we describe the node protection behavior of PLR S, for the specific case where the active segment is a prefix segment for the neighbor F itself.

5.2. The active segment is an adjacency segment

We define hereafter the FRR behavior applied by S for any packet received with an active adjacency segment S-F for which protection was enabled. We distinguish the case where this active segment is followed by another adjacency segment from the case where it is followed by a node segment.

5.2.1. Protecting [Adjacency, Adjacency] segment lists

If the next segment in the list is an Adjacency segment, then the packet has to be conveyed to F.

To do so, S applies a "NEXT" operation on Adj(S-F) and then two consecutive "PUSH" operations: first it pushes a node segment for F, and then it pushes a protection list allowing to reach F while bypassing S-F.

Upon failure of S-F, a packet reaching S with a segment list matching [adj(S-F),adj(M),...] will thus leave S with a segment list matching [RT(F),node(F),adj(M)], where RT(F) is the repair tunnel for destination F.

In Section 5.3.2, we describe the TI-LFA behavior of PLR S when node protection is applied and the two first segments are Adjacency Segments.

5.2.2. Protecting [Adjacency, Node] segment lists

If the next segment in the stack is a node segment, say for node T, the packet segment list matches [adj(S-F),node(T),...].

A first solution would consist in steering the packet back to F while avoiding S-F. To do so, S applies a "NEXT" operation on Adj(S-F) and then two consecutive "PUSH" operations: first it pushes a node segment for F, and then it pushes a repair list allowing to reach F while bypassing S-F.

Upon failure of S-F, a packet reaching S with a segment list matching [adj(S-F),node(T),...] will thus leave S with a segment list matching [RT(F),node(F),node(T)].

Another solution is to not steer the packet back via F but rather follow the new shortest path to T. In this case, S just needs to apply a "NEXT" operation on the Adjacency segment related to S-F, and push a repair list redirecting the traffic to a node Q, whose path to node segment T is not affected by the failure.

Upon failure of S-F, packets reaching S with a segment list matching [adj(L), node(T), ...], would leave S with a segment list matching [RT(Q),node(T), ...]. Note that this second behavior is the one followed for node protection, as described in Section 5.3.1.

5.3. Protecting SR policy midpoints against node failure

As planned in the previous version of this document, we describe the behavior of a node S configured to interpret the failure of link S->F as the node failure of F, in the specific case where the active segment of the packet received by S is a Prefix SID of F represented as "F"), or an Adjacency SID for the link S-F (represented as "S->F").

5.3.1. Protecting {F, T, D} or {S->F, T, D}

We describe the protection behavior of S when

1. the active segment is a prefix SID for a neighbor F, or an adjacency segment S->F

2. the primary interface used to forward the packet failed

3. the segment following the active segment is a prefix SID (for node T)

4. node protection is active for that interface.

The TILFA Node FRR behavior becomes equivalent to:

1. Pop; the segment F or S->F is removed

2. Confirm that the next segment is in the SRGB of F, meaning that the next segment is a prefix segment, e.g. for node T

3. Identify T (as per the SRGB of F)

4. Pop the next segment and push T’s segment based on the local SRGB

5. forward the packet according to T.

5.3.2. Protecting {F, F->T, D} or {S->F, F->T, D}

We describe the protection behavior of S when

1. the active segment is a prefix SID for a neighbor F, or an adjacency segment S->F

2. the primary interface used to forward the packet failed

3. the segment following the active segment is an adjacency SID (F->T)

4. node protection is active for that interface.

The TILFA Node FRR behavior becomes equivalent to:

1. Pop; the segment F or S->F is removed

2. Confirm that the next segment is an adjacency SID of F, say F->T

3. Identify T (as per the set of Adjacency Segments of F)

4. Pop the next segment and push T’s segment based on the local SRGB

5. forward the packet according to T.

6. Security Considerations

The behavior described in this document is internal functionality to a router that result in the ability to guarantee an upper bound on the time taken to restore traffic flow upon the failure of a directly connected link or node. As such no additional security risk is introduced by using the mechanisms proposed in this document.

7. IANA Considerations

No requirements for IANA

8. Conclusions

This document proposes a mechanism that is able to pre-calculate a backup path for every primary path so as to be able to protect against the failure of a directly connected link or node. The mechanism is able to calculate the backup path irrespective of the topology as long as the topology is sufficiently redundant.

9. References

9.1. Normative References

9.2. Informative References

[1] Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-08 (work in progress), May 2016.

[2] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, January 2010.

[3] Filsfils, C., Francois, P., Shand, M., Decraene, B., Uttaro, J., Leymann, N., and M. Horneffer, "Loop-Free Alternate (LFA) Applicability in Service Provider (SP) Networks", RFC 6571, June 2012.

[4] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <http://www.rfc-editor.org/info/rfc7490>.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Pierre Francois
pfrpfr@gmail.com

Ahmed Bashandy
Cisco Systems
170 West Tasman Dr, San Jose, CA 95134, USA
Email: bashandy@cisco.com

Clarence Filsfils
Cisco Systems
Brussels, Belgium
Email: cfilsfil@cisco.com

Bruno Decraene
Orange
Issy-les-Moulineaux
FR
Email: bruno.decraene@orange.com

Stephane Litkowski
Orange
FR
Email: stephane.litkowski@orange.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: June 2017

Pierre Francois
Individual Contributor
Clarence Filsfils
Ahmed Bashandy
Cisco Systems, Inc.
Stephane Litkowski
Orange

December 14, 2016

Loop avoidance using Segment Routing
draft-francois-rtgwg-segment-routing-uloop-01

Abstract

This document presents a mechanism aimed at providing loop avoidance in the case of IGP network convergence event. The solution relies on the temporary use of SR policies ensuring loop-freeness over the post-convergence paths from the converging node to the destination.

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

This Internet-Draft will expire on June 14, 2017.

Table of Contents

1. Introduction

Forwarding loops happen during the convergence of the IGP, as a result of transient inconsistency among forwarding states of the nodes of the network.

This document provides a mechanism leveraging Segment Routing to ensure loop-freeness during the IGP reconvergence process following a link-state change event.

We use Figure 1 to illustrate the mechanism. In this scenario, all the IGP link metrics are 1, excepted R3-R4 whose metric is 100. We consider the traffic from S to D.

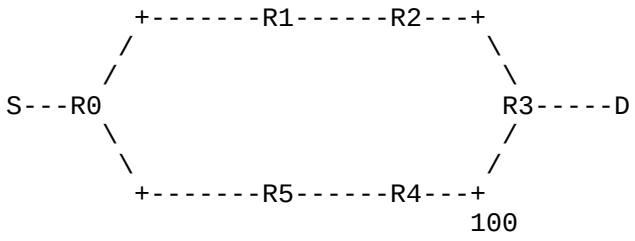


Figure 1 Illustrative scenario, failure of link R2-R3

When the link between R2 and R3 fails, traffic sent from S to D, initially flowing along S-R0-R1-R2-R3-D is subject to transient forwarding loops while routers update their forwarding state for destination D. For example, if R0 updates its FIB before R5, packets for D may loop between R0 and R5. If R5 updates its FIB before R4, packets for D may loop between R5 and R4.

Using segment routing, a headend can enforce an explicit path without creating any state along the desired path. As a result, a converging node can enforce traffic on the post-convergence path in a loop-free manner, using a list of segments (typically short). We suggest that the converging node enforces its post-convergence path to the destination when applying this behavior to ease operation (predictability of path, less capacity planning issues ...); nodes converge over their new optimal path, but temporarily use an SR policy to ensure loop-freeness over that path.

In our example, R0 can temporarily steer traffic destined to D over SR path [NodeSID(R4), AdjSID(R4->R3), D]. By doing so, packets for D will be forwarded by R5 as per NodeSID(R4), and by R4 as per AdjSID(R4->R3). From R3 on, the packet is forwarded as per destination D. As a result, traffic follows the desired path, regardless of the forwarding state for destination D at R5 and R4. After some time, the normal forwarding behavior (without using an SR policy) can be applied; routers will converge to their final forwarding state, still consistently forwarding along the post-convergence paths across the network.

1.1. 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 RFC-2119

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

2. Loop-free two-stage convergence process

Upon a topology change, when a node R converging for destination D does not trust the loop-freeness of its post-convergence path for destination D, it applies the following two-stage convergence process for destination D.

Stage 1: After computing the new path to D, for a configured amount of time C, R installs a FIB entry for D that steers packets to D via a loop-free SR path. C is assumed to be configured as per the worst-case convergence time of a node, network-wide. The SR path is computed when the event occurs.

Stage 2: After C elapses, R installs the normal post-convergence FIB entry for D, i.e. without any additional segments inserted that ensure the loop-free property.

Loop-freeness is ensured during this process, because:

1. Paths made of non up-to-date routers are loop-free.

Routers which forward as per the initial state of the network are consistent.

2. A packet reaching a node in stage 1 is ensured to reach its destination.

When a packet reaches a router in stage 1, it is steered on a SR path ensuring a loop-free post-convergence path, whatever the state of other routers on the path.

3. Paths made of a mix of routers in stage 1 and stage 2 are consistent.

After C milliseconds, all routers are forwarding as per their post-convergence paths, either expressed classically or as a loop-free SR path.

In our example, when R2-R3 fails, R0 forwards traffic for destination D over SR Path [NodeSID(R4), AdjSID(R4->R3), D], for C milliseconds. During that period, packets sent by R0 to D are loop-free as per the application of the policy. When C elapses, R0 now uses its normal post-convergence path to the destination, forwarding packets for D as is to R5.

R5 also implements loop avoidance, and has thus temporarily used a loop-avoiding SR policy for D. This policy is [AdjSID(R4->R3), D], oif R5->R4. If R5 is still applying the stage 1 behavior, the packet will be forwarded using this policy, and will thus safely reach the destination. If R5 also had moved to stage 2, it forwards the packet as per its normal post-convergence path, via R4. The forwarding state of R4 for D at stage 1 and stage 2 are the same: oif R4->R3, as forwarding packets for destination D as is to R3 ensures a loop-free post-convergence path.

3. Computing loop-avoiding SR policies

The computation to turn a post-convergence path into a loop-free list of segments is outside the scope of this document. It is a local behavior at a node.

In a future revision of this document, we may provide a reference approach to compute loop-avoiding policies for link up, link metric increase, link down, link metric decrease, node up, and node down events. TI-LFA Repair Tunnel

4. Analysis

In this section, we review the main characteristics of the proposed solution. These characteristics are illustrated in [3].

4.1. Incremental Deployment

There is no requirement for a full network upgrade to get benefits from the solution.

(1) Nodes that are upgraded bring benefit for the traffic passing through them.

(2) Nodes that are not upgraded to support SR-based loop-avoidance will cause the micro-loops that they were causing before, unless they get avoided by the local behavior of a node supporting the behavior.

4.2. No impact on capacity planning

By ensuring loop-free post-convergence paths, the behavior remains in line with the natural expected convergence process of the IGP. Enabling SR-based loop-avoidance hence does not require consideration for capacity planning, compared to any loop avoidance mechanism that lets traffic follow a different path than the post-convergence one. The behavior is local. Nothing is expected from remote nodes except the basic support of Prefix and Adjacency SID’s.

5. Security Considerations

The behavior described in this document is internal functionality to a router that result in the ability to explicitly steer traffic over the post convergence path after a remote topology change in a manner that guarantees loop freeness. As such no additional security risk is introduced by using the mechanisms proposed in this document.

6. IANA Considerations

No requirements for IANA

7. Contributors

Additional contributors: Bruno Decraene and Peter Psenak.

8. References

8.1. Normative References

8.2. Informative References

[1] Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-10 (work in progress), November 2016.

[2] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, January 2010.

[3] Litkowski, S., "Avoiding micro-loops using Segment Routing", MPLS World Congress , 2016.

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Pierre Francois
pfrpfr@gmail.com

Ahmed Bashandy
Cisco Systems
170 West Tasman Dr, San Jose, CA 95134, USA
Email: bashandy@cisco.com

Clarence Filsfils
Cisco Systems
Brussels, Belgium
Email: cfilsfil@cisco.com

Stephane Litkowski
Orange
Email: stephane.litkowski@orange.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2017

A. Lindem, Ed.
Cisco Systems
Y. Qu
Huawei
D. Yeung
Arrcus, Inc
I. Chen
Jabil
J. Zhang
Juniper Networks

April 29, 2017

Routing Key Chain YANG Data Model
draft-ietf-rtgwg-yang-key-chain-24.txt

Abstract

This document describes the key chain YANG data model. Key chains are commonly used for routing protocol authentication and other applications requiring symmetric keys. A key chain is a list of elements each containing a key string, send lifetime, accept lifetime, and algorithm (authentication or encryption). By properly overlapping the send and accept lifetimes of multiple key chain elements, key strings and algorithms may be gracefully updated. By representing them in a YANG data model, key distribution can be automated.

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 October 31, 2017.

Table of Contents

1. Introduction

This document describes the key chain YANG [YANG-1.1] data model. Key chains are commonly used for routing protocol authentication and other applications requiring symmetric keys. A key chain is a list of elements each containing a key string, send lifetime, accept lifetime, and algorithm (authentication or encryption). By properly

overlapping the send and accept lifetimes of multiple key chain elements, key strings and algorithms may be gracefully updated. By representing them in a YANG data model, key distribution can be automated.

In some applications, the protocols do not use the key chain element key directly, but rather a key derivation function is used to derive a short-lived key from the key chain element key (e.g., the Master Keys used in [TCP-AO]).

1.1. Requirements Notation

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

1.2. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Section 3.3. The following tree notation is used.

- Brackets "[" and "]" enclose YANG list keys. These YANG list keys should not be confused with the key-chain keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations, and "-n" notifications.
- Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Problem Statement

This document describes a YANG [YANG-1.1] data model for key chains. Key chains have been implemented and deployed by a large percentage of network equipment vendors. Providing a standard YANG model will facilitate automated key distribution and non-disruptive key

rollover. This will aid in tightening the security of the core routing infrastructure as recommended in [IAB-REPORT].

A key chain is a list containing one or more elements containing a Key ID, key string, send/accept lifetimes, and the associated authentication or encryption algorithm. A key chain can be used by any service or application requiring authentication or encryption using symmetric keys. In essence, the key-chain is a reusable key policy that can be referenced wherever it is required. The key-chain construct has been implemented by most networking vendors and deployed in many networks.

A conceptual representation of a crypto key table is described in [CRYPTO-KEYTABLE]. The crypto key table also includes keys as well as their corresponding lifetimes and algorithms. Additionally, the key table includes key selection criteria and envisions a deployment model where the details of the applications or services requiring authentication or encryption permeate into the key database. The YANG key-chain model described herein doesn’t include key selection criteria or support this deployment model. At the same time, it does not preclude it. The draft [YANG-CRYPTO-KEYTABLE] describes augmentations to the key chain YANG model in support of key selection criteria.

2.1. Applicability

Other YANG modules may reference ietf-key-chain YANG module key-chain names for authentication and encryption applications. A YANG type has been provided to facilitate reference to the key-chain name without having to specify the complete YANG XML Path Language (XPath) selector.

2.2. Graceful Key Rollover using Key Chains

Key chains may be used to gracefully update the key string and/or algorithm used by an application for authentication or encryption. To achieve graceful key rollover, the receiver MAY accept all the keys that have a valid accept lifetime and the sender MAY send the key with the most recent send lifetime. One scenario for facilitating key rollover is to:

1. Distribute a key chain with a new key to all the routers or other network devices in the domain of that key chain. The new key’s accept lifetime should be such that it is accepted during the key rollover period. The send lifetime should be a time in the future when it can be assured that all the routers in the domain of that key are upgraded. This will have no immediate impact on the keys used for transmission.

2. Assure that all the network devices have been updated with the updated key chain and that their system times are roughly synchronized. The system times of devices within an administrative domain are commonly synchronized (e.g., using Network Time Protocol (NTP) [NTP-PROTO]). This also may be automated.

3. When the send lifetime of the new key becomes valid, the network devices within the domain of key chain will using the new key for transmissions.

4. At some point in the future, a new key chain with the old key removed may be distributed to the network devices within the domain of the key chain. However, this may be deferred until the next key rollover. If this is done, the key chain will always include two keys; either the current and future key (during key rollovers) or the current and previous keys (between key rollovers).

Since the most recent send lifetime is defined as the one with the latest start-time, specification of "always" will prevent using the graceful key rollover technique described above. Other key configuration and usage scenarios are possible but these are beyond the scope of this document.

3. Design of the Key Chain Model

The ietf-key-chain module contains a list of one or more keys indexed by a Key ID. For some applications (e.g., OSPFv3 [OSPFV3-AUTH]), the Key ID is used to identify the key chain key to be used. In addition to the Key ID, each key chain key includes a key-string and a cryptographic algorithm. Optionally, the key chain keys include send/accept lifetimes. If the send/accept lifetime is unspecified, the key is always considered valid.

Note that different key values for transmission versus acceptance may be supported with multiple key chain elements. The key used for transmission will have a valid send-lifetime and invalid accept-lifetime (e.g., has an end-time equal to the start-time). The key used for acceptance will have a valid accept-lifetime and invalid send-lifetime.

Due to the differences in key chain implementations across various vendors, some of the data elements are optional. Finally, the crypto algorithm identities are provided for reuse when configuring legacy authentication and encryption not using key-chains.

A key-chain is identified by a unique name within the scope of the network device. The "key-chain-ref" typedef SHOULD be used by other YANG modules when they need to reference a configured key-chain.

3.1. Key Chain Operational State

The key chain operational state is included in the same tree as key chain configuration consistent with Network Management Datastore Architecture [NMDA]. The timestamp of the last key chain modification is also maintained in the operational state. Additionally, the operational state includes an indication of whether or not a key chain key is valid for sending or acceptance.

3.2. Key Chain Model Features

Features are used to handle differences between vendor implementations. For example, not all vendors support configuration of an acceptance tolerance or configuration of key strings in hexadecimal. They are also used to support of security requirements (e.g., TCP-AO Algorithms [TCP-AO-ALGORITHMS]) not yet implemented by vendors or only a single vendor.

It is common for an entity with sufficient permissions to read and store a device’s configuration which would include the contents of this model. To avoid unnecessarily seeing and storing the keys in clear-text, this model provides the aes-key-wrap feature. More details are described in Security Considerations Section 5.

3.3. Key Chain Model Tree

```
+--rw key-chains
   +--rw key-chain* [name]
   |  +--rw name                        string
   |  +--rw description?                string
   |  +--rw accept-tolerance {accept-tolerance}?
   |  |  +--rw duration?   uint32
   |  +--ro last-modified-timestamp?   yang:date-and-time
   |  +--rw key* [key-id]
   |     +--rw key-id                     uint64
   |     +--rw lifetime
   |     |  +--rw (lifetime)?
   |     |     +--:(send-and-accept-lifetime)
   |     |     |  +--rw send-accept-lifetime
   |     |     |     +--rw (lifetime)?
   |     |     |        +--:(always)
   |     |     |        |  +--rw always?            empty
   |     |     |        +--:(start-end-time)
   |     |     |           +--rw start-date-time?
```

```
|     |     |           |       yang:date-and-time
|     |     |           +--rw (end-time)?
|     |     |              +--:(infinite)
|     |     |              |  +--rw no-end-time?       empty
|     |     |              +--:(duration)
|     |     |              |  +--rw duration?          uint32
|     |     |              +--:(end-date-time)
|     |     |                 +--rw end-date-time?
|     |     |                            yang:date-and-time
|     |     +--:(independent-send-accept-lifetime)
|     |     |   {independent-send-accept-lifetime}?
|     |     +--rw send-lifetime
|     |     |  +--rw (lifetime)?
|     |     |     +--:(always)
|     |     |     |  +--rw always?            empty
|     |     |     +--:(start-end-time)
|     |     |        +--rw start-date-time?
|     |     |        |       yang:date-and-time
|     |     |        +--rw (end-time)?
|     |     |           +--:(infinite)
|     |     |           |  +--rw no-end-time?       empty
|     |     |           +--:(duration)
|     |     |           |  +--rw duration?          uint32
|     |     |           +--:(end-date-time)
|     |     |              +--rw end-date-time?
|     |     |                         yang:date-and-time
|     |     |        +--rw accept-lifetime
|     |     |           +--rw (lifetime)?
|     |     |              +--:(always)
|     |     |              |  +--rw always?            empty
|     |     |              +--:(start-end-time)
|     |     |                 +--rw start-date-time?
|     |     |                 |       yang:date-and-time
|     |     |                 +--rw (end-time)?
|     |     |                    +--:(infinite)
|     |     |                    |  +--rw no-end-time?       empty
|     |     |                    +--:(duration)
|     |     |                    |  +--rw duration?          uint32
|     |     |                    +--:(end-date-time)
|     |     |                       +--rw end-date-time?
|     |     |                                  yang:date-and-time
|     |     +--rw crypto-algorithm identityref
|     |     +--rw key-string
|     |     |  +--rw (key-string-style)?
|     |     |     +--:(keystring)
|     |     |     |  +--rw keystring?            string
|     |     |     +--:(hexadecimal) {hex-key-string}?
|     |     |        +--rw hexadecimal-string?   yang:hex-string
```

```
      |     +--ro send-lifetime-active?     boolean
      |     +--ro accept-lifetime-active?   boolean
      +--rw aes-key-wrap {aes-key-wrap}?
         +--rw enable?   boolean
```

4. Key Chain YANG Model

```
<CODE BEGINS> file "ietf-key-chain@2017-04-18.yang"
module ietf-key-chain {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-key-chain";
  prefix key-chain;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }

  organization
    "IETF RTG (Routing) Working Group";
  contact
    "Acee Lindem - acee@cisco.com";
  description
    "This YANG module defines the generic configuration
     data for key-chain. It is intended that the module
     will be extended by vendors to define vendor-specific
     key-chain configuration parameters.

     Copyright (c) 2017 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2017-04-18 {
    description
      "Initial RFC Revision";
    reference "RFC XXXX: A YANG Data Model for key-chain";
  }
```

```
     feature hex-key-string {
       description
         "Support hexadecimal key string.";
     }

     feature accept-tolerance {
       description
         "Support the tolerance or acceptance limit.";
     }

     feature independent-send-accept-lifetime {
       description
         "Support for independent send and accept key lifetimes.";
     }

     feature crypto-hmac-sha-1-12 {
       description
         "Support for TCP HMAC-SHA-1 12 byte digest hack.";
     }

     feature clear-text {
       description
         "Support for clear-text algorithm. Usage is
          NOT RECOMMENDED.";
     }

     feature aes-cmac-prf-128 {
       description
         "Support for AES Cipher based Message Authentication
          Code Pseudo Random Function.";
     }

     feature aes-key-wrap {
       description
         "Support for Advanced Encryption Standard (AES) Key Wrap.";
     }

     feature replay-protection-only {
       description
         "Provide replay-protection without any authentication
          as required by protocols such as Bidirectional
          Forwarding Detection (BFD).";
     }
     identity crypto-algorithm {
       description
         "Base identity of cryptographic algorithm options.";
     }
```

```
     identity hmac-sha-1-12 {
       base crypto-algorithm;
       if-feature "crypto-hmac-sha-1-12";
       description
         "The HMAC-SHA1-12 algorithm.";
     }

     identity aes-cmac-prf-128 {
       base crypto-algorithm;
       if-feature "aes-cmac-prf-128";
       description
         "The AES-CMAC-PRF-128 algorithm - required by
          RFC 5926 for TCP-AO key derivation functions.";
     }

     identity md5 {
       base crypto-algorithm;
       description
         "The MD5 algorithm.";
     }

     identity sha-1 {
       base crypto-algorithm;
       description
         "The SHA-1 algorithm.";
     }

     identity hmac-sha-1 {
       base crypto-algorithm;
       description
         "HMAC-SHA-1 authentication algorithm.";
     }

     identity hmac-sha-256 {
       base crypto-algorithm;
       description
         "HMAC-SHA-256 authentication algorithm.";
     }

     identity hmac-sha-384 {
       base crypto-algorithm;
       description
         "HMAC-SHA-384 authentication algorithm.";
     }

     identity hmac-sha-512 {
       base crypto-algorithm;
       description
```

```
        "HMAC-SHA-512 authentication algorithm.";
    }

    identity clear-text {
      base crypto-algorithm;
      if-feature "clear-text";
      description
        "Clear text.";
    }

    identity replay-protection-only {
      base crypto-algorithm;
      if-feature "replay-protection-only";
      description
        "Provide replay-protection without any authentication as
         required by protocols such as Bidirectional Forwarding
         Detection (BFD).";
    }

    typedef key-chain-ref {
      type leafref {
        path
        "/key-chain:key-chains/key-chain:key-chain/key-chain:name";
      }
      description
        "This type is used by data models that need to reference
         configured key-chains.";
    }

    grouping lifetime {
      description
        "Key lifetime specification.";
      choice lifetime {
        default "always";
        description
          "Options for specifying key accept or send lifetimes";
        case always {
          leaf always {
            type empty;
            description
              "Indicates key lifetime is always valid.";
          }
        }
        case start-end-time {
          leaf start-date-time {
            type yang:date-and-time;
            description
              "Start time.";
```

```
          }
          choice end-time {
            default "infinite";
            description
              "End-time setting.";
            case infinite {
              leaf no-end-time {
                type empty;
                description
                  "Indicates key lifetime end-time is infinite.";
              }
            }
            case duration {
              leaf duration {
                type uint32 {
                  range "1..2147483646";
                }
                units "seconds";
                description
                  "Key lifetime duration, in seconds";
              }
            }
            case end-date-time {
              leaf end-date-time {
                type yang:date-and-time;
                description
                  "End time.";
              }
            }
          }
        }
      }
    }
    grouping key-common {
      description
        "Key-chain key data nodes common to
         configuration and state.";
    }

    container key-chains {
      description
        "All configured key-chains on the device.";
      list key-chain {
        key "name";
        description
          "List of key-chains.";
        leaf name {
          type string;
```

```
          description
            "Name of the key-chain.";
        }
        leaf description {
          type string;
          description
            "A description of the key-chain";
        }
        container accept-tolerance {
          if-feature "accept-tolerance";
          description
            "Tolerance for key lifetime acceptance (seconds).";
          leaf duration {
            type uint32;
            units "seconds";
            default "0";
            description
              "Tolerance range, in seconds.";
          }
        }
        leaf last-modified-timestamp {
          type yang:date-and-time;
          config false;
          description
            "Timestamp of the most recent update to the key-chain";
        }
        list key {
          key "key-id";
          description
            "Single key in key chain.";
          leaf key-id {
            type uint64;
            description
              "Numeric value uniquely identifying the key";
          }
          container lifetime {
            description
              "Specify a key's lifetime.";
            choice lifetime {
              description
                "Options for specification of send and accept
                 lifetimes.";
              case send-and-accept-lifetime {
                description
                  "Send and accept key have the same lifetime.";
                container send-accept-lifetime {
                  description
                    "Single lifetime specification for both
```

```
                    send and accept lifetimes.";
                uses lifetime;
              }
            }
            case independent-send-accept-lifetime {
              if-feature "independent-send-accept-lifetime";
              description
                "Independent send and accept key lifetimes.";
              container send-lifetime {
                description
                  "Separate lifetime specification for send
                   lifetime.";
                uses lifetime;
              }
              container accept-lifetime {
                description
                  "Separate lifetime specification for accept
                   lifetime.";
                uses lifetime;
              }
            }
          }
        }
        leaf crypto-algorithm {
          type identityref {
            base crypto-algorithm;
          }
          mandatory true;
          description
            "Cryptographic algorithm associated with key.";
        }
        container key-string {
          description
            "The key string.";
          nacm:default-deny-all;
          choice key-string-style {
            description
              "Key string styles";
             case keystring {
               leaf keystring {
                type string;
                description
                  "Key string in ASCII format.";
              }
            }
            case hexadecimal {
              if-feature "hex-key-string";
              leaf hexadecimal-string {
```

```
                    type yang:hex-string;
                    description
                      "Key in hexadecimal string format. When compared
                       to ASCII, specification in hexadecimal affords
                       greater key entropy with the same number of
                       internal key-string octets. Additionally, it
                       discourages usage of well-known words or
                       numbers.";
                  }
                }
              }
            }
            leaf send-lifetime-active {
              type boolean;
              config false;
              description
                "Indicates if the send lifetime of the
                 key-chain key is currently active.";
               }
            leaf accept-lifetime-active {
              type boolean;
              config false;
              description
                "Indicates if the accept lifetime of the
                 key-chain key is currently active.";
            }
          }
        }
        container aes-key-wrap {
          if-feature "aes-key-wrap";
          description
            "AES Key Wrap encryption for key-chain key-strings. The
             encrypted key-strings are encoded as hexadecimal key
             strings using the hex-key-string leaf.";
          leaf enable {
            type boolean;
            default "false";
            description
              "Enable AES Key Wrap encryption.";
          }
        }
      }
    }
    <CODE ENDS>
```

5. Security Considerations

The YANG module defined in this document is designed to be accessed via network management protocols such as NETCONF [NETCONF] or RESTCONF [RESTCONF]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [NETCONF-SSH]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [TLS].

The NETCONF access control model [NETCONF-ACM] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content. The key strings are not accessible by default and NETCONF Access Control Mode [NETCONF-ACM] rules are required to configure or retrieve them.

When configured, the key-strings can be encrypted using the AES Key Wrap algorithm [AES-KEY-WRAP]. The AES key-encryption key (KEK) is not included in the YANG model and must be set or derived independent of key-chain configuration. When AES key-encryption is used, the hex-key-string feature is also required since the encrypted keys will contain characters that are not representable in the YANG string built-in type [YANG-1.1]. It is RECOMMENDED that key-strings be encrypted using AES key-encryption to prevent key-chains from being retrieved and stored with the key-strings in clear text. This recommendation is independent of the access protection that is availed from the NETCONF Access Control Model (NACM) [NETCONF-ACM].

The clear-text algorithm is included as a YANG feature. Usage is NOT RECOMMENDED except in cases where the application and device have no other alternative (e.g., a legacy network device that must authenticate packets at intervals of 10 milliseconds or less for many peers using Bidirectional Forwarding Detection [BFD]). Keys used with the clear-text algorithm are considered insecure and SHOULD NOT be reused with more secure algorithms.

Similarly, the MD5 and SHA-1 algorithms have been proven to be insecure ([Dobb96a], [Dobb96b], and [SHA-SEC-CON]) and usage is NOT RECOMMENDED. Usage should be confined to deployments where it is required for backward compatibility.

Implementations with keys provided via this model should store them using best current security practices.

6. IANA Considerations

This document registers a URI in the IETF XML registry [XML-REGISTRY]. Following the format in [XML-REGISTRY], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-key-chain
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 [YANG-1.0].

name: ietf-key-chain
namespace: urn:ietf:params:xml:ns:yang:ietf-key-chain
prefix: key-chain
reference: RFC XXXX

7. Contributors

Contributors' Addresses

Yi Yang
SockRate

Email: yi.yang@sockrate.com

8. References

8.1. Normative References

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

[NETCONF-ACM] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

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

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

[YANG-1.0]
Bjorklund, M., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[YANG-1.1]
Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

8.2. Informative References

[AES-KEY-WRAP]
Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 5649, August 2009.

[BFD] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[CRYPTO-KEYTABLE]
Housley, R., Polk, T., Hartman, S., and D. Zhang, "Table of Cryptographic Keys", RFC 7210, April 2014.

[Dobb96a] Dobbertin, H., "Cryptanalysis of MD5 Compress", Technical Report (Presented at the RUMP Session of EuroCrypt 1996), 2 May 1996.

[Dobb96b] Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes Vol. 2, No. 2, Summer 1996.

[IAB-REPORT]
Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, August 2007.

[NETCONF-SSH]
Wasserman, M., "NETCONF over SSH", RFC 6242, June 2011.

[NMDA] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watson, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-01.txt (work in progress), March 2017.

[NTP-PROTO]
Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[OSPFV3-AUTH]
Bhatia, M., Manral, V., and A. Lindem, "Supporting Authentication Trailer for OSPFv3", RFC 7166, March 2014.

[RESTCONF]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[SHA-SEC-CON]
Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, February 2011.

[TCP-AO] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[TCP-AO-ALGORITHMS]
Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, June 2010.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol", RFC 5246, August 2008.

[YANG-CRYPTO-KEYTABLE]
Chen, I., "YANG Data Model for RFC 7210 Key Table", draft-chen-rtg-key-table-yang-00.txt (work in progress), November 2015.

Appendix A. Examples

A.1. Simple Key Chain with Always Valid Single Key

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>keychain-no-end-time</name>
      <description>
        A key chain with a single key that is always valid for
        transmission and reception.
      </description>
      <key>
        <key-id>100</key-id>
        <lifetime>
          <send-accept-lifetime>
            <always/>
          </send-accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <keystring>keystring_in_ascii_100</keystring>
        </key-string>
      </key>
    </key-chain>
  </key-chains>
</data>
```

A.2. Key Chain with Keys having Different Lifetimes

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>keychain2</name>
      <description>
        A key chain where each key contains different send time
        and accept time and a different algorithm illustrating
        algorithm agility.
      </description>
      <key>
        <key-id>35</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-01-01T00:00:00Z</start-date-time>
            <end-date-time>2017-02-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2016-12-31T23:59:55Z</start-date-time>
            <end-date-time>2017-02-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <keystring>keystring_in_ascii_35</keystring>
        </key-string>
      </key>
      <key>
        <key-id>36</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-02-01T00:00:00Z</start-date-time>
            <end-date-time>2017-03-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2017-01-31T23:59:55Z</start-date-time>
            <end-date-time>2017-03-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-512</crypto-algorithm>
        <key-string>
          <hexadecimal-string>fe:ed:be:af:36</hexadecimal-string>
        </key-string>
      </key>
    </key-chain>
  </key-chains>
</data>
```

A.3. Key Chain with Independent Send and Accept Lifetimes

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>keychain2</name>
      <description>
        A key chain where each key contains different send time
        and accept times.
      </description>
      <key>
        <key-id>35</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-01-01T00:00:00Z</start-date-time>
            <end-date-time>2017-02-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2016-12-31T23:59:55Z</start-date-time>
            <end-date-time>2017-02-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <keystring>keystring_in_ascii_35</keystring>
        </key-string>
      </key>
      <key>
        <key-id>36</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-02-01T00:00:00Z</start-date-time>
            <end-date-time>2017-03-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2017-01-31T23:59:55Z</start-date-time>
            <end-date-time>2017-03-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <hexadecimal-string>fe:ed:be:af:36</hexadecimal-string>
        </key-string>
      </key>
    </key-chain>
  </key-chains>
</data>
```

Appendix B. Acknowledgments

The RFC text was produced using Marshall Rose's xml2rfc tool.

Thanks to Brian Weis for fruitful discussions on security requirements.

Thanks to Ines Robles for Routing Directorate QA review comments.

Thanks to Ladislav Lhotka for YANG Doctor comments.

Thanks to Martin Bjorklund for additional YANG Doctor comments.

Thanks to Tom Petch for comments during IETF last call.

Thanks to Matthew Miller for comments made during the Gen-ART review.

Thanks to Vincent Roca for comments made during the Security Directorate review.

Thanks to Warren Kumari, Ben Campbell, Adam Roach, and Benoit Claise for comments received during the IESG review.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Yingzhen Qu
Huawei

Email: yingzhen.qu@huawei.com

Derek Yeung
Arrcus, Inc

Email: derek@arrcus.com

Ing-Wher Chen
Jabil

Email: ing-wher_chen@jabil.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

A. Kumar
J. Kolhe
S. Ghemawat
L. Ryan
Google

July 8, 2016

gRPC Protocol
draft-kumar-rtgwg-grpc-protocol-00

Abstract

This document presents gRPC protocol specification.

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.

Table of Contents

1. Introduction

This document serves as a detailed description for an implementation of gRPC carried over HTTP2 draft 17 framing. It assumes familiarity with the HTTP2 specification.

2. Outline

The following is the general sequence of message atoms in a GRPC request and response message stream.

- Request -> Request-Headers *Length-Prefixed-Message EOS
- Response -> (Response-Headers *Length-Prefixed-Message Trailers) / Trailers-Only

3. Protocol Requests

At a high level, the protocol has the following request and response fields.

- Request-Headers -> Call-Definition *Custom-Metadata
- Call-Definition -> Method Scheme Path TE [Authority] [Timeout] Content-Type [Message-Type] [Message-Encoding] [Message-Accept-Encoding] [User-Agent]
- Method -> ":method POST"
- Scheme -> ":scheme " ("http" / "https")
- Path -> ":path" {path identifying method within exposed API}
- Authority -> ":authority" {virtual host name of authority}
- TE -> "te" "trailers" # Used to detect incompatible proxies

- Timeout -> "grpc-timeout" TimeoutValue TimeoutUnit
- TimeoutValue -> {positive integer as ASCII string of at most 8 digits}
- TimeoutUnit -> Hour / Minute / Second / Millisecond / Microsecond / Nanosecond
- Hour -> "H"
- Minute -> "M"
- Second -> "S"
- Millisecond -> "m"
- Microsecond -> "u"
- Nanosecond -> "n"
- Content-Type -> "content-type" "application/grpc" [("+proto" / "+json" / {custom})]
- Content-Coding -> "identity" / "gzip" / "deflate" / "snappy" / {custom}
- Message-Encoding -> "grpc-encoding" Content-Coding
- Message-Accept-Encoding -> "grpc-accept-encoding" Content-Coding *("," Content-Coding)
- User-Agent -> "user-agent" {structured user-agent string}
- Message-Type -> "grpc-message-type" {type name for message schema}
- Custom-Metadata -> Binary-Header / ASCII-Header
- Binary-Header -> {Header-Name "-bin" } {base64 encoded value}
- ASCII-Header -> Header-Name ASCII-Value
- Header-Name -> 1*(%x30-39 / %x61-7A / "_" / "-" / ".") ; 0-9 a-z _ - .
- ASCII-Value -> 1*(%x20-%x7E) ; space and printable ASCII

HTTP2 requires that reserved headers, ones starting with ":" appear before all other headers. Additionally implementations should send

Timeout immediately after the reserved headers and they should send the Call-Definition headers before sending Custom-Metadata.

If Timeout is omitted a server should assume an infinite timeout. Client implementations are free to send a default minimum timeout based on their deployment requirements.

Custom-Metadata is an arbitrary set of key-value pairs defined by the application layer. Header names starting with "grpc-" but not listed here are reserved for future GRPC use and should not be used by applications as Custom-Metadata.

Note that HTTP2 does not allow arbitrary octet sequences for header values so binary header values must be encoded using Base64 as per https://tools.ietf.org/html/rfc4648#section-4. Implementations MUST accept padded and un-padded values and should emit un-padded values. Applications define binary headers by having their names end with "-bin". Runtime libraries use this suffix to detect binary headers and properly apply base64 encoding and decoding as headers are sent and received.

Custom-Metadata header order is not guaranteed to be preserved except for values with duplicate header names. Duplicate header names may have their values joined with "," as the delimiter and be considered semantically equivalent. Implementations must split Binary-Headers on "," before decoding the Base64-encoded values.

ASCII-Value should not have leading or trailing whitespace. If it contains leading or trailing whitespace, it may be stripped. The ASCII-Value character range defined is more strict than HTTP. Implementations must not error due to receiving an invalid ASCII-Value that’s a valid field-value in HTTP, but the precise behavior is not strictly defined: they may throw the value away or accept the value. If accepted, care must be taken to make sure that the application is permitted to echo the value back as metadata. For example, if the metadata is provided to the application as a list in a request, the application should not trigger an error by providing that same list as the metadata in the response. Servers may limit the size of Request-Headers, with a default of 8 KiB suggested. Implementations are encouraged to compute total header size like HTTP/2’s SETTINGS_MAX_HEADER_LIST_SIZE: the sum of all header fields, for each field the sum of the uncompressed field name and value lengths plus 32, with binary values’ lengths being post-Base64.

Servers may limit the size of Request-Headers, with a default of 8 KiB suggested. Implementations are encouraged to compute total header size like HTTP/2’s SETTINGS_MAX_HEADER_LIST_SIZE: the sum of all header fields, for each field the sum of the uncompressed field

name and value lengths plus 32, with binary values' lengths being post-Base64.

The repeated sequence of Length-Prefixed-Message items is delivered in DATA frames.

- Length-Prefixed-Message -> Compressed-Flag Message-Length Message
- Compressed-Flag -> 0 / 1 # encoded as 1 byte unsigned integer
- Message-Length -> {length of Message} # encoded as 4 byte unsigned integer
- Message \u002D-> *{binary octet}

A Compressed-Flag value of 1 indicates that the binary octet sequence of Message is compressed using the mechanism declared by the Message-Encoding header. A value of 0 indicates that no encoding of Message bytes has occurred. Compression contexts are NOT maintained over message boundaries, implementations must create a new context for each message in the stream. If the Message-Encoding header is omitted then the Compressed-Flag must be 0.

For requests, EOS (end-of-stream) is indicated by the presence of the END_STREAM flag on the last received DATA frame. In scenarios where the Request stream needs to be closed but no data remains to be sent implementations MUST send an empty DATA frame with this flag set.

4. Responses

- Response -> (Response-Headers *Length-Prefixed-Message Trailers) / Trailers-Only
- Response-Headers -> HTTP-Status [Message-Encoding] [Message-Accept-Encoding] Content-Type *Custom-Metadata
- Trailers-Only -> HTTP-Status Content-Type Trailers
- Trailers -> Status [Status-Message] *Custom-Metadata
- HTTP-Status -> ":status 200"
- Status -> "grpc-status"
- Status-Message -> "grpc-message"

Response-Headers and Trailers-Only are each delivered in a single HTTP2 HEADERS frame block. Most responses are expected to have both

headers and trailers but Trailers-Only is permitted for calls that produce an immediate error. Status must be sent in Trailers even if the status code is OK.

For responses end-of-stream is indicated by the presence of the END_STREAM flag on the last received HEADERS frame that carries Trailers.

Implementations should expect broken deployments to send non-200 HTTP status codes in responses as well as a variety of non-GRPC content-types and to omit Status and Status-Message. Implementations must synthesize a Status and Status-Message to propagate to the application layer when this occurs.

Clients may limit the size of Response-Headers, Trailers, and Trailers-Only, with a default of 8 KiB each suggested.

5. Examples

Sample unary-call showing HTTP2 framing sequence

Request

```
HEADERS (flags = END_HEADERS)
:method = POST
:scheme = http
:path = /google.pubsub.v2.PublisherService/CreateTopic
:authority = pubsub.googleapis.com
grpc-timeout = 1S
content-type = application/grpc+proto
grpc-encoding = gzip
authorization = Bearer y235.wef315yfh138vh31hv93hv8h3v

DATA (flags = END_STREAM)
<Length-Prefixed Message>
```

Response

```
HEADERS (flags = END_HEADERS)
:status = 200
grpc-encoding = gzip

DATA
<Length-Prefixed Message>

HEADERS (flags = END_STREAM, END_HEADERS)
grpc-status = 0 # OK
trace-proto-bin = jher831yy13JHy3hc
```

6. User Agents

While the protocol does not require a user-agent to function it is recommended that clients provide a structured user-agent string that provides a basic description of the calling library, version and platform to facilitate issue diagnosis in heterogeneous environments. The following structure is recommended to library developers

User-Agent \u002D-> "grpc-" Language ?("-" Variant) "/" Version ?(" (" *(AdditionalProperty ";") ")")

7. HTTP2 Transport Mapping

All GRPC calls need to specify an internal ID. We will use HTTP2 stream-ids as call identifiers in this scheme. NOTE: These id's are contextual to an open HTTP2 session and will not be unique within a given process that is handling more than one HTTP2 session nor can they be used as GUIDs.

DATA frame boundaries have no relation to Length-Prefixed-Message boundaries and implementations should make no assumptions about their alignment.

When an application or runtime error occurs during an RPC a Status and Status-Message are delivered in Trailers. In some cases it is possible that the framing of the message stream has become corrupt and the RPC runtime will choose to use an RST_STREAM frame to indicate this state to its peer. RPC runtime implementations should interpret RST_STREAM as immediate full-closure of the stream and should propagate an error up to the calling application layer.

HTTP2 Code#	GRPC Code
NO_ERROR(0)	INTERNAL - An explicit GRPC status of OK should have been sent but this might be used to aggressively lameduck in some scenarios.
PROTOCOL_ERROR(1)	INTERNAL
INTERNAL_ERROR(2)	INTERNAL
FLOW_CONTROL_ERROR(3)	INTERNAL
SETTINGS_TIMEOUT(4)	INTERNAL
STREAM_CLOSED	FRAME_SIZE_ERROR
STREAM_CLOSED	INTERNAL
REFUSED_STREAM	UNAVAILABLE - Indicates that no processing occurred and the request can be retried, possibly elsewhere.
CANCEL(8)	Mapped to call cancellation when sent by a client.Mapped to CANCELLED when sent by a server. Note that servers should only use this mechanism when they need to cancel a call but the payload byte sequence is incomplete.
COMPRESSION_ERROR	INTERNAL
CONNECT_ERROR	INTERNAL
ENHANCE_YOUR_CALM	RESOURCE_EXHAUSTED ...with additional error detail provided by runtime to indicate that the exhausted resource is bandwidth.
INADEQUATE_SECURITY	PERMISSION_DENIED ... with additional detail indicating that permission was denied as protocol is not secure enough for call.

Table 1: Error Code Mapping

The HTTP2 specification mandates the use of TLS 1.2 or higher when TLS is used with HTTP2. It also places some additional constraints on the allowed ciphers in deployments to avoid known-problems as well as requiring SNI support. It is also expected that HTTP2 will be used in conjunction with proprietary transport security mechanisms about which the specification can make no meaningful recommendations.

GOAWAY Frame Sent by servers to clients to indicate that they will no longer accept any new streams on the associated connections. This frame includes the id of the last successfully accepted stream by the server. Clients should consider any stream initiated after the last successfully accepted stream as UNAVAILABLE and retry the call elsewhere. Clients are free to continue working with the already accepted streams until they complete or the connection is terminated. Servers should send GOAWAY before terminating a connection to reliably inform clients which work has been accepted by the server and is being executed.

PING Frame Both clients and servers can send a PING frame that the peer must respond to by precisely echoing what they received. This is used to assert that the connection is still live as well as providing a means to estimate end-to-end latency. If a server initiated PING does not receive a response within the deadline expected by the runtime all outstanding calls on the server will be closed with a CANCELLED status. An expired client initiated PING will cause all calls to be closed with an UNAVAILABLE status. Note that the frequency of PINGs is highly dependent on the network environment, implementations are free to adjust PING frequency based on network and application requirements.

Connection failure If a detectable connection failure occurs on the client all calls will be closed with an UNAVAILABLE status. For servers open calls will be closed with a CANCELLED status.

8. Normative references

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

Authors' Addresses

Abhishek Kumar
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: abhikumar@google.com

Jayant Kolhe
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jkolhe@google.com

Sanjay Ghemawat
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: sanjay@google.com

Louis Ryan
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: lryan@google.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 27, 2017

A. Lindem, Ed.
Cisco Systems
L. Berger, Ed.
LabN Consulting, L.L.C.
D. Bogdanovic

C. Hopps
Deutsche Telekom
July 26, 2016

Network Device YANG Organizational Models
draft-rtgyangdt-rtgwg-device-model-05

Abstract

This document presents an approach for organizing YANG models in a comprehensive structure that may be used to configure and operate network devices. The structure is itself represented as a YANG model, with all of the related component models logically organized in a way that is operationally intuitive, but this model is not expected to be implemented. The identified component modules are expected to be defined and implemented on common network devices.

This document is derived from work submitted to the IETF by members of the informal OpenConfig working group of network operators and is a product of the Routing Area YANG Architecture design team.

Table of Contents

1. Introduction

"Operational Structure and Organization of YANG Models" [I-D.openconfig-netmod-model-structure], highlights the value of organizing individual, self-standing YANG [RFC6020] models into a more comprehensive structure. This document builds on that work and presents a derivative structure for use in representing the networking infrastructure aspects of physical and virtual devices. [I-D.openconfig-netmod-model-structure] and earlier versions of this document presented a single device-centric model root, this document no longer contains this element. Such an element would have translated to a single device management model that would be the root

of all other models and was judged to be overly restrictive in terms of definition, implementation, and operation.

The document presents a notional network device YANG organizational structure that provides a conceptual framework for the models that may be used to configure and operate network devices. The structure is itself presented as a YANG module, with all of the related component modules logically organized in a way that is operationally intuitive. This network device model is not expected to be implemented, but rather provide as context for the identified representative component modules with are expected to be defined, and supported on typical network devices.

This document refers to two new modules that are expected to be implemented. These models are defined to support the configuration and operation of network-devices that allow for the partitioning of resources from both, or either, management and networking perspectives. Two forms of resource partitioning are referenced:

The first form provides a logical partitioning of a network device where each partition is separately managed as essentially an independent network element which is 'hosted' by the base network device. These hosted network elements are referred to as logical network elements, or LNEs, and are supported by the logical-network-element module defined in [LNE-MODEL]. The module is used to identify LNEs and associate resources from the network-device with each LNE. LNEs themselves are represented in YANG as independent network devices; each accessed independently. Optionally, and when supported by the implementation, they may also be accessed from the host system. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

The second form provides support what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026]. In this form of resource partitioning multiple control plane and forwarding/bridging instances are provided by and managed via a single (physical or logical) network device. This form of resource partitioning is referred to as Network Instances and are supported by the network-instance module defined in [NI-MODEL]. Configuration and operation of each network-instance is always via the network device and the network-instance module.

This document was motivated by, and derived from, [I-D.openconfig-netmod-model-structure]. The requirements from that document have been combined with the requirements from "Consistent Modeling of Operational State Data in YANG",

[I-D.openconfig-netmod-opstate], into "NETMOD Operational State Requirements", [I-D.ietf-netmod-opstate-reqs]. This document is aimed at the requirement related to a common model-structure, currently Requirement 7, and also aims to provide a modeling base for Operational State representation.

The approach taken in this (and the original) document is to organize the models describing various aspects of network infrastructure, focusing on devices, their subsystems, and relevant protocols operating at the link and network layers. The proposal does not consider a common model for higher level network services. We focus on the set of models that are commonly used by network operators, and suggest a corresponding organization.

A significant portion of the text and model contained in this document was taken from the -00 of [I-D.openconfig-netmod-model-structure].

1.1. Status of Work and Open Issues

This version of the document and structure are a product of the Routing Area YANG Architecture design team and is very much a work in progress rather than a final proposal. This version is a major change from the prior version and this change was enabled by the work on the previously mentioned Schema Mount.

Schema Mount enables a dramatic simplification of the presented device model, particularly for "lower-end" devices which are unlikely to support multiple network instances or logical network elements. Should structural-mount/YSDL not be available, the more explicit tree structure presented in earlier versions of this document will need to be utilized.

The top open issues are:

1. This document will need to match the evolution and standardization of [I-D.openconfig-netmod-opstate] or [I-D.ietf-netmod-opstate-reqs] by the Netmod WG.

2. Interpretation of different policy containers requires clarification.

3. It may make sense to use the identityref structuring with hardware and QoS model.

4. Which document(s) define the base System management, network services, and oam protocols modules is TBD. This includes the

possibility of simply using RFC7317 in place of the presented System management module.

5. The model will be updated once the "opstate" requirements are addressed.

2. Module Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

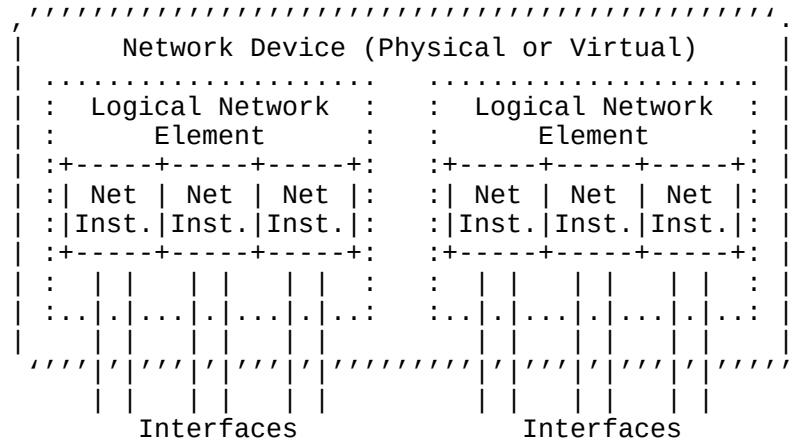


Figure 1: Module Element Relationships

A model for LNEs is described in [LNE-MODEL] and the model for network instances is covered in [NI-MODEL].

The presented notional network device module can itself be thought of as a "meta-model" as it describes the relationships between individual models. We choose to represent it also as a simple YANG module consisting of other models, which are in fact independent top

level individual models. Although it is never expected to be implemented.

The presented modules do not follow the hierarchy of any Particular implementation, and hence is vendor-neutral. Nevertheless, the structure should be familiar to network operators and also readily mapped to vendor implementations.

The overall structure is:

```
module: ietf-network-device
   +--rw modules-state             [I-D.ietf-netconf-yang-library]
   |
   +--rw interfaces                [RFC7223]
   +--rw hardware
   +--rw qos
   |
   +--rw system-management         [RFC7317 or derived]
   +--rw network-services
   +--rw oam-protocols
   |
   +--rw routing                   [I-D.ietf-netmod-routing-cfg]
   +--rw mpls
   +--rw ieee-dot1Q
   |
   +--rw acls                      [I-D.ietf-netmod-acl-model]
   +--rw key-chains                [I-D.ietf-rtgwg-yang-key-chain]
   |
   +--rw logical-network-elements [I-D.rtgyangdt-rtgwg-lne-model]
   +--rw network-instances         [I-D.rtgyangdt-rtgwg-ni-model]
```

The network device is composed of top level modules that can be used to configure and operate a network device. (This is a significant difference from earlier versions of this document where there was a strict model hierarchy.) Importantly the network device structure is the same for a physical network device or a logical network device, such as those instantiated via the logical-network-element model. Extra spacing is included to denote different types of modules included.

YANG library [I-D.ietf-netconf-yang-library] is included as it used to identify details of the top level modules supported by the (physical or logical) network device. Th ability to identify supported modules is particularly important for LNEs which may have a set of supported modules which differs from the set supported by the host network device.

The interface management model [RFC7223] is included at the top level. The hardware module is a placeholder for a future device-specific configuration and operational state data model. For example, a common structure for the hardware model might include chassis, line cards, and ports, but we leave this unspecified. The quality of service (QoS) section is also a placeholder module for device configuration and operational state data which relates to the treatment of traffic across the device. This document references augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

System management, network services, and oam protocols represent new top level modules that are used to organize data models of similar functions. Additional information on each is provided below.

The routing and MPLS modules provide core support for the configuration and operation of a devices control plane and data plane functions. IEEE dot1Q [IEEE-8021Q] is an example of another module that provides similar functions for VLAN bridging, and other similar modules are also possible. Each of these modules is expected to be LNE and NI unaware, and to be instantiated as needed as part of the LNE and NI configuration and operation supported by the logical-network-element and network-instance modules. (Note that this is a change from [I-D.ietf-netmod-routing-cfg] which is currently defined with VRF/NI semantics.)

The access control list (ACL) and key chain modules are included as examples of other top level modules that may be supported by a network device.

The logical network element and network instance modules enable LNEs and NIs respectively and are defined below.

2.1. Interface Model Components

Interfaces are a crucial part of any network device’s configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [RFC7223].

The logical-network-element and network-instance modules defined in [LNE-MODEL] and [NI-MODEL] augment the existing interface management model in two ways: The first, by the logical-network-element module, adds an identifier which is used on physical interface types to identify an associated LNE. The second, by the network-instance module, adds a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [RFC7277].

The interface related augmentations are as follows:

```
module: ietf-logical-network-element
augment /if:interfaces/if:interface:
   +--rw bind-lne-name?   string

module: ietf-network-instance
augment /if:interfaces/if:interface:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv4:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv6:
   +--rw bind-network-instance-name?   string
```

The following is an example of envisioned combined usage. The interfaces container includes a number of commonly used components as examples:

```
            +--rw if:interfaces
            |  +--rw interface* [name]
            |     +--rw name                        string
            |     +--rw lne:bind-lne-name?          string
            |     +--rw ethernet
            |     |  +--rw ni:bind-network-instance-name? string
            |     |  +--rw aggregates
            |     |  +--rw rstp
            |     |  +--rw lldp
            |     |  +--rw ptp
            |     +--rw vlans
            |     +--rw tunnels
            |     +--rw ipv4
            |     |  +--rw ni:bind-network-instance-name? string
            |     |  +--rw arp
            |     |  +--rw icmp
            |     |  +--rw vrrp
            |     |  +--rw dhcp-client
            |     +--rw ipv6
            |        +--rw ni:bind-network-instance-name? string
            |        +--rw vrrp
            |        +--rw icmpv6
            |        +--rw nd
            |        +--rw dhcpv6-client
```

The [RFC7223] defined interface model is structured to include all interfaces in a flat list, without regard to logical or virtual instances (e.g., VRFs) supported on the device. The bind-lne-name and bind-network-instance-name leaves provide the association between an interface and its associated LNE and NI (e.g., VRF or VSI).

2.2. System Management

[Editor’s note: need to discuss and resolve relationship between this structure and RFC7317 and determine if 7317 is close enough to simply use as is.]

System management is expected to reuse definitions contained in [RFC7317]. It is expected to be instantiated per device and LNE. Its structure is shown below:

```
    module: ietf-network-device
       +--rw system-management
       |  +--rw system-management-global
       |  +--rw system-management-protocol* [type]
       |     +--rw type    identityref
```

System-management-global is used for configuration information and state that is independent of a particular management protocol. System-management-protocol is a list of management protocol specific elements. The type-specific sub-modules are expected to be defined.

The following is an example of envisioned usage:

```
module: ietf-network-device
   +--rw system-management
      +--rw system-management-global
      |  +--rw statistics-collection
      |  ...
      +--rw system-management-protocol* [type]
      |  +--rw type=syslog
      |  +--rw type=dns
      |  +--rw type=ntp
      |  +--rw type=ssh
      |  +--rw type=tacacs
      |  +--rw type=snmp
      |  +--rw type=netconf
```

2.3. Network Services

A device may provide different network services to other devices, for example a device my act as a DHCP server. The model may be instantiated per device, LNE, and NI. An identityref is used to identify the type of specific service being provided and its associated configuration and state information. The defined structure is as follows:

```
module: ietf-network-device
   +--rw network-services
   |  +--rw network-service* [type]
   |     +--rw type    identityref
```

The following is an example of envisioned usage: Examples shown below include a device-based Network Time Protocol (NTP) server, a Domain Name System (DNS) server, and a Dynamic Host Configuration Protocol (DHCP) server:

```
module: ietf-network-device
   +--rw network-services
      +--rw network-service* [type]
         +--rw type=ntp-server
         +--rw type=dns-server
         +--rw type=dhcp-server
```

2.4. OAM Protocols

OAM protocols that may run within the context of a device are grouped within the oam-protocols model. The model may be instantiated per device, LNE, and NI. An identifyref is used to identify the information and state that may relate to a specific OAM protocol. The defined structure is as follows:

```
module: ietf-network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type    identityref
```

The following is an example of envisioned usage. Examples shown below include Bi-directional Forwarding Detection (BFD), Ethernet Connectivity Fault Management (CFM), and Two-Way Active Measurement Protocol (TWAMP):

```
module: ietf-network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type=bfd
         +--rw type=cfm
         +--rw type=twamp
```

2.5. Routing

Routing protocol and IP forwarding configuration and operation information is modeled via a routing model, such as the one defined in [I-D.ietf-netmod-routing-cfg].

The routing module is expected to include all IETF defined control plane protocols, such as BGP, OSPF, LDP and RSVP-TE. It is also expected to support configuration and operation of or more routing information bases (RIB). A RIB is a list of routes complemented with administrative data. Finally, policy is expected to be represented within each control plane protocol and RIB.

The anticipated structure is as follows:

```
 module: ietf-network-device
     +--rw rt:routing                [I-D.ietf-netmod-routing-cfg]
        +--rw control-plane-protocol* [type]
        |  +--rw type      identityref
        |  +--rw policy
        +--rw rib* [name]
           +--rw name           string
           +--rw description?   string
           +--rw policy
```

2.6. MPLS

MPLS data plane related information is grouped together, as with the previously discussed modules, is unaware of VRFs/NIs. The model may be instantiated per device, LNE, and NI. MPLS control plane protocols are expected to be included in Section 2.5. MPLS may reuse and build on [I-D.openconfig-mpls-consolidated-model] or other emerging models and has an anticipated structure as follows:

```
  module: ietf-network-device
       +--rw mpls
          +--rw global
          +--rw lsps* [type]
             +--rw type    identityref
```

Type refers to LSP type, such as static, traffic engineered or routing congruent. The following is an example of such usage:

```
  module: ietf-network-device
       +--rw mpls
          +--rw global
          +--rw lsps* [type]
              +--rw type=static
              +--rw type=constrained-paths
              +--rw type=igp-congruent
```

3. Security Considerations

The network-device model structure described in this document does not define actual configuration and state data, hence it is not directly responsible for security risks.

Each of the component models that provide the corresponding configuration and state data should be considered sensitive from a security standpoint since they generally manipulate aspects of network configurations. Each component model should be carefully evaluated to determine its security risks, along with mitigations to reduce such risks.

LNE portion is TBD

NI portion is TBD

4. IANA Considerations

This YANG model currently uses a temporary ad-hoc namespace. If it is placed or redirected for the standards track, an appropriate namespace URI will be registered in the "IETF XML Registry" [RFC3688]. The YANG structure modules will be registered in the "YANG Module Names" registry [RFC6020].

5. Network Device Model Structure

```
<CODE BEGINS> file "ietf-network-device@2016-05-01.yang"
module ietf-network-device {

  yang-version "1";

  // namespace
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-device";

  prefix "nd";

  // import some basic types

  // meta
  organization "IETF RTG YANG Design Team Collaboration
                with OpenConfig";

  contact
      "Routing Area YANG Architecture Design Team -
       <rtg-dt-yang-arch@ietf.org>";

  description
    "This module describes a model structure for YANG
     configuration and operational state data models. Its intent is
     to describe how individual device protocol and feature models
     fit together and interact.";

  revision "2016-05-01" {
    description
      "IETF Routing YANG Design Team Meta-Model";
    reference "TBD";
  }

  // extension statements
```

```
     // identity statements

     identity oam-protocol-type {
         description
             "Base identity for derivation of OAM protocols";
     }

     identity network-service-type {
         description
             "Base identity for derivation of network services";
     }

      identity system-management-protocol-type {
         description
             "Base identity for derivation of system management
              protocols";
      }

      identity oam-service-type {
         description
             "Base identity for derivation of Operations,
              Administration, and Maintenance (OAM) services.";
      }

      identity control-plane-protocol-type {
         description
             "Base identity for derivation of control-plane protocols";
      }

      identity mpls-lsp-type {
         description
             "Base identity for derivation of MPLS LSP typs";
      }

     // typedef statements

     // grouping statements

     grouping ribs {
       description
         "Routing Information Bases (RIBs) supported by a
          network-instance";
       container ribs {
           description
               "RIBs supported by a network-instance";
           list rib {
               key "name";
               min-elements "1";
```

```
            description
                "Each entry represents a RIB identified by the
               'name' key. All routes in a RIB must belong to the
                same address family.

                For each routing instance, an implementation should
                provide one system-controlled default RIB for each
                supported address family.";
            leaf name {
                type string;
                description
                    "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                type string;
                description
                    "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                description "Policy specific to RIB";
            }
        }
      }
    }

    // top level device definition statements
    container ietf-yang-library {
      description
        "YANG Module Library as defined in
         draft-ietf-netconf-yang-library";
    }

    container interfaces {
      description
       "Interface list as defined by RFC7223/RFC7224";
    }

    container hardware {
      description
        "Hardware / vendor-specific data relevant to the platform.
        This container is an anchor point for platform-specific
        configuration and operational state data.  It may be further
        organized into chassis, line cards, ports, etc.  It is
        expected that vendor or platform-specific augmentations
        would be used to populate this part of the device model";
    }
```

```
container qos {
  description "QoS features, for example policing, shaping, etc.";
}

container system-management {
    description
      "System management for physical or virtual device.";
    container system-management-global {
        description "System management - with reuse of RFC 7317";
    }
    list system-management-protocol {
        key "type";
        leaf type {
            type identityref {
                base system-management-protocol-type;
            }
            mandatory true;
            description
                "Syslog, ssh, TACAC+, SNMP, NETCONF, etc.";
        }
        description "List of system management protocol
                     configured for a logical network
                     element.";
    }
}

container network-services {
    description
        "Container for list of configured network
         services.";
    list network-service {
        key "type";
        description
            "List of network services configured for a
             network instance.";
        leaf type {
            type identityref {
                base network-service-type;
            }
            mandatory true;
            description
                "The network service type supported within
                 a network instance, e.g., NTP server, DNS
                 server, DHCP server, etc.";
        }
    }
}
```

```
     container oam-protocols {
         description
             "Container for configured OAM protocols.";
         list oam-protocol {
             key "type";
             leaf type {
                 type identityref {
                     base oam-protocol-type;
                 }
                 mandatory true;
                 description
                     "The Operations, Administration, and
                      Maintenance (OAM) protocol type, e.g., BFD,
                      TWAMP, CFM, etc.";
             }
             description
                 "List of configured OAM protocols.";
         }
     }

     container routing {
       description
         "The YANG Data Model for Routing Management revised to be
          Network Instance / VRF independent. ";
       // Note that there is no routing or network instance
       list control-plane-protocol {
           key "type";
           description
               "List of control plane protocols configured for
                a network instance.";
           leaf type {
               type identityref {
                   base control-plane-protocol-type;
               }
               description
                   "The control plane protocol type, e.g., BGP,
                    OSPF IS-IS, etc";
           }
           container policy {
               description
                   "Protocol specific policy,
                   reusing [RTG-POLICY]";
           }
       }
       list rib {
           key "name";
           min-elements "1";
           description
```

```
                "Each entry represents a RIB identified by the
               'name' key. All routes in a RIB must belong to the
                same address family.

                For each routing instance, an implementation should
                provide one system-controlled default RIB for each
                supported address family.";
            leaf name {
                type string;
                description
                    "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                type string;
                description
                    "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                description "Policy specific to RIB";
            }
          }
        }

        container mpls {
            description "MPLS and TE configuration";
            container global {
                description "Global MPLS configuration";
            }
            list lsps {
                key "type";
                description
                    "List of LSP types.";
                leaf type {
                    type identityref {
                        base mpls-lsp-type;
                    }
                    mandatory true;
                    description
                        "MPLS and Traffic Engineering protocol LSP types,
                         static, LDP/SR (igp-congruent),
                         RSVP TE (constrained-paths) , etc.";
                }
            }
        }

        container ieee-dot1Q {
```

```
      description
        "The YANG Data Model for VLAN bridges as defined by the IEEE";
    }

    container ietf-acl {
      description "Packet Access Control Lists (ACLs) as specified
                     in draft-ietf-netmod-acl-model";
    }

    container ietf-key-chain {
      description "Key chains as specified in
                   draft-ietf-rtgwg-yang-key-chain;";
    }

    container logical-network-element {
      description
        "This module is used to support multiple logical network
         elements on a single physical or virtual system.";
    }

    container network-instance {
      description
        "This module is used to support multiple network instances
         within a single physical or virtual device.  Network
         instances are commonly know as VRFs (virtual routing
         and forwarding) and VSIs (virtual switching instances).";
    }
    // rpc statements

    // notification statements

  }
  <CODE ENDS>
```

6. References

6.1. Normative References

[LNE-MODEL]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Logical Network Element Model", draft-rtgyangdt-rtgwg-lne-model-00.txt (work in progress), May 2016.

[NI-MODEL]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Network Instance Model", draft-rtgyangdt-rtgwg-ni-model-00.txt (work in progress), May 2016.

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

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <http://www.rfc-editor.org/info/rfc4026>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

6.2. Informative References

[I-D.ietf-netconf-yang-library]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", draft-ietf-netconf-yang-library-05 (work in progress), April 2016.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-03 (work in progress), January 2016.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-20 (work in progress), October 2015.

[I-D.openconfig-mpls-consolidated-model]
George, J., Fang, L., eric.osborne@level3.com, e., and R. Shakir, "MPLS / TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model-02 (work in progress), October 2015.

[I-D.openconfig-netmod-model-structure]
Shaikh, A., Shakir, R., D'Souza, K., and L. Fang, "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure-00 (work in progress), March 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[IEEE-8021Q]
Holness, M., "IEEE 802.1Q YANG Module Specifications", IEEE-Draft http://www.ieee802.org/1/files/public/docs2015/new-mholness-yang-8021Q-0515-v04.pdf, May 2015.

Appendix A. Acknowledgments

This document is derived from draft-openconfig-netmod-model-structure-00. We thank the Authors of that document and acknowledge their indirect contribution to this work. The authors include: Anees Shaikh, Rob Shakir, Kevin D'Souza, Luyuan Fang, Qin Wu, Stephane Litkowski and Gang Yan.

This work was discussed in and produced by the Routing Area Yang Architecture design team. Members at the time of writing included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Gang Yan.

The identityref approach was proposed by Mahesh Jethanandani.

The RFC text was produced using Marshall Rose's xml2rfc tool.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Lou Berger (editor)
LabN Consulting, L.L.C.

Email: lberger@labn.net

Dean Bogdanovic

Email: ivandean@gmail.com

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 21, 2017

V. Talwar
J. Kolhe
A. Shaikh
Google
J. George
Cisco

January 17, 2017

Use cases for gRPC in network management
draft-talwar-rtgwg-grpc-use-cases-01

Abstract

gRPC is an open, high-performance RPC framework designed for efficient low-latency cross-service communications. This document describes use cases for gRPC in network management and other services, particularly streaming telemetry.

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 July 21, 2017.

Table of Contents

1. Introduction

gRPC is a high performance universal RPC framework to connect distributed systems [GRPC-WWW]. gRPC emerged from an internal Google framework called Stubby which has been used to connect large numbers microservices running within and across data centers for over a decade. Having a uniform, cross-platform RPC infrastructure allowed Google to deploy fleet-wide improvements in efficiency, security, reliability and behavioral analysis critical to supporting the incredible growth of these services. gRPC is the next generation of Stubby, built in the open originally for services, as well as last mile computing use cases like mobile, browser, IOT [GRPC-DESIGN]. It is based on standards like HTTP/2 [RFC7540] and is extensible and pluggable by design.

This document describes use cases for the gRPC protocol [I-D.kumar-rtgwg-grpc-protocol] in network management, including monitoring, configuration management and programmatic operations. We

also summarize a number of additional use cases where gRPC is currently being applied.

2. gRPC use cases

2.1. Network management

Below we discuss several gRPC applications related to network management with a focus on monitoring and telemetry. gRPC is already implemented by several network device vendors as a primary transport for monitoring data based on the streaming telemetry paradigm.

2.1.1. Streaming telemetry motivation and overview

Network operations depend fundamentally on the availability of accurate, near real-time data to drive a variety of management systems, including traffic control systems, fault recovery systems, and demand and capacity forecasting systems. This data consists of information about the control plane (e.g., protocol operations), management plane (e.g., system availability, statistics, and counters), and data plane (e.g., packet and flow statistics).

In addition to the variety of data, the volume of monitoring and management data continues to increase significantly. Modern, high-density platforms with thousands of interfaces and numerous hardware and software modules means potentially collecting millions of objects and running tens of thousands of CLI commands every few minutes in a large-scale network. Network monitoring data is increasingly used to manage mission-critical systems such as real-time monitoring, centralized traffic engineering, server selection and load balancing. Hence it requires efficient, secure, and scalable mechanisms for data transport, encoding, and control.

Most networks rely on traditional management protocols such as SNMP [RFC1157] [RFC3410] for collecting monitoring data about the control and management planes, and SFLOW [RFC3176] or IPFIX [RFC7011] for the data plane. For control and management data in particular, SNMP is the primary tool, despite limitations which make it ill-suited for modern, large-scale networks, especially Web- and Internet-scale backbones, and large, high-capacity data center networks.

While SNMP is widely deployed and implemented in a variety of network environment, it suffers from a number of drawbacks:

- legacy implementations -- designed for devices with limited memory and little processing power; e.g., SNMPv2 supports multiple data items in a message, but is not optimized for high-volume data collection

- lack of discoverability -- discovering new elements requires walking the SNMP MIB periodically; on high-density platforms this is extremely computationally expensive

- lack of capability advertisements -- each object ID must be checked to know whether it is supported by the target platform

- rigid data structures -- whether using standard or vendor proprietary MIBs, the structure and format of the data cannot be easily extended or augmented

To address these drawbacks, a number of network operators proposed a new approach for network monitoring based on streaming telemetry (see an early proposal in [I-D.swhyte-i2rs-data-collection-system]). Streaming telemetry is based on a pub/sub push model in which target devices send data of interest over a streaming channel to a data collection system.

Some notable features of a streaming telemetry system include:

- targets stream data continuously based on a specified period (or as frequently as the target supports), or on a state change

- data is sent as soon as it is available, reducing the need to buffer, or to handle a single large for all data at once

- data may be sent incrementally, e.g., only for those data items that have changed

- ability to distribute the telemetry sources (e.g., directly to linecards) to avoid burdening the management CPU

- users issue subscription requests via RPC to the target to request only the data of interest

- data is exported in a well-structured, common format, e.g., based on YANG models of operational state data [I-D.openconfig-netmod-opstate]

- the target and collector communicate over a secure, authenticated, reliable channel that is long-lived and efficient

Streaming telemetry allows the network behavior to be observed through a time-series data stream. This is in contrast to the polling mechanism used in SNMP in which a monitoring client must periodically request the set of desired data, and walk the MIB to discover changes. The polling frequency is limited since the device

must be able to handle large requests for all interface or QoS counters, for example.

Open source implementations of streaming telemetry are currently being developed by several network vendors, including adapters to deliver data into time-series databases, messaging systems, and data visualization systems [ST-CISCO] [ST-JUNIPER] [ST-ARISTA].

2.1.2. Streaming telemetry with gRPC

gRPC provides a number of capabilities that makes it well-suited for network telemetry. Since its underlying transport is based on HTTP/2, it can exploit several key features:

- binary framing and header compression -- highly efficient encoding on the wire to enable bulk data transfer
- bidirectional streaming RPCs -- the target and collector can stream their data independently, and leverage application-level flow control
- flexible data encoding -- gRPC is payload agnostic, and can be used to transfer data encoded as XML, JSON, protocol buffer, or Thrift; as new data formats and encodings emerge for network data, the RPC layer can be easily adapted
- multi-language support -- open source gRPC IDLs are available for 10 programming languages, and service endpoints can be created on a number of operating systems, giving device vendors flexibility in implementation

gRPC-based telemetry stacks are now being implemented with some available as open source [ST-ARISTA]. A protocol specification for streaming telemetry based on gRPC is also available [GNMI-SPEC].

2.1.3. Network configuration management

gRPC offers a non-proprietary, modern alternative to vendor-specific configuration protocols or standards such as NETCONF [RFC6241] or TL1 [TL1]. Some of the benefits of using gRPC for configuration management include more flexible data encodings (e.g., no requirement to use XML), easier integration based on the large number of language implementations available, and more options for securing connections.

Several platforms now support gRPC configuration protocols using data based on YANG models [GRPC-CISCO] [GRPC-JUNIPER].

2.2. Additional use cases

2.2.1. Client Libraries for connecting polyglot systems

gRPC generates client libraries in 10 languages and thus allows developers to operate in their language of choice and system to communicate with any other system. These libraries offer idiomatic-to-language API surface such that every developer feels they are in their language native environment.

2.2.2. MicroServices

Designed as a general, high performance protocol to interconnect polyglot systems, gRPC is ideal for microservices communication, independent of where the services are deployed. A protocol that offers flow control, bidirectional streaming and a very compact serialization mechanism is ideally suited for connecting microservices at scale. It is already being adopted by large organizations like Square and Netflix for their microservices communications.

2.2.3. Browser and mobile applications communicating to gRPC Services

Mobile and Browser applications are becoming feature rich and more demanding by the day. User expectation is that apps are performant in various network conditions and drain minimal battery and computing power of device. gRPC provides native iOS and Android Java libraries for more efficient communication for applications with backend services such that battery, data are efficiently used and developers have more control of communication with servers using gRPC APIs.

2.2.4. High performance access to Cloud Services

The expectations from high request-rate cloud services like storage and pub/sub messaging systems are to be very efficient and low cost from a compute and networking point of view. Hence, gRPC based APIs are being used for services like Google Cloud BigTable and Google Cloud PubSub. External products like etcd (underlying storage system for kubernetes) also relies on gRPC.

2.2.5. Secure and low overhead communications in embedded systems

With its integrated authentication model and a IDL like nano-protobuf, gRPC could be ideal for secure device-to-device and device-to-cloud communication as well. This use case is still under development.

2.2.6. Unified inter-process and remote communication

gRPC can provide a unified programming model for both inter-process communication and remote service communication. This use case is still under development.

3. Security Considerations

As applied to network configuration and monitoring, any transport protocol and RPC framework must have support for secure, authenticated communication. gRPC supports a number of security mechanisms that are suitable for use in network management, including TLS-based transport, and client and server authentication. These will be detailed further in subsequent drafts.

4. IANA Considerations

None at this time. In the future, there may be proposals to designate specific application ports for gRPC-based telemetry and configuration traffic.

5. References

5.1. Normative references

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <http://www.rfc-editor.org/info/rfc1157>.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, DOI 10.17487/RFC3410, December 2002, <http://www.rfc-editor.org/info/rfc3410>.

[RFC3176] Phaal, P., Panchen, S., and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks", RFC 3176, DOI 10.17487/RFC3176, September 2001, <http://www.rfc-editor.org/info/rfc3176>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <http://www.rfc-editor.org/info/rfc7011>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

5.2. Informative references

[GRPC-DESIGN]
Ryan, L., "gRPC Motivation and Design Principles", September 2015, <http://www.grpc.io/posts/principles>.

[GRPC-WWW]
"gRPC Web site (grpc.io)", September 2015, <http://www.grpc.io>.

[ST-ARISTA]
"Arista Networks goarista GitHub repository", July 2016, <https://github.com/aristanetworks/goarista>.

[ST-CISCO]
"Cisco Systems bigmuddy GitHub repository", April 2016, <https://github.com/cisco/bigmuddy-network-telemetry-stacks>.

[GRPC-CISCO]
"Cisco Systems grpc GitHub repository", February 2016, <https://github.com/CiscoDevNet/grpc-getting-started>.

[ST-JUNIPER]
"Juniper Networks open-nti GitHub repository", July 2016, <https://github.com/Juniper/open-nti>.

[GRPC-JUNIPER]
Juniper Networks, "Next-Generation Network Configuration and Management", May 2016, <https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000632-en.pdf>.

[GNMI-SPEC]
Borman, P., Hines, M., Lebsack, C., Morrow, C., Shaikh, A., and R. Shakir, "gRPC Network Management Interface", November 2016, <https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md>.

[TL1] Telcordia, "GR-831-CORE - Operations Application Messages - Language for Operations Application Messages", November 1996, <http://telecom-info.telcordia.com/site-cgi/ido/docs.cgi?ID=SEARCH&DOCUMENT=GR-831&>.

[I-D.kumar-rtgwg-grpc-protocol]
Kumar, A., Kolhe, J., Ghemawat, S., and L. Ryan, "gRPC Protocol", kumar-rtgwg-grpc-protocol-00 (work in progress), July 2016.

[I-D.swhyte-i2rs-data-collection-system]
Whyte, S., Hines, M., and W. Kumari, "Bulk Network Data Collection System", draft-swhyte-i2rs-data-collection-system-00 (work in progress), October 2013.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

Appendix A. Change summary

A.1. Changes between revisions -00 and -01

- Added reference to gRPC Network Management Interface specification.
- Updated author contact information.

Authors' Addresses

Varun Talwar
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: varuntalwar@google.com

Jayant Kolhe
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jkolhe@google.com

Anees Shaikh
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: aashaikh@google.com

Joshua George
Cisco
170 W Tasman Dr
San Jose, CA 95134
US

Email: jgeorge@cisco.com