opsawg
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2016

P. Lapukhov
Facebook
R. Chang
Barefoot Networks

June 10, 2016

Data-plane probe for in-band telemetry collection
draft-lapukhov-dataplane-probe-01

Abstract

Detecting and isolating network faults in IP networks has traditionally been done using tools like ping and traceroute (see [RFC7276]) or more complex systems built on similar concepts of active probing and path tracing. While using active synthetic probes is proven to be helpful in detecting data-plane faults, isolating fault location is a much harder problem, especially in diverse networks with multiple active forwarding planes (e.g. IP and MPLS). Moreover, existing end-to-end tools do not generally support functionality beyond dealing with packet loss - for example, they are hardly useful for detecting and reporting transient (i.e. milli- or even micro-second) network congestion.

Modern network forwarding hardware can allow for more sophisticated data-plane functionality that provides substantial improvement to the isolation and identification capabilities of network elements. For example, it has become possible to encode a snapshot of a network element's state within the packet payload as it transits the device. One example of such state would be queue depth on the egress port taken by that specific packet. When combined with a unique device identifier embedded in the same packet, this could allow for precise time and topological identification of the the congested location within the network.

This document proposes a format for requesting and embedding telemetry information in active probes, i.e. packet designated for actively testing the network while not carrying application traffic. These active probes could be conveyed over multiple protocols (ICMP, UDP, TCP, etc.) and the document does not prescribe any particular transport. In addition, this document provides recommendations on handling the active probes by devices that do not support the required data-plane functionality.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Detecting and isolating faults in IP networks may involve multiple tools and approaches, but by far the two most popular utilities used by operators are ping and traceroute. The ping utility provides the basic end-to-end connectivity check by sending a special ICMP packet. There are other variants of ping that work using TCP or UDP probes, but may require a special responder application (for UDP) on the other end of the probed connection.

This type of active probing approach has its limitations. First, it operates end-to-end and thus it is impossible to tell where in the path the fault has happened from simply observing the packet loss ratios. Secondly, in multipath (ECMP) scenarios it can be difficult to fully and/or deterministically exercise all the possible paths connecting two end-points.

The traceroute utility has multiple variants as well - UDP, ICMP and TCP based, for instance, and special variant for MPLS LSP testing. Practically all variants follow the same model of operations: varying TTL field setting in outgoing probes and analyzing the returned ICMP unreachable messages. This does allow isolating the fault down to the IP hop that is losing packets, but has its own limitations. As with the ping utility, it becomes complicated to explore all possible ECMP paths in the network. This is especially problematic in large Clos fabric topologies that are very common in large data-center networks. Next, many network devices limit the rate of outgoing ICMP messages as well as the rate of "exception" packets "punted" to the control plane processor. This puts a functional limit on the packet rate that the traceroute can probe a given hop with, and hence impacts the resolution and time to isolate a fault. Lastly, the treatment for these control packets is often different from the packets that take regular forwarding path: the latter are normally not redirected to the control plane processor and handled purely in the data-plane hardware.

Modern network processing elements (both hardware and software based) are capable of packet handling beyond basic forwarding and simple

header modifications. Of special interest is the ability to capture and embed instantaneous state from the network element and encode this state directly into the transit packet. One example would be to record the transit device's name, ingress and egress port identifiers, queueing delays, timestamps and so on. By collecting this state along each network device in the path, it becomes trivial to trace a probe's path through the network as well as record transit device characteristics. Extending this model, one could build a tool that combines the useful properties of ping and traceroute using a single packet flight through the network, without the constraints of control plane (aka "slow path") processing. To aid in the development of such tooling, this document defines a format for requesting and embedding telemetry information in the body of active probing packets.

2. Data plane probe

This section defines the structure of the active data-plane probe.

2.1. Probe transport

This document does not prescribe any specific encapsulation for the data-plane probe. For example, the probe could be embedded inside a UDP packet, or within an IPv6 extension header.

2.2. Probe structure

The probe consists of a fixed-size "Header" and arbitrary number of variable-length "telemetry data frames" following the header. Frames are variable length, and each frame, in turn, consists of multiple "telemetry record" fields defined below in this document. The records are added per the request of the telemetry information specified in the header.

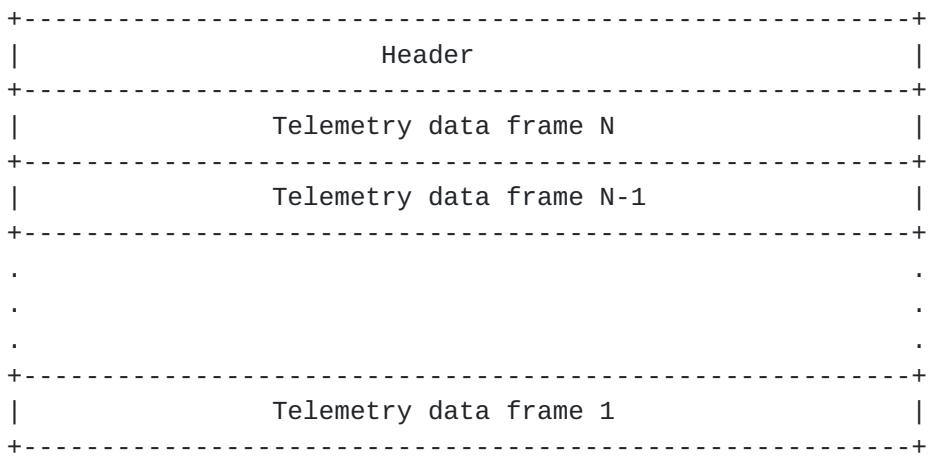

Figure 1: Probe layout

Notice that the first frame is at the end of the packet. For efficient hardware implementation, new frames are pushed onto the stack at each hop. This eliminates the need for the transit network elements to inspect the full packet and allows for arbitrarily long packets as the MTU allows.

2.3. Header Format

The probe payload starts with a fixed-size header. The header identifies the packet as a data-plane probe packet, and encodes basic information shared by all telemetry records.

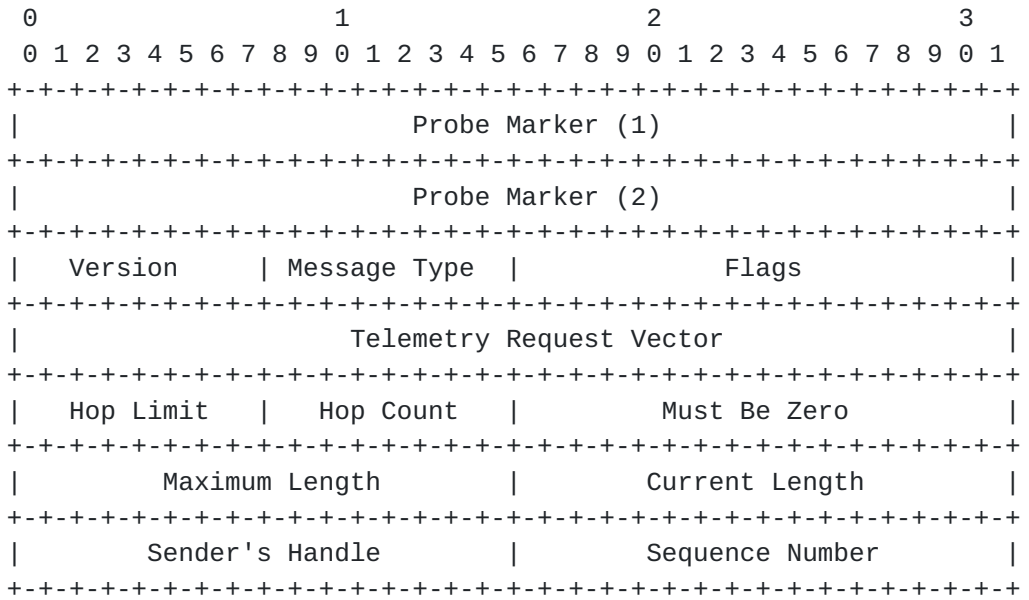

Figure 2: Header Format

(1) The "Probe Marker" fields are arbitrary 32-bit values generally used by the network elements to identify the packet as a probe packet. These fields should be interpreted as unsigned integer values, stored in network byte order. For example, a network element may be configured to recognize a UDP packet destined to port 31337 and having 0xDEAD 0xBEEF as the values in "Probe Marker" field as an active probe, and treat it respectively.

(2) "Version Number" is currently set to 1.

(3) The "Message Type" field value could be either "1" - "Probe" or "2" - "Probe Reply"

(4) The "Flags" field is 8 bits, and defines the following flags:

(5)

(1) "Overflow" (O-bit) (least significant bit). This bit is set by the network element if the number of records on the packet is at the maximum limit as specified by the packet: i.e. the packet is already "full" of telemetry information.

(6) "Telemetry Request Vector" is a 32-bit long field that requests well-known inband telemetry information from the network elements on the path. A bit set in this vector translates to a request of a particular type of information. The following types/bits are currently defined, starting with the least significant bit first:

(1) Bit 0: Device identifier.

(2) Bit 1: Timestamp.

(3) Bit 2: Queueing delay.

(4) Bit 3: Ingress/Egress port identifiers.

(5) Bit 31: Opaque state snapshot request.

(7) "Hop Limit" is defined only for "Message Type" of "1" ("Probe"). For "Probe Reply" the "Hop Limit" field must be set to zero. This field is treated as an integer value representing the number of network elements. See the Section 4 section on the intended use of the field.

(8) The "Hop Count" field specifies the current number of hops of capable network elements the packet has transit through. It

begins with zero and must be incremented by one for every network element that adds a telemetry record. Combined with a push mechanism, this simplifies the work for the subsequent network element and the packet receiver. The subsequent network element just needs to parse the template and then insert new record(s) immediately after the template.

(9) The "Max Length" field specifies the maximum length of the telemetry payload in bytes. Given that the sender knows the minimum path MTU, the sender can set the maximum of payload bytes allowed before exceeding the MTU. Thus, a simple comparison between "Current Length" and "Max Length" allows to decide whether or not data could be added.

(10) The "Current Length" field specifies the current length of data stored in the probe. This field is incremented by eacn network element by the number of bytes it has added with the telemetry data frame.

(11) The "Sender's Handle" field is set by the sender to allow the receiver to identify a particular originator of probe packets. Along with "Sequence Number" it allows for tracking of packet order and loss within the network.

2.4. Telemetry Data Frame and Telemetry Data Records

Each telemetry data frame is constructed by concatenating multiple telemetry data record, per the request in "Telemetry Request Vector" fields of the dataplane probe header. The frame starts with a 16-bit length field, which reflects the frame size in bytes, excluding the length of the field itself. Following the "Frame Length" field is a "Telemetry Response Vector" field: this vector corresponds to the records the network element was capable of recording in the frame. The body of the frame is constructed by appending fixed-size records corresponding to every bit set in "Telemetry Response Vector". All of the records, except the one requested by 31st bit ("Opaque State Snapshot") are fixed size, with their lengths defined in Section 3. The order of the records in the frame follows the order of the bits in the "Telemetry Request Vector" (also reflected in "Telemetry Response Vector"). Finally, if requested, a variable-length field is appended at the end of the frame, with the length field occupying the first 8 bits. This "length" field reflects the length of the opaque data excluding the length field itself.

If inserting a new telemetry record would cause "Current Length" to exceed "Max Length", no record is added and the overflow "O-bit" must be set to "1" in the probe header.

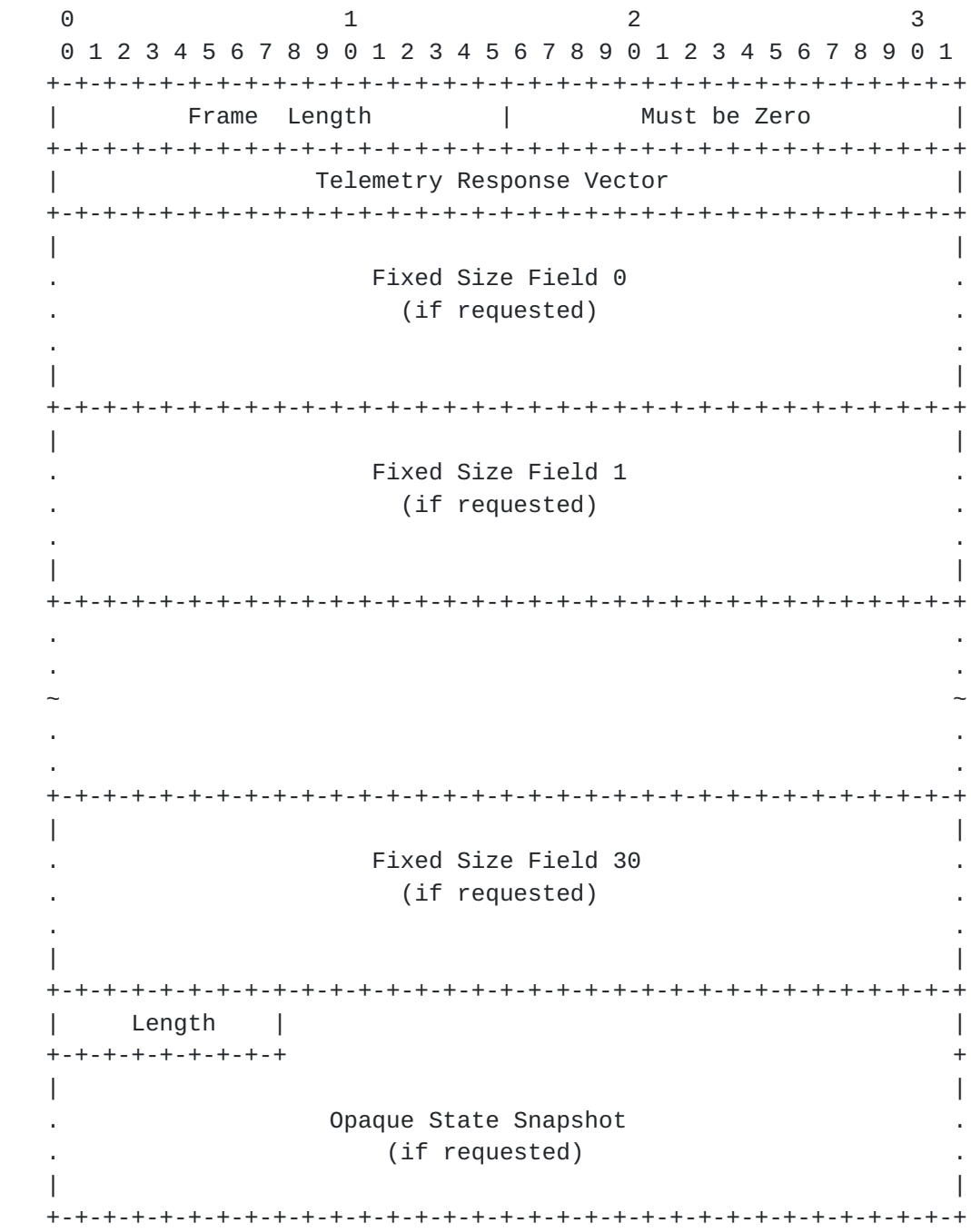

Figure 3: Telemetry Frame Format

3. Telemetry Record Types

This section defines some of the telemetry record types that could be supported by the network elements.

3.1. Device Identifier

This record is used to identify the device reporting telemetry information. This document does not prescribe any specific identifier format. In general, it is expected to be configured by the operator. The length of this record is 32-bit.

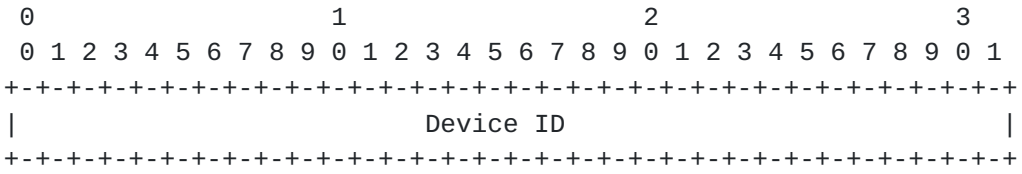

Figure 4: Device Identifier

3.2. Timestamp

This telemetry record encodes the time data associated with the packet. Most existing hardware support timestamping for IEEE1588. To leverage existing hardware capabilities, packet receive time is stored similarly as 48-bits of seconds, 32-bits of nanoseconds, and residence time is in 48-bits of nanoseconds. The length of this record is 128 bits.

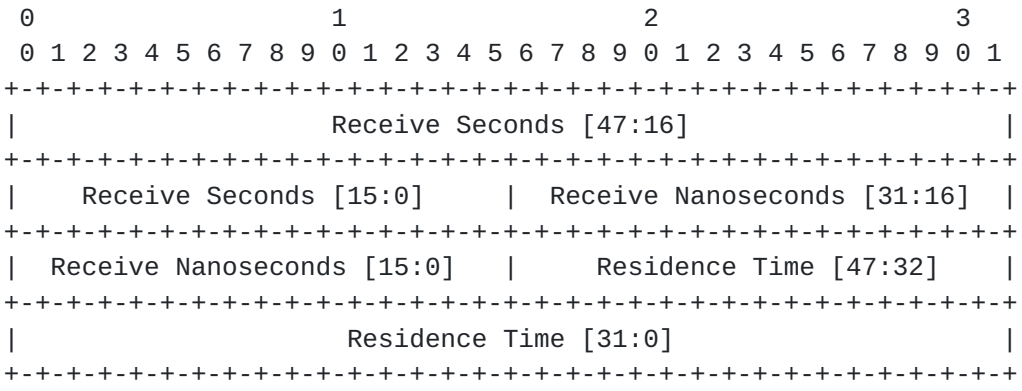

Figure 5: Timestamp

3.3. Queueing Delay

This record encodes the amount of time that the frame has spent queued in the network element. This is only recorded if packet has been queued, and defines the time spent in memory buffers. This could be helpful to detect queueing-related delays in the network. If the queueing delay exceeds the maximum number of 2+ seconds allowed by the 31-bit number, the network element must set the overflow "O-bit". In case of the cut-through switching operation this must be set to zero. The length of this record is 32 bits.

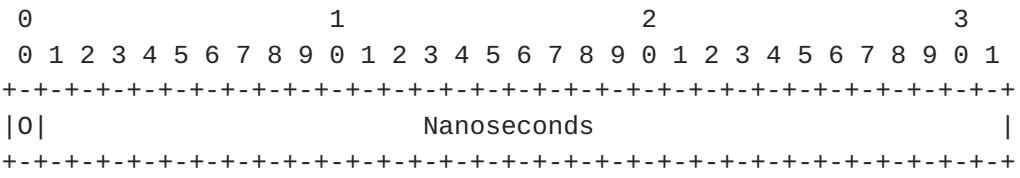

Figure 6: Queueing Delay

3.4. Ingress/Egress Port IDs

This record stores the ingress and egress physical ports used to receive and send packet respectively. Here, "physical port" means a unit with actual MAC and PHY devices associated - not any logical subdivision based, for example, on protocol level tags (e.g. VLAN). The port identifiers are opaque, and defined as 16-bit entries. For example, those could be the corresponding SNMP ifIndex values. The length of this record is 32 bits.

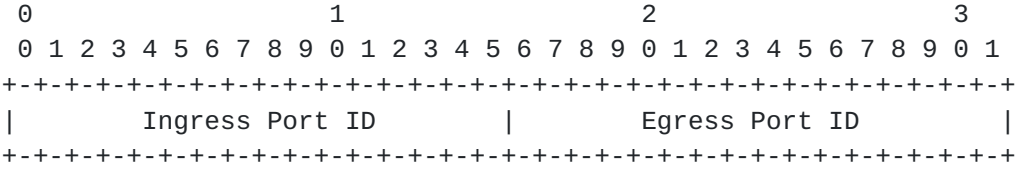

Figure 7: Ingress/Egress Port IDs

3.5. Opaque State Snapshot

This record has variable size. It allows the network element to store arbitrary state in the probe, without a pre-defined schema. The schema needs to made known to the analyzer by some out-of-band means. The 16-bit "Schema Id" field in the record is supposed to let the analyzer know which particular schema to use, and it is expected to be configured on the network element by the operator. This ID is expected to be configured on the device by the network operator.

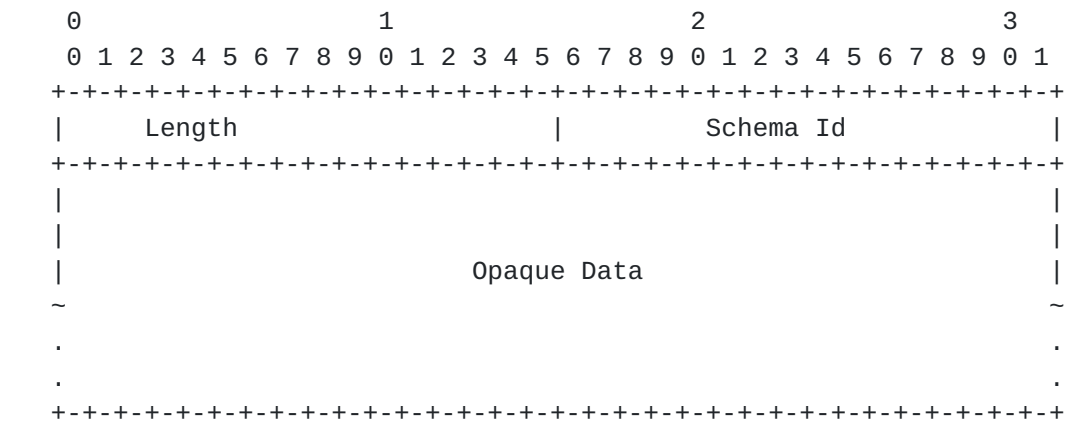

Figure 8: Opaque State

4. Operating in loopback mode

In "loopback" mode the flow of probes is "turned back" at some network element. The network element that "turns" packets around is identified using the "Hop Limit" field. The network element that receives a "Probe" type packet having "Hop Limit" value equal to "Hop Count" is required to perform the following:

Change the "Message Type" field to "Probe Reply", and keep the "Hop Limit" at zero.

Swap the destination/source IP addresses in the transport header to send the packet back to the originator.

Add a new telemetry data frame corresponding to the new forwarding information.

This way, the original probe is routed back to originator. Notice that the return path may be different from the path that the original probe has taken. This path will be recorded by the network elements as the reply is transported back to the sender. Using this technique one may progressively test a path until its breaking point.

If a network element is incapable of redirecting packets back to the originator, another option would be exporting those packets to a network analyzer device, using some sort of encapsulation header.

5. Processing Probe Packet

5.1. Detecting a probe

As mentioned previously, a combination of techniques need to be used to differentiate the active probes. This may include, but should not be limited to using just the known position of "Probe Id" fields.

6. Non-Capable Devices

Non-capable devices are those that cannot process a probe natively in the fast-path data plane. Further, there could be two types of such devices: those that can still process it via the control-plane software, and those that can not. The control-plane processing should be triggered by use of the "Router-Alert" option for IPv4 of IPv6 packets (see [RFC2113] or [RFC2711]) added by the originator of the probe. A control-plane capable device is expected to interpret and fill-in as much telemetry-record data as it possibly could, given the limited abilities.

Network elements that are not capable of processing the data-plane probes are expected to perform regular packet forwarding. If a network element receives a packet with the router-alert option set, but has no special configuration to detect such probes, it should process it according to [RFC6398]. Absence of the router alert option leaves the non dataplane-capable devices with the only option of processing the probe using traditional forwarding.

7. Handling data-plane probes in the MPLS domain

In general, the payload of an MPLS packet is opaque to the network element. However, in many cases the network element still performs a lookup beyond the MPLS label stack, e.g. to obtain information such as L4 ports for load balancing. It may be possible to perform data-plane probe classification in the same manner, additionally using the "Probe Marker" to distinguish the probe packets.

In accordance to [RFC6178] Label Edge Routers (LERs) are required not to impose an MPLS router-alert label for packets carrying the router-alert option. It may be beneficial to enable such translation, so that an end-to-end validation could be performed if a control-plane capable MPLS network element is present on the probe's path.

8. Multi-chip device considerations

TBD

9. IANA Considerations

None

10. References

10.1. Normative References

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, DOI 10.17487/RFC2113, February 1997, <http://www.rfc-editor.org/info/rfc2113>.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <http://www.rfc-editor.org/info/rfc2711>.

[RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <http://www.rfc-editor.org/info/rfc6398>.

[RFC6178] Smith, D., Mullooly, J., Jaeger, W., and T. Scholl, "Label Edge Router Forwarding of IPv4 Option Packets", RFC 6178, DOI 10.17487/RFC6178, March 2011, <http://www.rfc-editor.org/info/rfc6178>.

10.2. Informative References

[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>.

Authors' Addresses

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

Email: remy@barefootnetworks.com