opsawg
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

P. Lapukhov
Facebook

March 18, 2016

Data-plane probe for in-band telemetry collection
draft-lapukhov-dataplane-probe-00

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 has proven to be 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 enable 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 elements forwarding state within the packet payload as it transits the device. One example of such device/network 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 standard format for embedding telemetry information in UDP-based probing packets, i.e. packets designated for testing the network while not carrying application traffic. These active probes could be conveyed over multiple protocols (ICMP, UDP, TCP, etc.) but this document specifically focuses on UDP, given its simple semantics. 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

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on September 19, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 quite 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, queue depths, 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 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 packets.

2.1. Probe transport

This document assumes the use of IP/UDP for data-plane probing (either IPv4 or IPv6). A receiving application may listen on a pre-defined UDP port to collect and possibly echo back the information embedded in the probe. One potential limitation to this methodology is the size of the probe packet, as some data-plane faults may only impact packets of a given size or range of sizes. In this case, the data-plane probe may not be able to detect such issues, given the requirement to pre-allocate storage in the packet body.

2.2. Probe structure

The sender is responsible for constructing a packet large enough to hold all records to be added by the network elements. Concurrently, the probes must not exceed the minimum MTU allowed along the path, so it is assumed that the sender either knows the needed MTU or relies on well-known mechanisms for path MTU discovery. After adding the mandatory protocol (IP, UDP, etc.) headers, the packet payload is built according to the following layout:

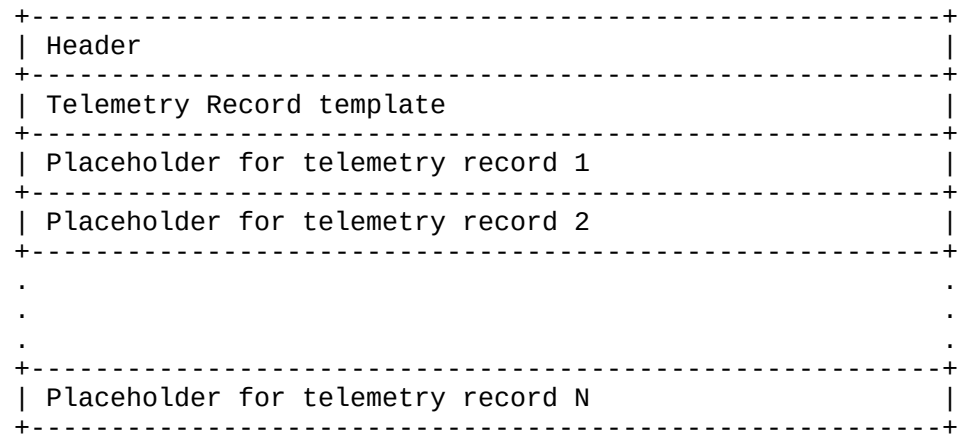

Figure 1: Probe layout

Notice that all record placeholders are equal size, as prescribed by the telemetry record template, and that space for those must be pre-allocated by the sender of the packet. Each record corresponds to a single network element on the path from sender to receiver of the packet.

2.3. Header Format

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

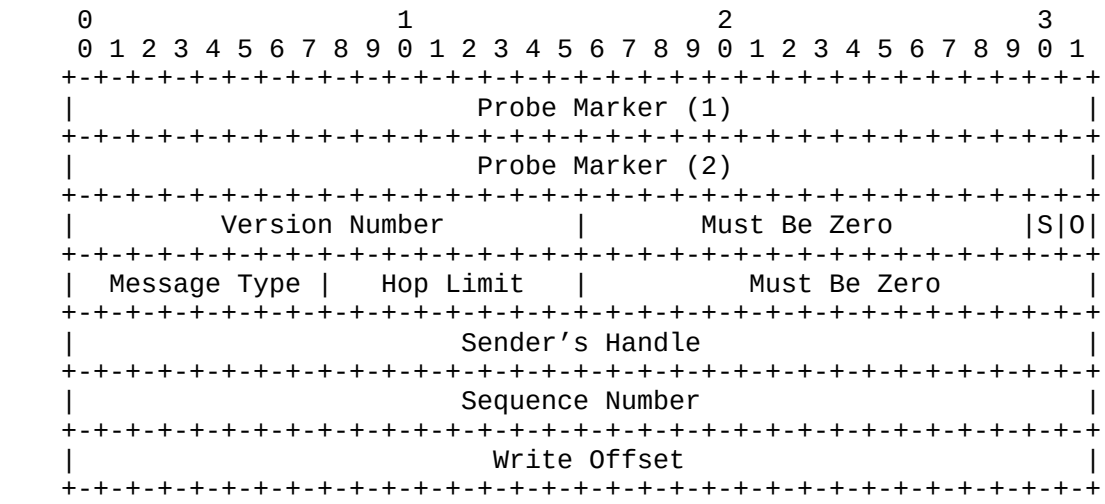

Figure 2: Header Format

(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 "Global Flags" field is 8 bits, and defines the following flags:

    (1) "Overflow" (O-bit) (least significant bit). This bit is set by the network element if there is no record placeholder available: i.e. the packet is already "full" of telemetry information.

    (2) "Sealed" (S-bit). This bit instructs the network element to forward the packet WITHOUT embedding telemetry data, even if it matches the probe identification rules. This mechanism could be used to send "realistic" probes of arbitrary size after the network path associated with the combination of source/destination IP addresses and ports has been previously established. The network element must not inspect the "Telemetry Record Template" field for "sealed" probes.

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

(5) "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 and decremented by every network element in the path as "Probe" propagates. See the Section 4 section on the intended use of the field.

(6) 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.

(7) The "Write Offset" field specifies the offset for the next telemetry record to be written in the probe packet body. It counts from the start of the packet body and must be initially set to the first octet after the "Record Template" field. It must be incremented by every network element that adds a telemetry record, without overflowing the storage. This simplifies the work for the subsequent network element - it just needs to parse the template and then add the data at the "Write Offset".

2.4. Telemetry Record Template

The following figure defines the "Record Template". This template uses type-length fields to describe the telemetry data records as added by network elements. The most significant bit in the "Type" field must be set to zero.

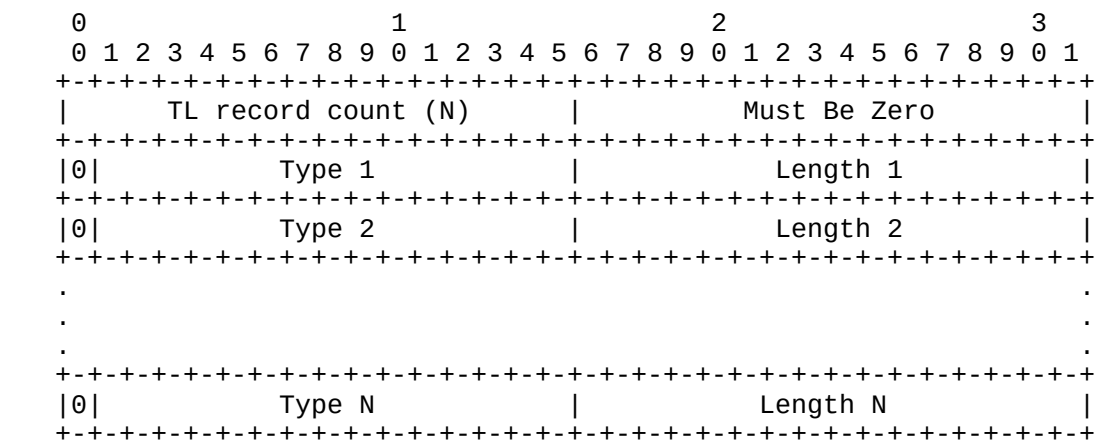

Figure 3: Record Template

2.5. Telemetry Record

This section defines the structure of a telemetry record. Every network element capable of reporting inband telemetry data must add a record as defined in the "Record Template" to the probe packet. The new record must be inserted at the "Write Offset" position in the packet payload, with the "Write Offset" subsequenly incremented by the size of the new record. The order of TLV elements must follow the order prescribed by the Figure 3 portion of the probe packet. The most significant bit in the type field ("S-bit") must be set to "1" if the network element was able to understand and record the requested telemetry type. That bit must be set to zero otherwise, along with the contents of the "Value" field. The length field is the TLV field length including the "Type" and "Length" fields.

If writing a new telemetry record to the packet body would cause it to exceed the packet size, no record is added and the overflow "O-bit" must be set to "1" in the probe header.

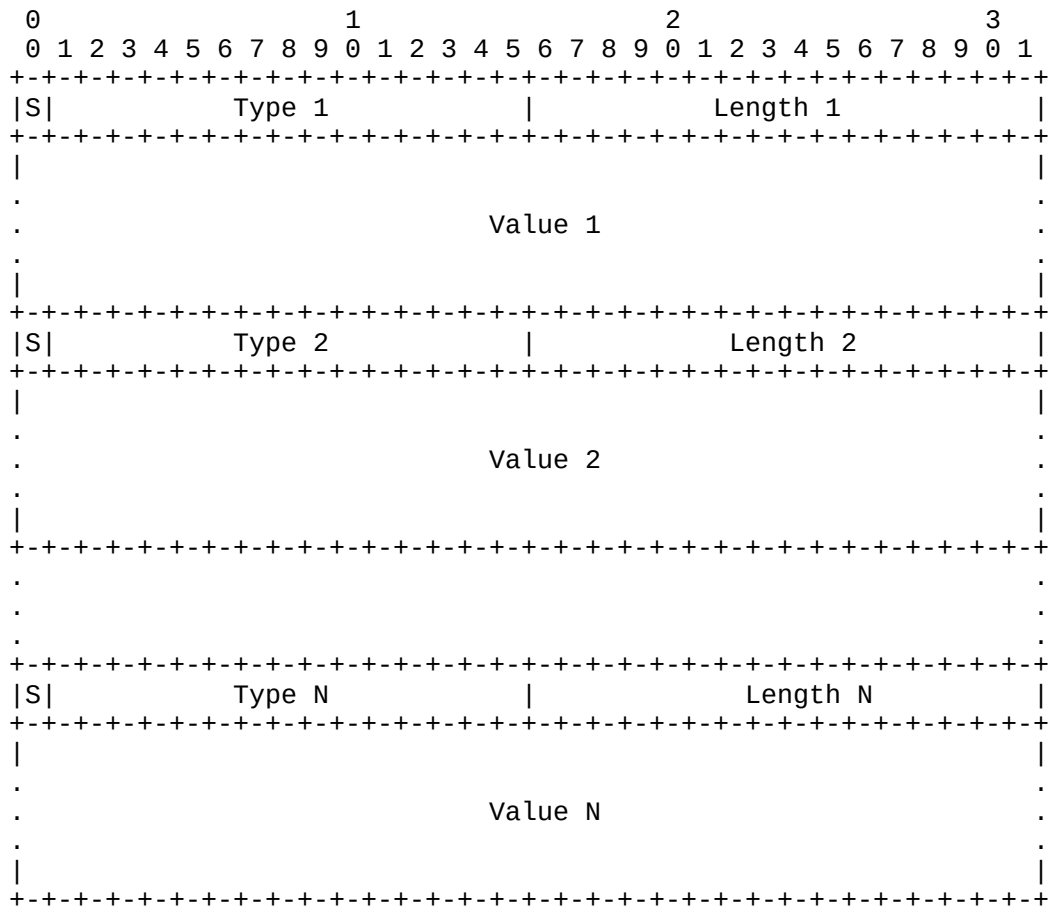

Figure 4: Telemetry Record Format

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 is used to identify the device reporting telemetry information. This document does not prescribe any specific identifier format.

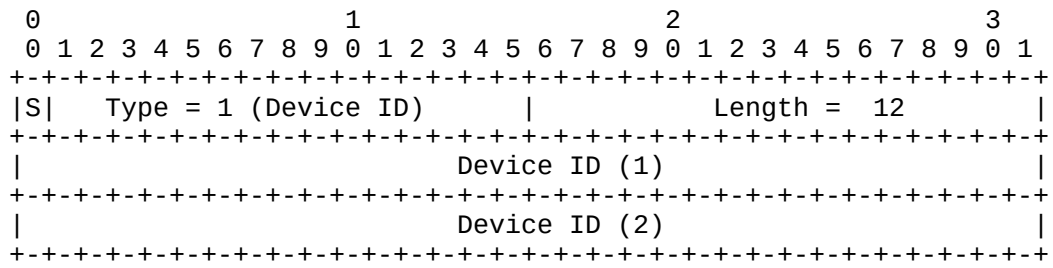

Figure 5: Device Identifier

3.2. Timestamp

This telemetry record encodes the time that the packet enters and leaves the device, in UTC. The "entering" time is recorded when the L2 header enters the processing pipeline. The "exit" time is recorded when the network elements starts serializing L2 header on egress port.

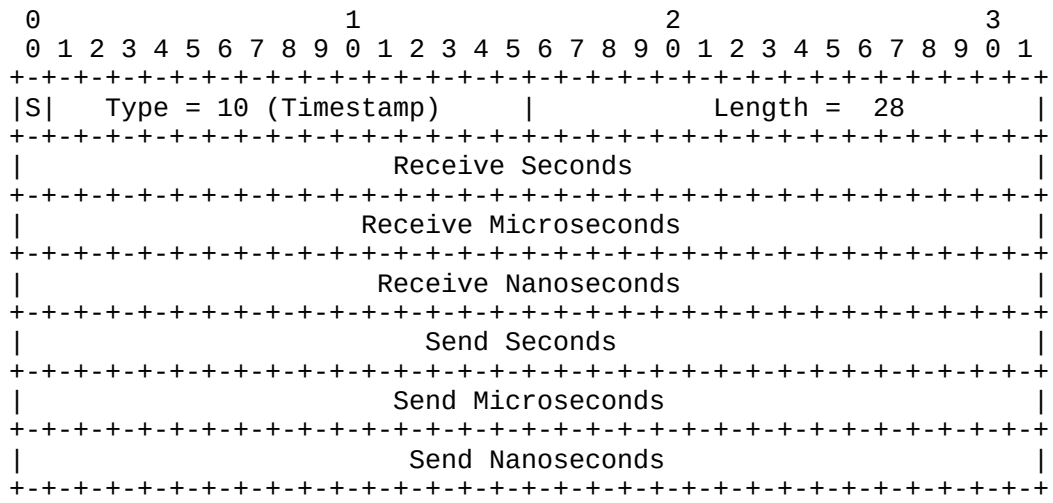

Figure 6: Timestamp

3.3. Queueing Delay

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. In case of the cut-through switching operation this must be set to zero.

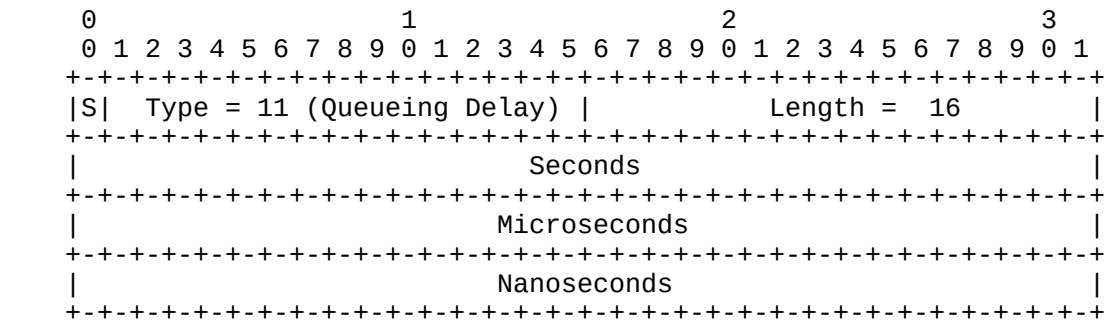

Figure 7: Queueing Delay

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 32-bit entries.

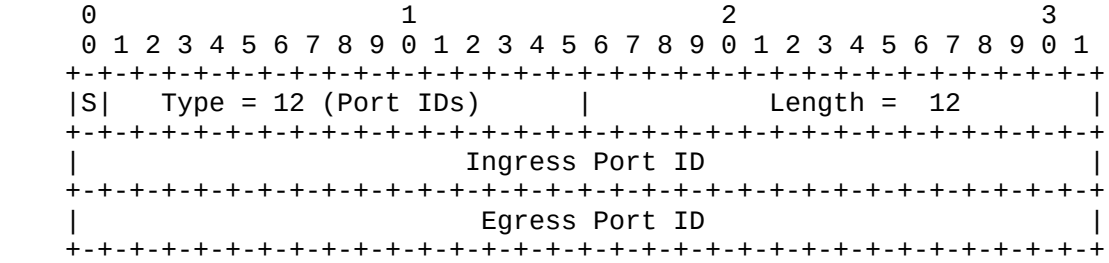

Figure 8: Ingress/Egress Port IDs

3.5. Forwarding Information

Records defined in this section require the network element to store forwarding information that was used to direct the packet to the next-hop. In the network that uses multiple forwarding plane implementations (e.g. IP and MPLS) the originator of the probe is required to populate the record template with all kinds of forwarding information it expects in the path. The network elements then populate the entries they know about, e.g. in IPv4-only network the "IPv6 Route" record will be left unfilled, and so will be "MPLS Route".

3.5.1. IPv6 Route

This record stores the IPv6 route that has been used for packet forwarding. If not used, then S-bit is set to zero, along with the value field.

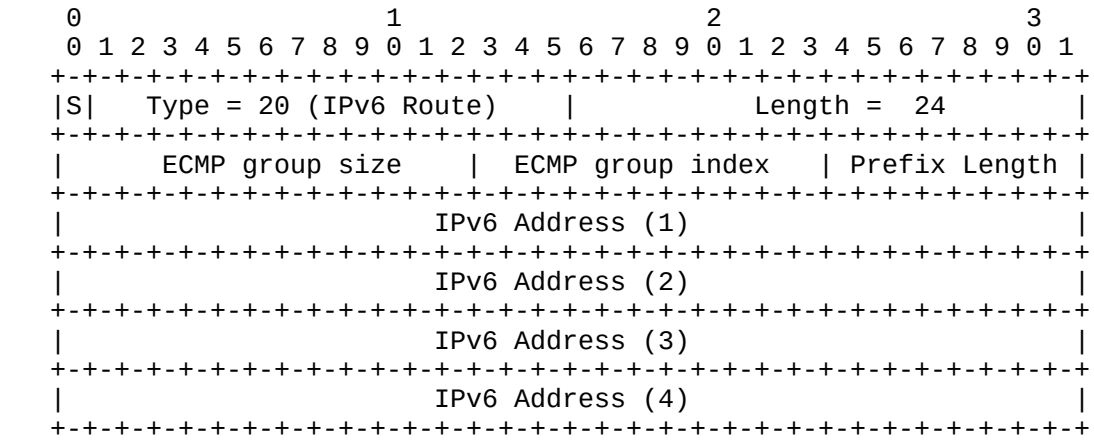

Figure 9: IPv6 Route

3.5.2. IPv4 Route

This record stores the IPv4 route that has been used for packet forwarding. If not used, then S-bit is set to zero, along with the value field.

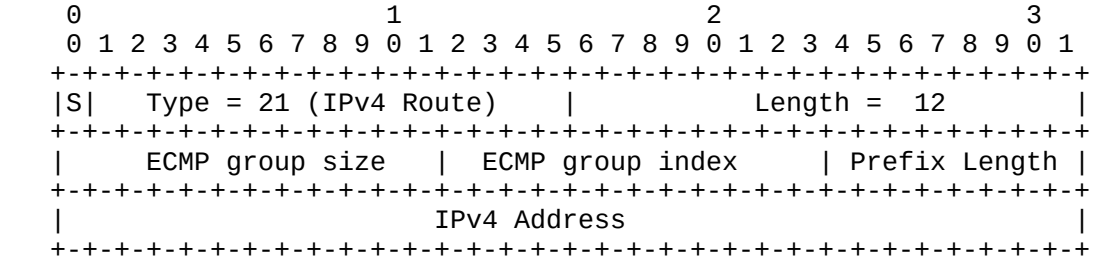

Figure 10: IPv4 Route

3.5.3. MPLS Route

This record stores the MPLS label mapping that has been used for packet forwarding. It is possible that inbound or outbound label set set to zero, if it was not used (e.g. on ingress or egress of the domain). At the edge of IP2MPLS or MPLS2IP domain it is expected

that the device would fill in the "MPLS Route" telemetry record along with the corresponding "IPv6 Route" or "IPv4 Route" records.

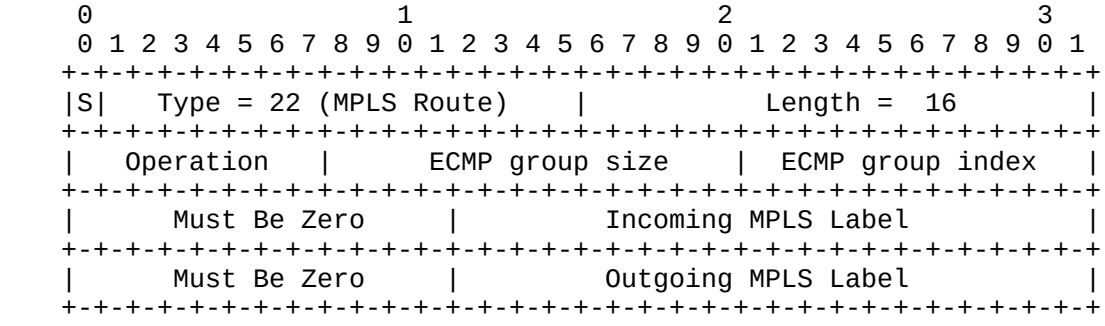

Figure 11: MPLS Route

There are three MPLS operations defined

"1" - Push

"2" - Pop

"3" - Swap

4. Operating in loopback mode

In "loopback" mode the flow of probes is "turned back" at a given 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 of "1" is required to perform the following:

Change the "Message Type" field to "Probe Reply" and set the "Hop Limit" to zero.

Swap the destination/source addresses and port values in the IP/UDP headers of the probe packet.

Add a telemetry record as required using the newly build IP/UDP headers to determine 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. Unlike

the traditional traceroute utility, however, the returning packets are the original probes, not the ICMP messages.

5. Processing Probe Packet

5.1. Detecting a probe

Since the probe looks like a regular UDP packet, the data-plane hardware needs a way to recognize it for special processing. This document does not prescribe a specific way to do that. For example, classification could be based on only the destination UDP port, or using more complex pattern matching techniques, e.g matching on the contents of "Probe Marker" field.

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. Acknowledgements

The author would like to thank L.J. Wobker and Changhoom Kim for reviewing and providing valuable comments for the initial version of this document.

11. References

11.1. 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>.

11.2. 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>.

Author’s Address

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

Email: petr@fb.com