

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	3
2. Data plane probe	4
2.1. Probe transport	4
2.2. Probe structure	4
2.3. Header Format	5
2.4. Telemetry Record Template	7
2.5. Telemetry Record	8
3. Telemetry Record Types	9
3.1. Device Identifier	9
3.2. Timestamp	10
3.3. Queueing Delay	10
3.4. Ingress/Egress Port IDs	11
3.5. Forwarding Information	11
3.5.1. IPv6 Route	12
3.5.2. IPv4 Route	12
3.5.3. MPLS Route	12
4. Operating in loopback mode	13
5. Processing Probe Packet	14
5.1. Detecting a probe	14

6. Non-Capable Devices	14
7. Handling data-plane probes in the MPLS domain	14
8. Multi-chip device considerations	15
9. IANA Considerations	15
10. Acknowledgements	15
11. References	15
11.1. Normative References	15
11.2. Informative References	15
Author's Address	15

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:

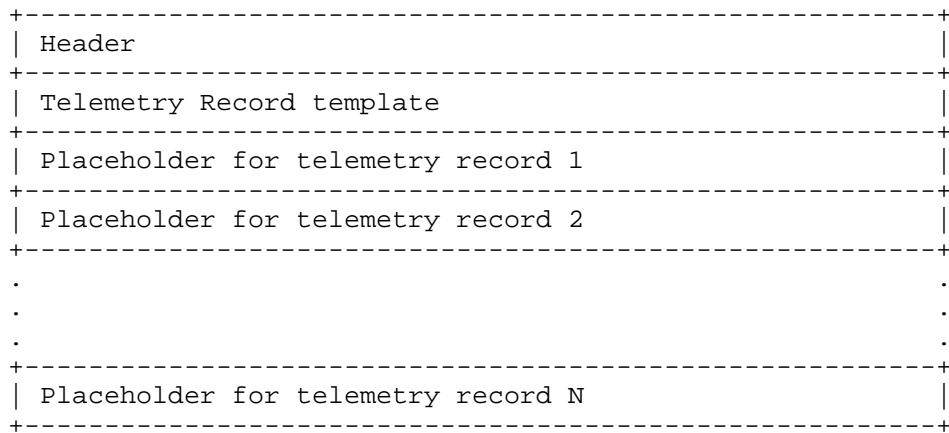


Figure 1: Probe layout

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.

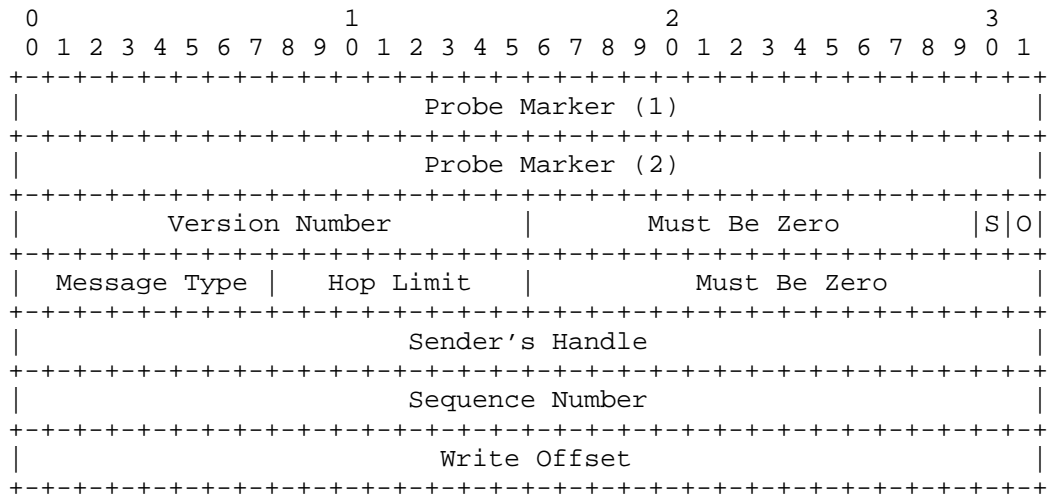


Figure 2: Header Format

- (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.

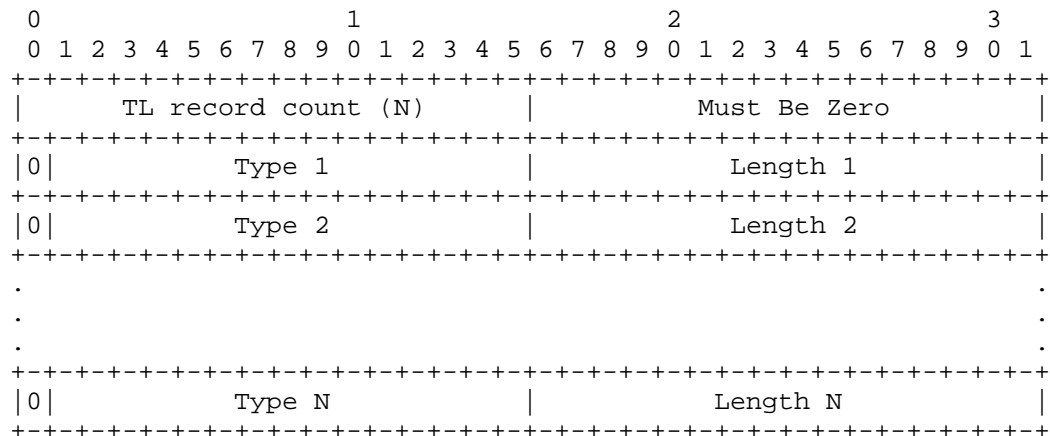


Figure 3: Record Template

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" subsequently 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.

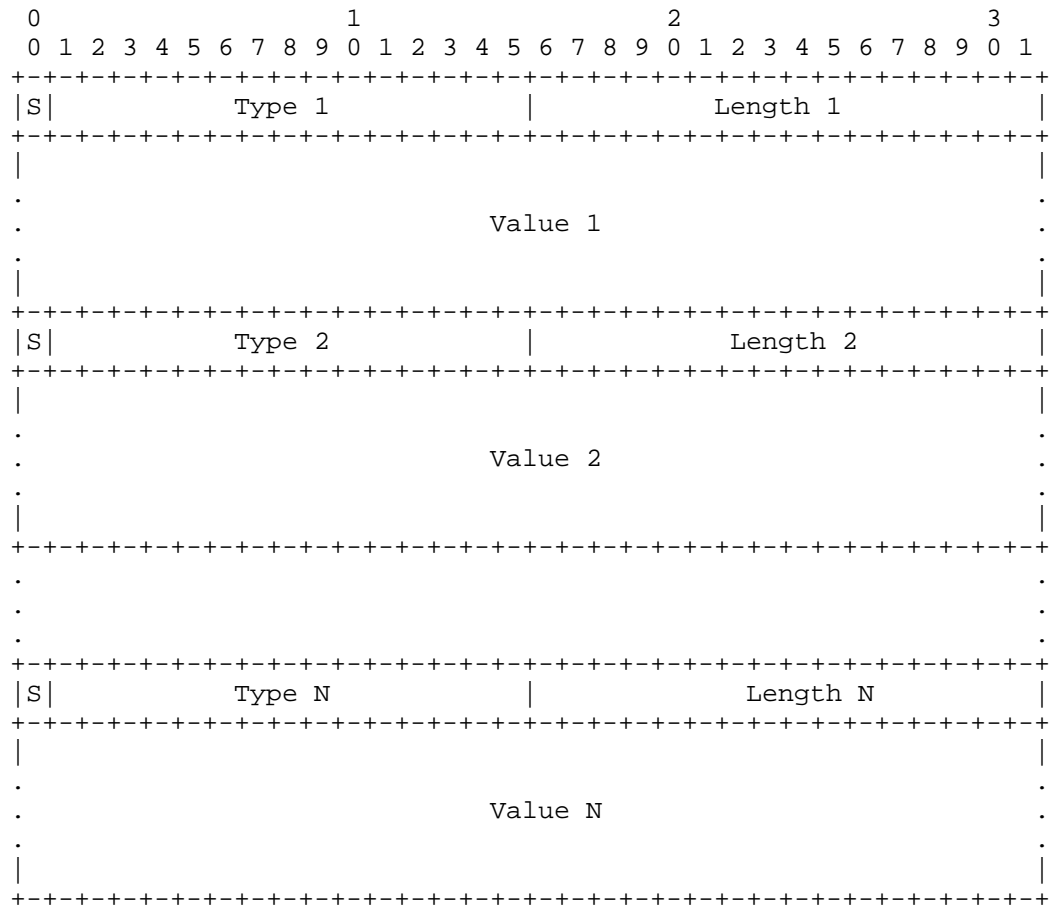


Figure 4: Telemetry Record Format

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.

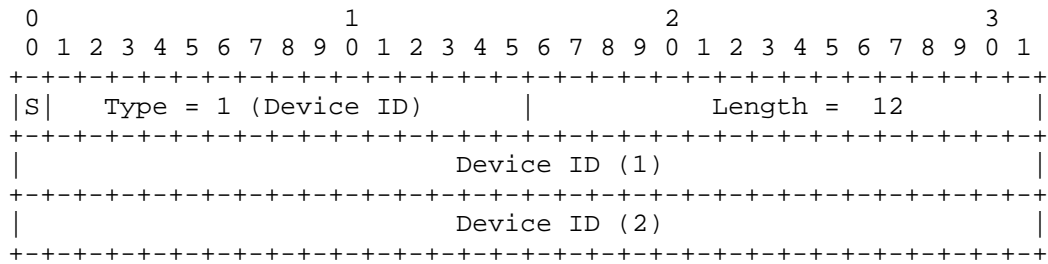


Figure 5: Device Identifier

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.

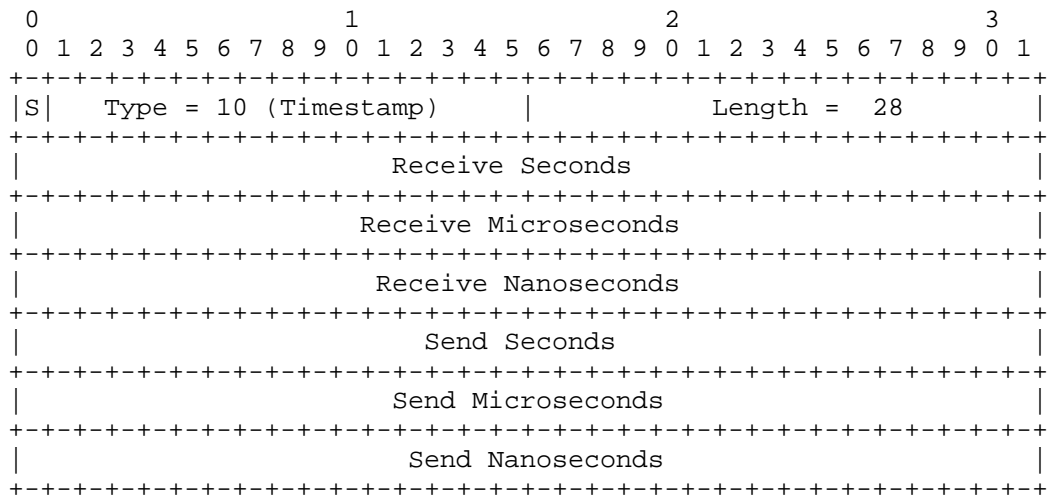


Figure 6: Timestamp

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.

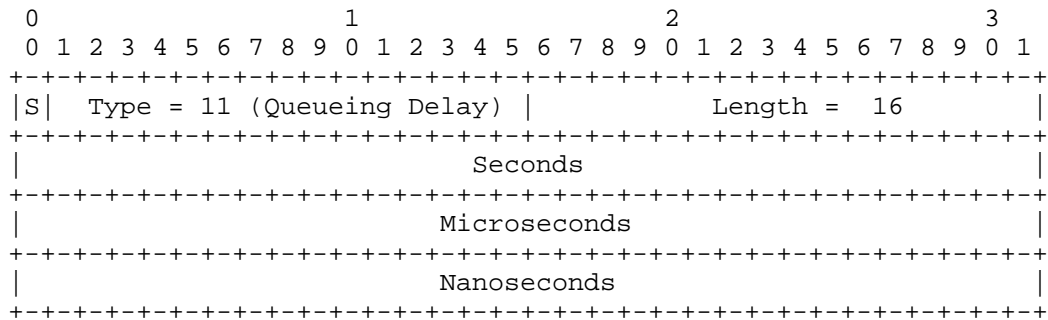


Figure 7: Queueing Delay

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.

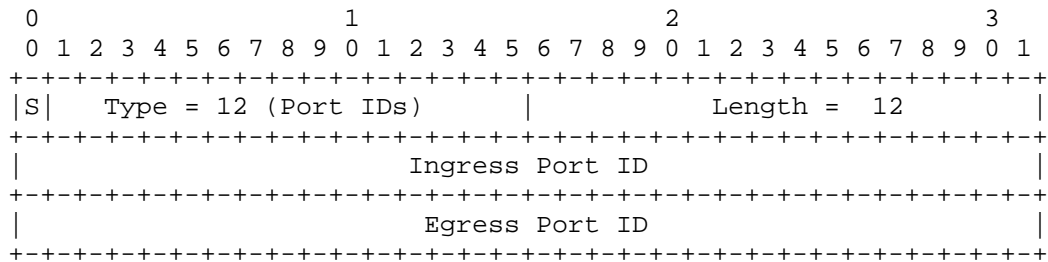


Figure 8: Ingress/Egress Port IDs

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.

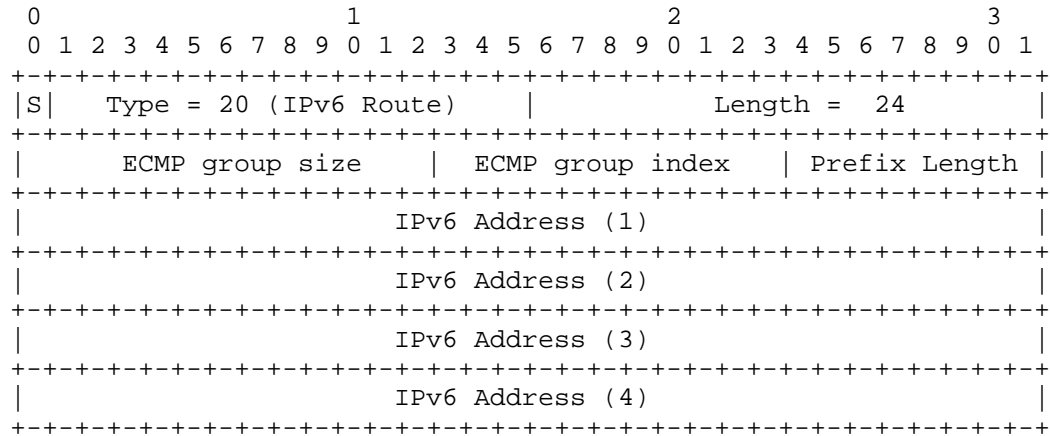


Figure 9: IPv6 Route

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.

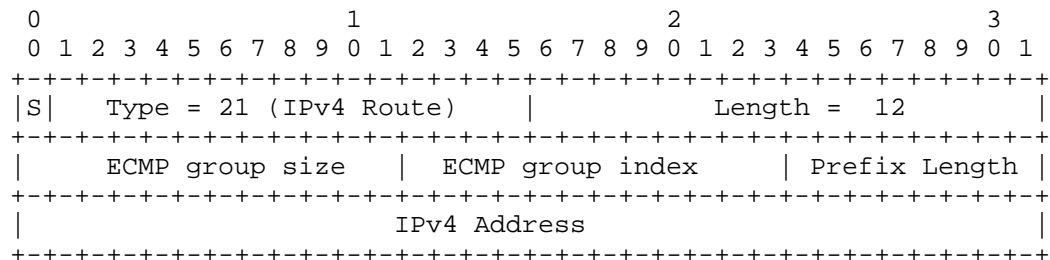


Figure 10: IPv4 Route

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.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|S|   Type = 22 (MPLS Route)   |           Length = 16           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Operation   |   ECMP group size   |   ECMP group index   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Must Be Zero   |   Incoming MPLS Label   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Must Be Zero   |   Outgoing MPLS Label   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

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 or 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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 24, 2016

E. Lear
Cisco Systems
January 21, 2016

Manufacturer Usage Description Framework
draft-lear-mud-framework-00

Abstract

A key presumption of the Internet architecture has been that devices are general purpose computers. By constraining the set of devices that connect to the Internet to non-general purpose devices, we can introduce a set of network capabilities that provides an additional layer of protection to those devices. One such capability is the Manufacturer Usage Description (MUD). This work builds on many existing network capabilities so as to be easily deployable by all involved. The focus of this work is primarily, but not exclusively, in the realm of security; and again primarily, but not exclusively, relating to smart objects.

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 24, 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	2
1.1. A Simple Example	3
1.2. Determining Intended Use	4
1.3. Types of Policies	5
2. The Manufacturer Usage Description Architecture	6
2.1. What does a MUD URI look like?	7
2.2. Communicating to the Manufacturer	7
2.3. Using YANG-based XML	7
2.4. Instantiating Policy	8
2.5. When Configuration Can't Change	8
3. Related Work	9
3.1. Relationship to ANIMA	9
4. Security Considerations	9
5. IANA Considerations	10
6. Acknowledgments	10
7. Informative References	10
Author's Address	11

1. Introduction

The Internet has largely been constructed on general purpose computers; those devices that may be used for a purpose that is specified by those who buy the device. [RFC1984] presumed that an end device would be most capable of protecting itself. This made sense when the typical device was a workstation or a mainframe, and it continues to make sense for general purpose computing devices today, including laptops, smart phones, and tablets.

[RFC7452] discusses design patterns for, and poses questions about, smart objects. Let us then posit a group of objects that are specifically not general purpose computers. These devices therefore have a purpose to their use. By definition, therefore, all other purposes are NOT intended. The combination of these two statements can be restated as a manufacturer usage description (MUD) that can be applied at various points within a network. Although this memo may seem to stress access requirements, usage intent also consists of quality of service needs a device may have.

We use the notion of "manufacturer" loosely in this context, to simply mean the entity or organization that will state how a device

is intended to be used. In the context of a lightbulb, this might indeed be the lightbulb manufacturer. In the context of a smarter device that has a built in Linux stack, it might be integrator of that device. The key points are that the device itself is expected to serve a limited purpose, and that there may exist an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The converse statement holds that general computing systems will benefit very little from MUD, as their manufacturers cannot envision a specific communication pattern to describe.

The intent of MUD is to therefore solve for the following problems:

- o Substantially reduce the threat surface on a device entering a network to those communications intended by the manufacturer.
- o Provide for a means to scale network policies to the ever-increasing number types of devices in the network.
- o Provide a means to address at least some vulnerabilities in a way that is faster than it might take to update systems. This will be particularly true for systems that are no longer supported by their manufacturer.
- o Keep the cost of implementation of such a system to the bare minimum.

No matter how good a MUD-enabled network is, it will never replace the need for manufacturers to patch vulnerabilities. It may, however, provide network administrators with some additional protection when those vulnerabilities exist.

1.1. A Simple Example

A light bulb is intended to light a room. It may be remotely controlled through the network; and it may make use of a rendezvous service of some form that an app on smart phone accesses. What we can say about that light bulb, then, is that all other network access is unwanted. It will not contact a news service, nor speak to the refrigerator, and it has no need of a printer or other devices. It has no Facebook friends. Therefore, an access list applied to it that states that it will only connect to the single rendezvous service will not impede the light bulb in performing its function, while at the same time allowing the network to provide both it and other devices an additional layer of protection.

1.2. Determining Intended Use

The notion of intended use is in itself not new. Network administrators apply access lists every day to allow for only such use. This notion of white listing was well described by Chapman and Zwicky in [FW95]. Programmatically profiling systems have existed for years as well. These systems make use of heuristics that take at least some time to assert what a system is.

A system could just as easily tell the network what sort of protection it requires without going into what sort of system it is. This would, in effect, be the converse of [RFC7488]. In seeking a general purpose solution, however, we assume that a device has so few capabilities that it will implement the least necessary capabilities to function properly. This is a basic economic constraint. Unless the network would refuse access to such a device, its developers would have no reason to implement such an approach. To date, such an assertion has held true.

If the network does not apply heuristics and a device is not capable of articulating what it needs from the network, perhaps there is a third approach that builds on capabilities already in both. There are four such potential capabilities for the network to determine what sort of client it has:

1. For those devices that are meant to operate in a secure environment [IEEE8021X] and [IEEE8021AR] provides a means for certificate-based device identification.
2. In the absense of DHCP in IPv6 (e.g., stateless address selection), [IEEE8021AB] can be used to learn the same information.
3. In the IP network context, every device needs an IP address. [RFC2131] specifies the dynamic host configuraiton protocol, necessary for all IPv4 and IPv6 implementations. Client use of a DHCP option would inform the network of what the device thinks it is, and provide a pointer to additional policy information.
4. Finally, for equipment that does not emit any information, it is possible for the access switch to proxy the information into the system.

With these capabilities, a device may impart some piece of information to the network. In the immortal words of David John Wheeler, "All problems in computer science can be solved by another level of indirection, except of course for the problem of too many indirections." Our means of providing this level of indirection is a

Universal Resource Identifier (URI) [RFC3986] that references a file put in place by someone who knows something about the device - the manufacturer. As we will later discuss, we can later relax whether it is indeed the manufacturer who is specifying the URI.

With a simple resolution of a URI, a file is retrieved. We are now to the point in the discussion where we have to decide how the manufacturer expresses intent. We have already stated that Things themselves have limited capabilities. Let us also assume that we in the networking business wish to stand on the shoulders of giants and also not reinvent the wheel. While such a wheel is not perfectly rounded for our purposes, YANG models [RFC6020] and their derivative XML provide sufficient richness for the manufacturer to clearly state at least simple intent. They are thus our starting point.

1.3. Types of Policies

Once we know how to determine intended use and who can determine it, there is still the question of what that sort of policies can in fact be intended. At least initially, we envision that as a beginning host-level access policies. The manufacturer may specify either specific hosts or certain classes. An example of a class might be "devices of a specified manufacturer type", where the manufacturer type itself is indicated simply by the authority of the MUD-URI. Another example might to allow or disallow local access. Just like other policies, these may be combined. For example:

```
Allow access to host controller.example.com with QoS AF11
Allow access to devices of the same manufacturer
Allow access to and from controllers who need to speak COAP
Allow access to local DNS/DHCP
Deny all other access
```

To add a bit more depth that should not be a stretch of anyone's imagination, one could also make use of port-based access lists. Thus a printer might have a description that states:

```
Allow access for port IPP or port LPD
Allow local access for port HTTP
Deny all other access
```

In this way anyone can print to the printer, but local access would be required for the management interface.

Other non-access policies may be possible as well. For instance, suppose a manufacturer is able to make use of an authentication infrastructure. That could be specified in the usage description such that the details could be filled in by the controller. In

addition, QoS policies are sufficiently mature and ubiquitous as to be valuable in this context as well. And so for instance, for voice/video services:

Set QoS AF13 to SIP-GW.EXAMPLE.COM

The converse highlights a design consideration: policies that are articulated by the manufacturer must be ubiquitously understood, or they may not be applied. That is- applying half a policy is not safe.

2. The Manufacturer Usage Description Architecture

With these components laid out we now have the basis for an architecture. This leads us to ASCII art.

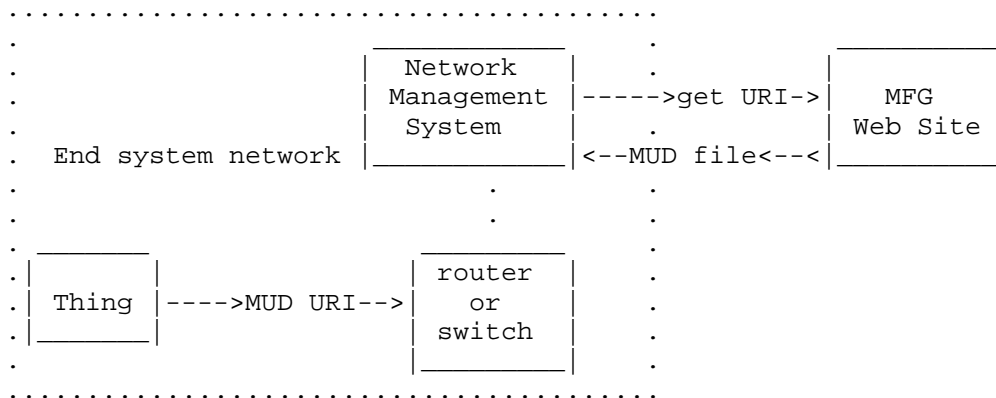


Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URIs and forwards them to the network management system for processing. This happens in different ways, depending on how the URI is communicated. For instance, in the case of DHCP, the DHCP server might receive the URI and then process it. In the case of IEEE 802.1X, the switch would tunnel the URI to the authentication server, who would then process it.

The information returned by the web site is valid for the duration of the device's connection, or as specified in the description. Thus if the device is mobile, when it moves on, any configuration in the switch is removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web site is run by or on behalf of the manufacturer. Its domain name is that of the authority found in the MUD URI. For legacy cases where Things cannot emit a URI, if the switch is able to determine the appropriate URI, it may proxy it, the trivial cases being a map between some registered device or port and a URI.

2.1. What does a MUD URI look like?

To begin with, MUD takes full advantage of both the https: scheme and the use of .well-known. HTTPS is important in this case because men in the middle could otherwise harm the operation of a class of devices. .well-known is used because we wish to add additional structure to the URI. And so the URI is specified in draft-lear-netmod-mud-pre0. It looks like this:

`https://manufacturer.example.com/.well-known/mud/v1/model/version#extra`

"model" represents a device model as the manufacturer wishes to represent it. It could be a brand name or something more specific. "version" provides a means to indicate what version the product is. Specifically if it has been updated in the field, this is the place where evidence of that update would appear. Once again, the field is opaque. From a controller standpoint, therefore, only comparison and matching operations are safe.

2.2. Communicating to the Manufacturer

We assume that the the manufacturer has at its disposal a web service running atop port 443 with standard HTTPS semantics, and that its capabilities are at par with today's web servers. We further assume that this web server has no semantic understanding itself of MUD. This poses us a particular challenge: either we are to cast in stone the model that is put in place, or we must find a mechanism by which the switch or its controller can choose an appropriate set of capabilities.

2.3. Using YANG-based XML

Because NETCONF is well distributed within network infrastructure and YANG has become the accepted way to generate schema for NETCONF, these we attempt to adapt the protocol and the modeling language, respectively. At some point in the near future, it will likely be the case that XML gives way to JSON[RFC7159]. YANG can be used for either, and so it seems even more appropriate to make good use of it. This work makes use of XML because of the breadth of toolsets available, and not for any love of angle brackets. That is subject to change.

The descriptions specified in MUD files should be based on relatively ubiquitous network capabilities. Access lists are such an example, and QoS policies follow closely behind. For security purposes, these policies must only apply to the device that is connecting, and should not modify other parts of a network element's configuration. The key scaling properties here are as follows:

- o A manufacturer should only have to maintain and distribute one file per device model.
- o A network management system need not retrieve that same file when the same model appears in multiple places in its network.
- o Updates should occur at periods specified by the manufacturer to manage load.

2.4. Instantiating Policy

The network management system receiving the MUD file must convert it into an access list that a network element understands, and apply it to an appropriate interface, limiting its applicability only to the device in question. In some cases, the policies will be abstract. For example, "local" would be translated to the set of networks that are within the same administrative domain. It is the network management system's responsibility to see that the configuration is removed when the device detaches, and that the configuration is consistent with other policies that might apply to that device. Importantly, network management systems should always defer to the network administrator's wishes. As such, a conflicting policy should not be deployed, but rather logged.

Human interaction may be required in some cases. In the home, one could imagine description simply being instantiated, whereas in the enterprise, someone may need to review the description before it is applied.

It is distinctly possible that a highly advanced enterprise would ignore any manufacturer recommendations altogether but still use the URI received from devices as a classifier.

2.5. When Configuration Can't Change

In some environments it may not be possible for policy reasons to make changes to network elements to instantiate usage descriptions as a means of enforcement. These very same descriptions may be used as a means to audit activity of a device to determine whether or not it is acting in accordance with the the manufacturer's intent.

3. Related Work

3.1. Relationship to ANIMA

[I-D.ietf-anima-bootstrapping-keyinfra] specifies a means by which a device is configured with appropriate credentials for a given network. This work specifies a means to configure the network rather than the device. In fact, one key assumption of MUD is that it will be extremely painful to make any end system changes.

4. Security Considerations

The three mentioned means for a device to emit a MUD URI each have their own security properties, and will be discussed in separate drafts. A risk they share in common, however, is that the URI could point to a site that contains malware. To avoid such problems, several countermeasures are suggested:

- o All XML should be well formed and validated against appropriate schema.
- o Only XML whose capability name spaces are known should be processed at all.
- o Any names within the XML (such as access-list or ACE names) should be replaced with local instances, so as to avoid overwriting existing configuration.
- o Controllers are encouraged to validate the reputation of the authority of the web site.

By emitting a URI the device may identify itself to an interloper. As it happens, most devices can be relatively easily fingerprinted based on their communications patterns. However, if this is of concern, devices should emit the URI to network controllers over secure channels.

Use of certain operations, such as SameManufacturer scale less well than others. Frequent connects and disconnects could cause configuration storms. To address this risk, as the number of changes increase, modifications to devices other than the one connecting should decrease or simply be scheduled. In as much as this is an attack, it can also be mitigated through device authorization mechanisms such as 802.1X.

5. IANA Considerations

The IANA is requested to enjoy a coffee or tea, as there is nothing in this document that otherwise requires their attention.

6. Acknowledgments

The author thanks Bernie Volz, Eric Vyncke, and Cullen Jennings for their helpful suggestions.

7. Informative References

[FW95] Chapman, D. and E. Zwicky, "Building Internet Firewalls", January 1995.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-ietf-anima-bootstrapping-keyinfra-01 (work in progress), October 2015.

[IEEE8021AB]
Institute for Electrical and Electronics Engineers, "Link Layer Discovery Protocol", 2005.

[IEEE8021AR]
Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.

[IEEE8021X]
Institute for Electrical and Electronics Engineers, "Port Based Network Access Control", 1998.

[RFC1984] IAB and , "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <<http://www.rfc-editor.org/info/rfc1984>>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<http://www.rfc-editor.org/info/rfc2131>>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<http://www.rfc-editor.org/info/rfc3986>>.

- [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>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.
- [RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <<http://www.rfc-editor.org/info/rfc7452>>.
- [RFC7488] Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control Protocol (PCP) Server Selection", RFC 7488, DOI 10.17487/RFC7488, March 2015, <<http://www.rfc-editor.org/info/rfc7488>>.

Author's Address

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Internet Engineering Task Force
Internet Draft
Intended status: Informational
Expires: July 2016

P. Unbehagen
D. Romascanu, Ed.
J. Seligson
C. Keene
Avaya
N. Bragg
Ciena
L. Beliveau
Windriver
January 2016

Auto-attach using LLDP with IEEE 802.1aq SPBM networks
draft-unbehagen-lldp-spb-02.txt

Abstract

This informational document describes a method that allows for the automatic attachment of end stations and network devices to a core network based on the individual services that are run or configured on the stations or devices, and the mapping of the services to the managed paths in the network.

Specifically the document describes a compact method based on the IEEE802.1AB Link Layer Discovery Protocol (LLDP) to automatically attach network devices not supporting IEEE 802.1ah to individual services in an IEEE 802.1aq Shortest Path Bridging (SPB) network.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-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 July 26, 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	3
2. Terminology	3
3. Requirements Language.....	4
4. Auto Attachment Framework and Applicability	4
5. Auto Attachment Architecture.....	6
5.1. Element Discovery.....	6
5.2. Service Requests.....	7
5.2.1. Element Inactivity Timeout.....	7
5.3. Server Mapping Request Processing	7
5.4. Server Mapping Response Processing	8
5.5. Service Mapping Timeout.....	8
6. Auto Attach LLDP Extensions.....	9
6.1. AA Element TLV	9
6.2. I-SID/VLAN Assignment TLV.....	12
7. Security Considerations.....	14
7.1. TLV Security Considerations.....	15
8. IANA Considerations.....	15
9. Further and Related Work.....	15
10. Log of Changes	15
10.1. Changes between draft-unbehagen-lldp-spb-01 and 02.....	15
11. Acknowledgments	16
12. References	16
12.1. Normative References.....	16
12.2. Informative References.....	17

1. Introduction

This informational document describes a compact method of using IEEE802.1AB Link Layer Discovery Protocol (LLDP) with IEEE 802.1aq Shortest Path Bridging (SPB) network to automatically attach network devices not supporting IEEE 802.1ah to individual services in a SPB network. These network devices typically do not support SPBM, MAC-in-MAC (802.1ah), nor I-SID usage and therefore cannot easily take advantage of the SPB infrastructure without manual configuration of attachment of VLANs to I-SIDs in multiple locations. A motivation for this draft is to suggest a useful means to simplify and automate connections to PBB L2VPN based service networks such as those defined in SPBM-EVPN.

2. Terminology

802.1aq - defines a technology for providing a link state protocol for the control of a common Ethernet switching layer.

802.1ah - Provider Backbone Bridges (PBBs), MAC-IN-MAC encapsulation

AAC - Auto Attach Client agent that resides on a non-SPB/PBB capable element that uses LLDPDUs to request I-SID assignment for the VLANs which have been configured on its network port.

AAS - Auto Attach Server agent that processes VLAN to I-SID requests from AAC elements that are connected to a SPB BEB

BCB - Backbone Core Bridge

BEB - Backbone Edge Bridge

B-TAG - Backbone VLAN Tag

C-TAG - Customer VLAN Tag

Element - Any end device or network node that may implement the auto attach functionality

I-SID - Backbone Service Instance Identifier

IS-IS - Intermediate System to Intermediate System Protocol

LAN - Local Area Network

LLDP - IEEE 802.1AB Link Layer Discovery Protocol

SPB - IEEE 802.1aq Shortest Path Bridging

SPBM - Shortest Path Bridging, MAC mode

VLAN - Virtual Local Area Network

3. Requirements Language

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

4. Auto Attachment Framework and Applicability

This section provides an overview of the behavior of auto attachment functionality into a SPBM [802.1aq] environment and is not intended to be interpreted as normative text. Note that while the scheme proposed in this document is applicable at the link layer and is the subject of proposed standardization work in the IEEE 802.1 Working Group [AA], it is consistent with work proposed in the IETF in the Network Virtualization Overlays (nvo3) and Interface to the Routing System (i2rs) Working Groups.

The purpose of Auto Attach is to allow a non-SPB device to connect to an SPB capable networking device. The non-SPB device is called an AA client (AAC) and the SPB capable networking device is called the AA server (AAS). An AA Client is a non-SPBM device that supports some form of I-SID/VLAN binding definition and, if connectivity permits, has the ability to advertise this data to a directly connected AA Server. An AA Server is a SPBM device that potentially accepts externally generated I-SID/VLAN assignments that can be used for automated configuration purposes. The client identifies itself to the server and then requests VLAN ID to SPB ISID binding(s). The server will either accept or reject each binding request. If accepted, any traffic on the (locally significant) VLAN is forwarded through the SPB cloud on the specified ISID.

A prototype of the extension proposed in the memo was successfully implemented and tested with Open vSwitch. IEEE 802.1aq SPB software is available from multiple vendors of Ethernet switches to connect end devices and non-SPB compliant switches to the SPB enabled backbone network. Edge switches in SPBM that utilize the 802.1ah PBB encapsulation are referred to as Backbone Edge Bridges (BEB). In

support of SPBM, these bridges map a VLAN ID on the UNI to an I-SID (Individual Service ID), as defined in IEEE 802.1ah. In order to facilitate an automatic way in which a AAC can request individual service connectivity from an SPBM Backbone Edge Bridge BEB acting as a AAS, this method of using IEEE 802.1AB Link Layer Discovery Protocol (LLDP) with IEEE 802.1aq Shortest Path Bridging network can be used. These widely deployed client devices typically do not support SPBM, IEEE 802.1ah and therefore cannot easily take advantage of the SPB infrastructure without manual configuration of attachment of VLANs to I-SIDs in multiple locations.

Elements that utilize this automated method for service assignment pass this data to attached SPBM capable BEB nodes where the mappings are processed and approved or rejected. Specific actions are taken on the non-SPBM devices, referred to as Auto Attach Clients (AAC), as well as the SPBM device, referred to as Auto-Attach Server (AAS), based on the outcome of the mapping request.

Conceptual SPB Auto Attach Model

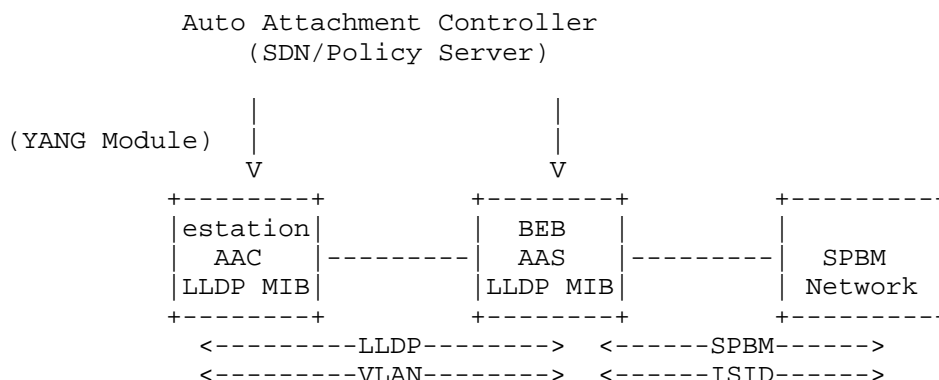


Figure 1: LLDP exchanges trigger IS-IS SPBM announcements

Figure 1 depicts a conceptual example of the process where an AAC can use LLDP to communicate the need to connect a VLAN to the appropriate I-SID on the SPB BEB it is attached to on its network uplink port. The IEEE 802.1AB Link Layer Discovery Protocol (LLDP) and the LLDP MIB are part of the Auto Attachment Framework and will be implemented in the Backbone Edge Bridges (BEB) and the Backbone Core Bridges (BCB).

An Auto Attach Client (AAC) can run in any device (end station -estation) that connects to an SBPM network. One field of applications can be for example Internet of Things (IoT) devices that connect to a network. The service association is automated with relative low resources, allowing connection of the devices to the appropriate network services and applications.

The different classes of devices that run AACs may be configured by means that are specific for the respective classes of applications or services. A YANG module will be defined as part of the Auto Attachment framework allowing for standard-based interaction with the Auto Attachment Controllers, which are instantiated as policy or Software Define Networks (SDN) servers.

5. Auto Attachment Architecture

5.1. Element Discovery

The first stage of establishing AA connectivity involves element discovery. An Auto Attach agent resides on all capable elements. Server agents control the Auto Attach (AA) of VLANs to I-SIDs on themselves when enabled to accept and process such requests from AAC elements. Typically this is done through a global service setting and through per-port settings that control the transmission of information in LLDPDUs on the appropriate links that interface AAC's and AAS's.

Once the required AA settings are enabled on the elements (e.g., the AA service and the per-port AA settings) the AA agent on each element type, both AAC and AAS, advertises its capabilities (i.e., server/client) through LLDPDU packets to each other.

Following discovery of AA capabilities by both the AAC and the AAS, the AA agent on each element is aware of all AA services currently provided by the network elements to which it is directly connected. Based on this information, an AAC agent can determine whether Auto Attach data, namely locally administered I-SID/VLAN assignments, should be exported to the AAS that is associated with an SPBM BEB to which it is attached to on its network uplink ports.

Initial Auto Attach functionality, when enabled, can be used to extract management VLAN data from the primary AA server advertisements and can use this data to update the in-band management VLAN and initiate IP address acquisition using techniques such as DHCP.

5.2. Service Requests

Service mappings can be established when two criteria are met:

1. AA Server found during discovery

Assuming that an administrator has defined one or more ports for auto attach mode a discovery message is sent out each port defined using LLDP. Element information is forwarded using LLDP TLV extensions defined in section 6.1.

2. I-SID/VLAN bindings are defined locally

Assuming that an administrator has defined one or more I-SID/VLAN assignments (or AAC bindings have been received for processing), an AAC sends the I-SID/VLAN assignment list to the discovered AAS. I-SID/VLAN data is exported using LLDP TLV extensions defined in section 6.2.

5.2.1. Element Inactivity Timeout

An AAC must handle primary AAS loss and this requires maintenance of a server's inactivity timer. If primary AAS advertisements are not received for a pre-determined amount of time, the I-SID/VLAN assignments accepted by the server are considered rejected. I-SID/VLAN assignment data is then defaulted (reverts to the 'pending' state) and the AA agent, which resides on the AAC, removes related settings.

5.3. Server Mapping Request Processing

Each I-SID/VLAN assignment in an AA request received by the AAS is processed individually and can be accepted or rejected. An assignment may be rejected for a number of reasons, such as server resource limitations or, for example, restrictions related only to the source AAC. Rejected assignments are passed back to the originating AAC with a rejected state and, if appropriate, an indication as to why the rejection occurred. Limited state information may be maintained on the server related to rejected I-SID/VLAN assignments.

Each VLAN that is associated with an accepted I-SID/VLAN assignment is instantiated on the AAS bridge if it does not already exist. These VLANs are designated SPBM UNI VLANs on a BEB. The port through which the AA I-SID/VLAN assignment list was received (i.e., the AAS downlink) must be a member of the VLAN(s) in the I-SID/VLAN assignment list that are accepted by the AAS. Port membership is

automatically updated when the UNI service (I-SID/VLAN/port) is created. To ensure that VLAN markings are maintained between switches, traffic on the downlink port MUST be tagged. The AA agent on the serving BEB handles all of these tasks automatically. No administrator intervention is required.

The AAS agent is responsible for tracking which, if any, of these actions are performed so that settings can be cleared when they are no longer needed. This can occur, for example, when configuration changes on an AAC updates the received I-SID/VLAN assignment list when an AAC associated with a downlink port changes or an AAC connection disappears entirely. Specifically, when an SPBM switched UNI-based VLAN and a switched UNI have been created on a downlink port because of an accepted AA I-SID/VLAN assignment (and not because of an explicit administrator port action), then the UNI and associated VLAN SHOULD be deleted when the related I-SID/VLAN assignment is cleared by the AAS.

5.4. Server Mapping Response Processing

Each VLAN that is associated with an AAC I-SID/VLAN assignment must be defined on the client's device. The port associated with the uplink connecting the AAC to the AAS must be a member of the VLAN(s) in the I-SID/VLAN assignment list that are sent to and accepted by the AAS. This allows traffic on these VLANs to pass through the switch into the SPB fabric when required. To ensure that VLAN markings are maintained between devices, traffic on the uplink port MUST be tagged. If a VLAN has not been created before the I-SID/VLAN assignment itself, it is automatically created by the AAC agent when a proposed assignment is accepted. Port tagging and the port VLAN membership update are also performed by the AAC automatically based on assignment acceptance. To ensure consistency, VLANs SHOULD NOT be deleted while they are referenced in any I-SID/VLAN assignments on the device.

5.5. Service Mapping Timeout

A "last updated" timestamp is associated with all active assignments on the AAS. When this value is not updated for a pre-determined amount of time, the I-SID/VLAN assignment is considered obsolete. Obsolete assignment data and related settings are removed by the AAS, subject to the constraints imposed by section 4.3.

The current I-SID/VLAN assignment list is advertised by an AAC at regular intervals (dictated by LLDP operation). During processing of this data, an AAS must handle list updates and delete assignments from previous advertisements that are no longer present. Though

these entries would be processed appropriately when they timeout, the AAS attempts to update the data in real-time and SHOULD initiate deletion immediately upon detection of this condition.

6. Auto Attach LLDP Extensions

The text in this section is not normative. The complete definition of the Auto Attach TLVs is provided in the IEEE 802.1Qcj [AA] Amendment of the IEEE 802.1Q standard.

The Auto Attach TLVs are implemented as extensions to the LLDP standard, using its flexible extension mechanism. They SHOULD be implemented as vendor-specific TLVs using TLV type 127 as described in the 802.1AB (LLDP) standard. TLVs supporting the exchange of AA element data and I-SID/VLAN assignment data have been defined below.

6.1. AA Element TLV

The Element TLV is used by an AA device to announce its capabilities to its LLDP peer on a given interface. Use of the Auto Attach functionality is encoded in to the 802.1AB LLDP Custom Element TLV as follows:

AA Element TLV

Type: 127 (7 bits)
Length: 49 octets (9 bits)
OUI: 3 octets
Subtype: 11 (1 octet)
HMAC-SHA256 Digest: 32 oct
Element Type: 6 bits
State: 6 bits
Mgmt VLAN: 12 bits
System ID: 10 octets

Subtype = 11 for AA Element TLV

HMAC-256 Digest:

The Element TLV data integrity and source validation is supported through the use of the HMAC-SHA256 message authentication algorithm. The HMAC-SHA256 generated digest size is 32 octets and the Element TLV includes a field to support the digest exchange between source and destination parties. Symmetric private keys are used for digest generation.

The HMAC-SHA256 data digest computation starts at [0-based] byte 38 of the TLV. The digest is then placed in the HMAC-SHA256 Digest field in the TLV prior to transmission. Upon receipt, the digest is again computed and the resulting digest is compared against the received digest. If the received digest is the same as the newly computed digest, the TLV is considered valid and processing can commence. If the comparison fails, the TLV is discarded and processing is terminated.

Element Type:

The element type identifies the capability of the advertising AA node. The AA Server describes an AAS capable device that can map incoming VLAN to I-SID and announce I-SID connectivity to the SPB network. AA Clients may operate in either tagged or untagged modes. If an AA client announces untagged, then the entire port MUST be mapped to the I-SID on the BEB.

The AA Element TLV can only exist once in a LLDPDU. It is included in all LLDPDUs when the Auto Attach service is enabled and when the per-port transmission flags associated with this TLV, as required by the 802.1AB standard, are enabled.

A number of AA Element Type values, including the AA Server and several AA Client element types, are currently defined. The list of supported element types will expand as additional devices incorporate AA signaling.

Currently supported Auto Attach Element Type values:

AA Element Type - Other (1)

AA Server (2)

AA Server No Authentication (3)

AA Client - Wireless Access Point Type 1 (4) [wireless clients get direct network attachment]

AA Client - Wireless Access Point Type 2 (5) [wireless clients get tunneled to a controller]

AA Client - Switch (6)

AA Client - Router (7)

AA Client - IP Phone (8)

AA Client - IP Camera (9)

AA Client - IP Video (10)

AA Client - Security Device (11) [FW, IPS/IDS, etc.]

AA Client - Virtual Switch (12)

AA Client - Server/Endpoint (13)

AA Client - SDN Controller (14)

AA Client - SPB-over-IP Network Device (15)

State:

The AA Element TLV State field settings indicate AA Client link tagging requirements in AA Client-sourced frames and current provisioning mode information (bits are numbered left to right):

Link VLAN Tagging Requirements (bit 1)

0 - All traffic tagged on link

1 - Tagged and untagged traffic on link

Automatic Provisioning Mode (bits 2/3)

0 - Automatic provisioning disabled

- 1 - SPB provisioning
- 2 - VLAN provisioning

System ID: conveys information that the TLV recipient can use to enforce connectivity restrictions. It includes System MAC Address, connection type and identifiers. Detailed specification of the System ID sub-fields is TBD.

6.2. I-SID/VLAN Assignment TLV

The AA I-SID/VLAN Assignment TLV is used by the AAC to announce I-SID/VLAN assignments that it would like supported by a directly connected AAS. It is also used by the AAS to announce that that I-SID/VLAN bindings processed by the AAS are active or rejected.

The AA I-SID/VLAN Assignment TLV can only exist once in a LLDPDU. It is only included in a LLDPDU when complementary AA element (i.e., AA server/ client) devices are directly connected. Data integrity and source validation is supported through the use of the HMAC-SHA256 message authentication algorithm. The HMAC-SHA256 generated digest size is 32 octets and the AA I-SID/VLAN Assignment TLV includes a field to support the digest exchange between source and destination parties.

Per-port TLV transmission flags must be enabled on the communicating devices as well. The AA Element TLV must also be present in the LLDPDU for the AA I-SID/VLAN Assignment TLV to be processed. The TLV cannot exceed the LLDP 512 byte TLV size limit, which implies a maximum of 94 I-SID/VLAN assignments in a LLDPDU

The format of the TLV is as follows:

Service Assignment TLV

```

+-----+
|  Type:   127 (7 bits)  |
+-----+
| Length: 41-506 octets (9 bits) |
+-----+
|  OUI:                3 octets  |
+-----+
```

Subtype:	12 (1 octet)
+-----+	
HMAC-SHA256 Digest:	32 octets
+-----+	
Assignment Status:	4 bits
+-----+	
VLAN:	12 bits
+-----+	
I-SID:	3 octets
+-----+	

The HMAC-SHA256 digest is computed for the series (1-94) of I-SID/VLAN assignments (i.e. data for the digest computation starts at [0-based] byte 38 of the TLV). The digest is then placed in the HMAC-SHA256 Digest field in the TLV prior to transmission. Upon receipt, the digest is again computed for the series (1-94) of I-SID/VLAN assignments in the received TLV and the resulting digest is compared against the received digest. If the received digest is the same as the newly computed digest, the TLV is considered valid and processing can commence. If the comparison fails, the TLV is discarded and processing is terminated. Additionally the value for the I-SID in the incoming LLDP exchanges SHOULD trigger an IS-IS SPBM announcement using normal IEEE 802.1aq mechanisms if not already being announced by the BEB.

The assignment status data is returned by the AA Server for each pending I-SID/VLAN assignment request. Assignment rejections may include information to indicate the reason for the rejection. A limited number of detailed rejection error codes will initially be supported.

Assignment Pending(1)

Assignment Accepted(2)

Rejection: Generic(3)

Rejection: AA resources unavailable(4) - the resources that are required for the Auto Attach agent to support additional I-SID/VLAN assignments are currently exhausted. The maximum number of assignments that can be supported has been reached.

Rejection: Duplicate(5)

Rejection: VLAN invalid(6) - the specified VLAN can't be used to create a switched UNI at this time. The VLAN already exists and is either inactive or has an incorrect type for this application.

Rejection: VLAN unknown(7)

Rejection: VLAN resources unavailable(8) - the maximum number of VLANs that can be supported by the device has been reached.

Rejection: Application interaction issue(9) - a failure has been detected during AA interactions with the VLAN and/or the SPBM applications. The VLAN operations to create the required SPBM switched UNI VLAN or enable port tagging may have failed or the SPBM operation to create the switched UNI may have failed

Please note that the status field is only valid when generated by an AA Server. Any Assignment TLVs which are received by an AA server are assumed to be requests. It is recommended that the status field of assignments generated by AA clients be set to 0 or 1.

VLAN: A VLAN value of 0 may indicate that AAC traffic is untagged.

7. Security Considerations

It is important to provide an option to ensure that the aforementioned Auto Attach communication is secure in terms of data integrity (i.e., the data has not been altered in transit) and authenticity (i.e., the data source is valid).

If communication is occurring between non-secure systems, the HMAC-SHA256 Digest data should always be zero and the digest data, regardless of the value, is ignored. A misconfiguration can occur with one system operating in secure mode and the other operating in non-secure mode. In this scenario, the Element TLV or the I-SID/VLAN Assignment TLV will always be discarded prior to processing by the system operating in secure mode.

These security requirements are satisfied by using an optional keyed-hash message authentication code (HMAC) to protect the AAC/AAS Element Discovery and I-SID/VLAN assignment exchanges. This type of message authentication allows communicating parties to verify that the contents of the message have not been altered and that the source is authentic. Use of this mechanism is optional and is controlled through a user-configurable attribute.

7.1. TLV Security Considerations

A HMAC-SHA256 digest is computed for Element TLV or for the series of I-SID/VLAN assignments, where the digest computation starts [0 based] at byte 38 of the TLV. The resulting digest is then placed in the TLV prior to sending. Where upon receipt of the digest, the contents are again computed in the same manner and the digests are compared, if the comparison fails then the TLV is discarded, otherwise if both digests are the same the TLV is considered valid and processed appropriately.

8. IANA Considerations

This memo includes no request to IANA.

Note: the section will be removed during conversion into an RFC by the RFC Editor.

9. Further and Related Work

The standard extensions to the IEEE 802.1AB (LLDP) [LLDP] protocol are developed by the IEEE 802.1 Working Group. The relevant project is IEEE 802.1Qcj [AA] for 'Automatic Attachment to Provider Backbone Bridges (PBB) services'.

Current open issues:

- Define whether an AA Proxy needs to be made part of the architecture and if yes, define its role
- Further details on the two AA TLVs fields
- Define semantics of I-SID value of 0
- Alignment with the (normative) definitions in IEEE 802.1Qcj (as they progress)

The Auto Attachment YANG data model is developed as [TBA1].

The Auto Attachment Controller logic (defining the VLAN/ISID mapping logic) is developed as [TBA2].

10. Log of Changes

(this section to be removed when RFC is published)

10.1. Changes between draft-unbehagen-lldp-spb-01 and 02

- Added definition of the Auto Attach framework and principal components

- Updated Figure 1
- Updated the AA Element TLV fields
- Added several Element Type values
- Added Rejection error codes in I-SID/VLAN Assignment TLV
- Defined references to the IEEE 802.1Qcj approved project
- Various editorial improvements and corrections

11. Acknowledgments

We would like to thank the following people (in no particular order) for their contributions:

Zenon Kuc

Cristian Mema

Roger Lapuh

Craig Griffin

Chris Buerger

Keith Krajewski

This document was prepared using 2-Word-v2.0.template.dot.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [LLDP] IEEE STD 802.1AB, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", 2005.
- [PBB] IEEE STD 802.1ah, "IEEE Standard for Local and Metropolitan Area Networks / Virtual Bridged Local Area Networks / Amendment 7: Provider Backbone Bridges", 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6329] Fedyk, D., Ashwood-Smith, P., Allan, D., Bragg, A. and P. Unbehagen, "IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging", RFC 6329, April 2012.

- [SPB] IEEE STD 802.1aq, "IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment 20: Shortest Path Bridging", 2012.

12.2. Informative References

- [AA] IEEE 802.1Qcj, "Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Automatic Attachment to Provider Backbone Bridging (PBB) services"

Authors' Addresses

Paul Unbehagen Jr, editor
Avaya
1300 W. 120th Avenue
Westminster, CO 80234
US
Email: unbehagen@avaya.com

Dan Romascanu
Avaya
Azrieli Center Holon
26, HaRokhmim Str., Bldg. D
Holon, 5885849
Israel
Phone: +972-3-645-8414
Email: dromasca@avaya.com

John Seligson
Avaya
4655 Great America Parkway
Santa Clara, CA 95054
US
Email: jseligso@avaya.com

Carl Keene
Avaya
600 Technology Park Dr
Boston, MA 01821
US
Email: ckeene@avaya.com

Nigel Bragg
Ciena
Ciena House
43-51 Worship Street
London, EC2A 2DX
Uk
Email: nbragg@ciena.com

Ludovic Beliveau
Wind River
Email: Ludovic.Beliveau@windriver.com