opsawg Z. Li
Internet-Draft R. Gu
Intended status: Standards Track China Mobile
Expires: June 19, 2019 J. Dong
Huawei Technologies
December 16, 2018

Export BGP community information in IP Flow Information Export (IPFIX)
draft-ietf-opsawg-ipfix-bgp-community-12

Abstract

By introducing new Information Elements (IEs), this draft extends the existing BGP-related IEs to enable IP Flow Information Export (IPFIX) to export BGP community information, including BGP standard communities defined in RFC1997, BGP extended communities defined in RFC4360, and BGP large communities defined in RFC8092. Network traffic information can then be accumulated and analyzed at the BGP community granularity, which represents the traffic of different kinds of customers, services, or geographical regions according to the network operator's BGP community planning. Network traffic information at the BGP community granularity is useful for network traffic analysis and engineering.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 5
3. BGP Community-based Traffic Collection 5
4. IEs for BGP Standard Community 6
5. IEs for BGP Extended Community 7
6. IEs for BGP Large Community 7
7. Operational Considerations 8
8. Security Considerations 9
9. IANA Considerations 9
10. Acknowledgements 11
11. References 12
  11.1. Normative References 12
  11.2. Informative References 12
Appendix A. Encoding Example 14
  A.1. Template Record 14
  A.2. Data Set 15
Authors' Addresses 16

1. Introduction

IP Flow Information Export (IPFIX) [RFC7011] provides network administrators with traffic flow information using the Information Elements (IEs) defined in [IANA-IPFIX] registries. Based on the traffic flow information, network administrators know the amount and direction of the traffic in their network, and can then optimize their network when needed. For example, the collected information could be used for traffic monitoring, and could optionally be used for traffic optimization according to operator's policy.

[IANA-IPFIX] has already defined the following IEs for traffic flow information exporting in different granularities: sourceIPv4Address, sourceIPv4Prefix, destinationIPv4Address, destinationIPv4Prefix, bgpSourceAsNumber, bgpDestinationAsNumber, bgpNextHopIPv4Address, etc. In some circumstances, however, especially when traffic engineering and optimization are executed in Tier 1 or Tier 2 operators' backbone networks, traffic flow information based on these

IEs may not be completely suitable or sufficient. For example, flow information based on IP address or IP prefix may provide much too fine granularity for a large network. On the contrary, flow information based on AS number may be too coarse.

BGP community is a BGP path attribute that includes standard communities [RFC1997], extended communities [RFC4360], and large communities [RFC8092]. The BGP community attribute has a variety of use cases, one of which is to use BGP community with planned specific values to represent groups of customers, services, and geographical or topological regions, as used by operators in their networks. Detailed examples can be found in [RFC4384], [RFC8195] and Section 3 of this document. To understand the traffic generated by different kinds of customers, from different geographical or topological regions, by different kinds of customers in different regions, we need the corresponding community information related to the traffic flow information exported by IPFIX. Network traffic statistics at the BGP community granularity are useful not only for the traffic analyzing, but also can then be used by other applications, such as traffic optimization applications located in an IPFIX Collector, SDN controller or PCE. [Community-TE] also states that analyzing network traffic information at the BGP community granularity is preferred for inbound traffic engineering. However, [IANA-IPFIX] lacks IEs defined for the BGP community attribute.

Flow information based on BGP community may be collected by an IPFIX Mediator defined in [RFC6183]. IPFIX Mediator is responsible for the correlation between flow information and BGP community. However, no IEs are defined in [RFC6183] for exporting BGP community information in IPFIX. Furthermore, to correlate the BGP community with the flow information, the IPFIX Mediator needs to learn BGP routes and perform lookups in the BGP routing table to get the matching entry for a specific flow. Neither BGP route learning nor routing table lookup are trivial for an IPFIX Mediator. The IPFIX Mediator is mainly introduced to reduce the performance requirement for the Exporter [RFC5982]. In fact, to obtain the information for the already defined BGP related IEs, such as bgpSourceAsNumber, bgpDestinationAsNumber, and bgpNextHopIPv4Address, etc, the Exporter has to hold the up-to-date BGP routing table and perform lookups in the table. The Exporter can obtain the BGP community information in the same procedure, thus the additional load added by exporting BGP community information is minimal if the Exporter is already exporting the existing BGP-related IEs. It is RECOMMENDED that the BGP community information be exported by the Exporter directly using IPFIX.

Through running BGP [RFC4271] or BMP [RFC7854] and performing lookups in the BGP routing table to correlate the matching entry for a specific flow, IPFIX Collectors and other applications, such as SDN controller or PCE, can determine the network traffic at the BGP community granularity. However, neither running BGP or BMP protocol nor routing table lookup are trivial for the IPFIX Collectors and other applications. Moreover, correlation between IPFIX flow information and the BGP RIB on the Exporter (such as a router) is more accurate, compared to the correlation on a Collector, since the BGP routing table may be updated when the IPFIX Collectors and other applications receive the IPFIX flow information. And as stated above, the Exporter can obtain the BGP community information during the same procedure when it obtains other BGP related information. So exporting the BGP community information directly by the Exporter to the Collector is both efficient and accurate. If the IPFIX Collectors and other applications only want to determine the network traffic at the BGP community granularity, they do not need to run the full BGP or BMP protocols when the BGP community information can be obtained by IPFIX. However, the BMP protocol has its own application scenario, and the mechanism introduced in this document is not meant to replace it.

By introducing new IEs, this draft extends the existing BGP-related IEs to enable IPFIX [RFC7011] to export BGP community information, including the BGP standard communities [RFC1997], BGP extended communities [RFC4360], and BGP large communities [RFC8092]. Flow information, including packetDeltaCount, octetDeltaCount [RFC7012], etc., can then be accumulated and analyzed by the Collector or other applications, such as an SDN controller or PCE [RFC4655], at the BGP community granularity, which is useful for measuring the traffic generated by different kinds of customers, from different geographical or topological regions according to the operator’s BGP community plan, and can then be used by the traffic engineering or traffic optimization applications, especially in the backbone network.

The IEs introduced in this document are applicable for both IPv4 and IPv6 traffic. Both the Exporter and the IPFIX Mediator can use these IEs to export BGP community information in IPFIX. When needed, the IPFIX Mediator or Collector can use these IEs to report BGP community related traffic flow information it gets either from Exporters or through local correlation to other IPFIX devices.

As stated above, the method introduced in this document is not the definitive and the only one to obtain BGP community information related to a specific traffic flow, but a possible, efficient and accurate one.

No new BGP community attributes are defined in this document.

Note that this document does not update the IPFIX specification [RFC7011] and the Information Model [RFC7012]. Rather, IANA's IPFIX registry [IANA-IPFIX] contains the current complete Information Element reference, per Section 1 of [RFC7012].

Please refer to [IANA-IPFIX] for the complete list of BGP-related IEs.

Please refer to Appendix A of this document for the encoding example and Section 3 for a detailed use case.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

IPFIX-specific terminology used in this document is defined in Section 2 of [RFC7011] and Section 2 of [RFC6183].

BGP standard community: The BGP Communities attribute defined in [RFC1997]. In order to distinguish it from BGP extended communities [RFC4360], and large communities [RFC8092], BGP Communities attribute is called BGP standard community in this document.

3. BGP Community-based Traffic Collection

[RFC4384] introduces the mechanism of using BGP standard community and extended community to collect the geographical and topological related information in the BGP routing system. [RFC8195] gives some examples of the application of BGP large communities to represent the geographical regions. Since the network traffic at the BGP community granularity represents the traffic generated by different kinds of customers, from different geographical regions according to the network operator's BGP community plan, it is useful for network operators to analyze and optimize the network traffic among different customers and regions. This section gives a use case in which the network operator uses the BGP community-based traffic information to adjust the network paths for different traffic flows.

Consider the following scenario, AS C provides a transit connection between ASes A and B. By tagging with different BGP communities, the routes of AS A and B are categorized into several groups respectively in the operator's plan. For example, communities A:X and A:Y are used for the routes originated from different geographical regions in AS A, and communities B:M and B:N are used for the routes representing the different kinds of customers in AS B, such as B:M is for the mobile customers and B:N is for the fixed line customers. By default, all traffic originating from AS A and destined to AS B (we call it traffic A-B) goes through path C1-C2-C3 (call it Path-1) in AS C. When the link between C1 and C2 is congested, we cannot simply steer all the traffic A-B from Path-1 to Path C1-C4-C3 (call it Path-2), because it will cause congestion in Path-2.

```
                                +----------+
                                | PCE/SDN  |
                        +-------|Controller|-------+
                        |       +----------+       |
                        |                          |
                        |           AS C           |
                |       |       +----------+       |       |
                |       |   +---|Router C2 |---+   |       |
                |       |   |   +----------+   |   |       |
       AS A     |       |   |100             50|   |       |     AS B
    +--------+  |  +---------+                +---------+  |  +--------+
    |Router A|--|--|Router C1|                |Router C3|--|--|Router B|
    +--------+  |  +---------+                +---------+  |  +--------+
    Community:  |           |100            100|           |  Community:
        A:X     |           |   +----------+   |           |     B:M
        A:Y     |           +---|Router C4 |---+           |     B:N
                                +----------+
```

Figure 1: BGP Community based Traffic Collection

If the PCE/SDN controller in AS C can obtain the network traffic information at the BGP community granularity, it can steer some traffic related to some BGP communities (when we consider only the source or destination of the traffic), or some BGP community pairs (when we consider both the source and the destination of the traffic) from Path-1 to Path-2 according to the utilization of different paths. For instance, steer the traffic generated by community A:X from Path-1 to Path-2 by deploying a route policy at Router C1, or steer the traffic from community A:Y to community B:M from Path-1 to Path-2. Using the IEs defined in this document, IPFIX can export the BGP community information related to a specific traffic flow together with other flow information. The traffic information can then be accumulated at the BGP community granularity and used by the PCE/SDN controller to steer the appropriate traffic from Path-1 to Path-2.

4. IEs for BGP Standard Community

[RFC1997] defines the BGP Communities attribute, called BGP Standard Community in this document, which describes a group of routes sharing some common properties. BGP Standard Community is treated as 32 bit value as stated in [RFC1997].

In order to export BGP standard community information along with other flow information defined by IPFIX, three new IEs are introduced. One is bgpCommunity, which is used to identify that the value in this IE is a BGP standard community. The other two are bgpSourceCommunityList and bgpDestinationCommunityList, which are both basicList [RFC6313] of bgpCommunity, and are used to export BGP standard community information corresponding to a specific flow's source and destination IP address respectively.

The detailed information of the three new IEs are shown in Section 9, IANA Considerations.

5. IEs for BGP Extended Community

[RFC4360] defines the BGP Extended Communities attribute, which provides a mechanism for labeling the information carried in BGP. Each Extended Community is encoded as an 8-octet quantity with the format defined in [RFC4360].

In order to export BGP Extended Community information together with other flow information by IPFIX, three new IEs are introduced. The first one is bgpExtendedCommunity, which is used to identify that the value in this IE is a BGP Extended Community. The other two are bgpSourceExtendedCommunityList and bgpDestinationExtendedCommunityList, which are both basicList [RFC6313] of bgpExtendedCommunity, and are used to export the BGP Extended Community information corresponding to a specific flow's source and destination IP address respectively.

The detailed information of the three new IEs are shown in Section 9, IANA Considerations.

6. IEs for BGP Large Community

[RFC8092] defines the BGP Large Communities attribute, which is suitable for use with all Autonomous System Numbers (ASNs) including four-octet ASNs. Each BGP Large Community is encoded as a 12-octet quantity with the format defined in [RFC8092].

In order to export BGP Large Community information together with other flow information by IPFIX, three new IEs are introduced. The first one is bgpLargeCommunity, which is used to identify that the value in this IE is a BGP Large Community. The other two are bgpSourceLargeCommunityList and bgpDestinationLargeCommunityList, which are both basicList [RFC6313] of bgpLargeCommunity, and are used to export the BGP Large Community information corresponding to a specific flow's source and destination IP address respectively.

The detailed information of the three new IEs are shown in Section 9, IANA Considerations.

7. Operational Considerations

The maximum length of an IPFIX message is 65535 bytes as per [RFC7011] , and the maximum length of a normal BGP message is 4096 bytes as per [RFC4271]. Since BGP communities, including standard, extended, and large communities, are BGP path attributes carried in BGP Update messages, the total length of these attributes can not exceed the length of a BGP message, i.e. 4096 bytes. So one IPFIX message with a maximum length of 65535 bytes has enough space to fit all the communities related to a specific flow, relating to both the source and destination IP addresses.

[I-D.ietf-idr-bgp-extended-messages] extends the maximum size of a BGP Update message to 65535 bytes. In that case, the BGP community information related to a specific flow could theoretically exceed the length of one IPFIX message. However, according to information regarding actual networks in the field, the number of BGP communities in one BGP route is usually no more than ten. Nevertheless, BGP speakers that support the extended message SHOULD only convey as many communities as possible without exceeding the 65536-byte limit of an IPFIX message. The Collector which receives an IPFIX message with maximum length and BGP communities contained in its data set SHOULD generate a warning or log message to indicate that the BGP communities may be truncated due to limited message space. In this case, it is recommended to configure the export policy of BGP communities to limit the BGP communities by including or excluding specific communities.

If needed, the IPFIX message length could be extended from 16 bits to 32 bits to solve this problem completely. The details of increasing the IPFIX message length is out of scope of this document.

To align with the size of the BGP extended community and large community attributes, the size of IE bgpExtendedCommunity and bgpLargeCommunity is 8 octets and 12 octets respectively. In the event that the bgpExtendedCommunity or bgpLargeCommunity IE is not of its expected size, the IPFIX Collector SHOULD ignore it. This is intended to protect implementations using BGP logic from calling their parsing routines with invalid lengths.

For the proper processing of the Exporter when it receives the template requesting to report the BGP community information (refer to

Appendix A for an example), the Exporter SHOULD obtain the corresponding BGP community information through BGP lookup using the corresponding source or destination IP address of the specific traffic flow. When exporting the IPFIX information to the Collector, the Exporter SHOULD include the corresponding BGP communities in the IPFIX message.

8. Security Considerations

This document defines new IEs for IPFIX. The same security considerations as for the IPFIX Protocol Specification [RFC7011] and Information Model [RFC7012] apply.

Systems processing BGP community information collected by IPFIX collectors need to be aware of the use of communities as an attack vector [Weaponizing-BGP], and only include BGP community information in their decisions where they are confident of its validity. Thus we can not assume that all BGP community information collected by IPFIX collectors is credible and accurate. It is RECOMMENDED to use only the IPFIX collected BGP community information that the processing system can trust, for example the BGP communities generated by the consecutive neighboring ASs within the same trust domain as the processing system (for instance, the consecutive neighboring ASs and the processing system are operated by one carrier).

[RFC7011] says that the storage of the information collected by IPFIX must be protected and confined its visibility to authorized users via technical as well as policy means to ensure the privacy of the information collected. [RFC7011] also provides mechanisms to ensure the confidentiality and integrity of IPFIX data transferred from an Exporting Process to a Collecting Proces. The mechanism to authenticate IPFIX Collecting and Exporting Processes is provided in [RFC7011], too. If sensitive information is contained in the community information, the above recommendations and mechanisms are recommended to be used. No additional privacy risks are introduced by this standard.

9. IANA Considerations

This draft specifies the following IPFIX IEs to export BGP community information along with other flow information.

The Element IDs for these IEs are requested to be assigned by IANA. The following table is for IANA’s use to place in each field in the registry.

ElementID	Name	Data Type	Data Type Semantics

TBA1	bgpCommunity	unsigned32	identifier
TBA2	bgpSourceCommunityList	basicList	list
TBA3	bgpDestinationCommunityList	basicList	list
TBA4	bgpExtendedCommunity	octetArray	default
TBA5	bgpSourceExtended CommunityList	basicList	list
TBA6	bgpDestinationExtended CommunityList	basicList	list
TBA7	bgpLargeCommunity	octetArray	default
TBA8	bgpSourceLargeCommunityList	basicList	list
TBA9	bgpDestinationLarge CommunityList	basicList	list

ElementID	Description	Units
TBA1	BGP community as defined in [RFC1997]	
TBA2	basicList of zero or more bgpCommunity IEs, containing the BGP communities corresponding with source IP address of a specific flow	
TBA3	basicList of zero or more bgpCommunity IEs, containing the BGP communities corresponding with destination IP address of a specific flow	
TBA4	BGP Extended Community as defined in [RFC4360] The size of this IE MUST be 8 octets	
TBA5	basicList of zero or more bgpExtendedCommunity IEs, containing the BGP Extended Communities corresponding with source IP address of a specific flow	
TBA6	basicList of zero or more bgpExtendedCommunity IEs, containing the BGP Extended Communities corresponding with destination IP address of a specific flow	

TBA7	BGP Large Community as defined in [RFC8092] The size of this IE MUST be 12 octets.	
TBA8	basicList of zero or more bgpLargeCommunity IEs, containing the BGP Large Communities corresponding with source IP address of a specific flow	
TBA9	basicList of zero or more bgpLargeCommunity IEs, containing the BGP Large Communities corresponding with destination IP address of a specific flow	

ElementID	Range	References	Requester	Revision	date
TBA1		RFC1997	this draft	0	
TBA2		RFC6313,RFC1997	this draft	0	
TBA3		RFC6313,RFC1997	this draft	0	
TBA4		RFC4360	this draft	0	
TBA5		RFC6313,RFC4360	this draft	0	
TBA6		RFC6313,RFC4360	this draft	0	
TBA7		RFC8092	this draft	0	
TBA8		RFC6313,RFC8092	this draft	0	
TBA9		RFC6313,RFC8092	this draft	0	

Figure 2: IANA Considerations

10. Acknowledgements

The authors would like to thank Benoit Claise and Paul Aitken for their comments and suggestions to promote this document. We also thank Tianran Zhou, Warren Kumari, Jeffrey Haas, Ignas Bagdonas, Stewart Bryant, Paolo Lucente, Job Snijders, Jared Mauch, Rudiger Volk, and Andrew Malis for their discussion, comments, and suggestions to improve this document..

11. References

11.1. Normative References

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

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, DOI 10.17487/RFC6313, July 2011, <https://www.rfc-editor.org/info/rfc6313>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[Community-TE] Shao, W., Devienne, F., Iannone, L., and JL. Rougier, "On the use of BGP communities for fine-grained inbound traffic engineering", Computer Science 27392(1):476-487, November 2015.

[I-D.ietf-idr-bgp-extended-messages] Bush, R., Patel, K., and D. Ward, "Extended Message support for BGP", draft-ietf-idr-bgp-extended-messages-27 (work in progress), December 2018.

[IANA-IPFIX] "IP Flow Information Export (IPFIX) Entities", <http://www.iana.org/assignments/ipfix/>.

[RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <https://www.rfc-editor.org/info/rfc1997>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4384] Meyer, D., "BGP Communities for Data Collection", BCP 114, RFC 4384, DOI 10.17487/RFC4384, February 2006, <https://www.rfc-editor.org/info/rfc4384>.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC5982] Kobayashi, A., Ed. and B. Claise, Ed., "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, DOI 10.17487/RFC5982, August 2010, <https://www.rfc-editor.org/info/rfc5982>.

[RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, DOI 10.17487/RFC6183, April 2011, <https://www.rfc-editor.org/info/rfc6183>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/info/rfc7012>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

[RFC8092] Heitz, J., Ed., Snijders, J., Ed., Patel, K., Bagdonas, I., and N. Hilliard, "BGP Large Communities Attribute", RFC 8092, DOI 10.17487/RFC8092, February 2017, <https://www.rfc-editor.org/info/rfc8092>.

[RFC8195] Snijders, J., Heasley, J., and M. Schmidt, "Use of BGP Large Communities", RFC 8195, DOI 10.17487/RFC8195, June 2017, <https://www.rfc-editor.org/info/rfc8195>.

[Weaponizing-BGP]
Streibelt, F., Lichtblau, F., Beverly, R., and et al., "Weaponizing BGP Using Communities", November 2018, <https://datatracker.ietf.org/meeting/103/materials/slides-103-grow-bgp-communities-spread-their-wings-01>.

Appendix A. Encoding Example

In this section, we provide an example to show the encoding format for the new introduced IEs.

Flow information, including BGP communities, is shown in the following table. In this example, all the fields are reported by IPFIX.

Source IP	Destination IP	BGP community corresponding with Source IP	BGP community corresponding with Destination IP
1.1.1.1	2.2.2.2	1:1001,1:1002,8:1001	2:1002,8:1001
3.3.3.3	4.4.4.4	3:1001,3:1002,8:1001	4:1001,8:1001

Figure 3: Flow information including BGP communities

A.1. Template Record

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SET ID = 2           |         Length = 24           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Template ID = 256        |        Field Count = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    SourceIPv4Address = 8    |        Field length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| DestinationIPv4Address = 12 |        Field length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| bgpSourceCommunityList= TBA2|      Field length = 0xFFFF    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| bgpDestinationCommunityList |      Field length = 0xFFFF    |
| |         = TBA3              |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Template Record Encoding Format

In this example, the Template ID is 256, which will be used in the Data Record. The field length for bgpSourceCommunityList and bgpDestinationCommunityList is 0xFFFF, which means the length of this IE is variable, and the actual length of this IE is indicated by the list length field in the basic list format as per [RFC6313].

A.2. Data Set

The data set is represented as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SET ID = 256         |           Length = 92         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   SourceIPv4Address = 1.1.1.1                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                DestinationIPv4Address = 2.2.2.2               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |        List length = 17       |semantic=allof |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      bgpCommunity = TBA1      |          Field Len = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        BGP Source Community Value 1 = 1:1001                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        BGP Source Community Value 2 = 1:1002                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        BGP Source Community Value 3 = 8:1001                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     255       |        List length = 13       |semantic =allof|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      bgpCommunity = TBA1      |          Field Len = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         BGP Destination Community Value 1 = 2:1002            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         BGP Destination Community Value 2 = 8:1001            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   SourceIPv4Address = 3.3.3.3                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                DestinationIPv4Address = 4.4.4.4               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     255       |        List length = 17       |semantic =allof|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      bgpCommunity = TBA1      |          Field Len = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        BGP Source Community Value 1  = 3:1001                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        BGP Source Community Value 2  = 3:1002                 |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        BGP Source Community Value 3  = 8:1001                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     255       |         List length = 13       |semantic =allof|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      bgpCommunity = TBA1      |          Field Len = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         BGP Destination Community Value 1 = 4:1001            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         BGP Destination Community Value 2 = 8:1001            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Data Set Encoding Format

Authors' Addresses

Zhenqiang Li
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: li_zhenqiang@hotmail.com

Rong Gu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: gurong_cmcc@outlook.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2018

E. Lear
Cisco Systems
R. Droms
Google
D. Romascanu

June 15, 2018

Manufacturer Usage Description Specification
draft-ietf-opsawg-mud-25

Abstract

This memo specifies a component-based architecture for manufacturer usage descriptions (MUD). The goal of MUD is to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function. The initial focus is on access control. Later work can delve into other aspects.

This memo specifies two YANG modules, IPv4 and IPv6 DHCP options, an LLDP TLV, a URL, an X.509 certificate extension and a means to sign and verify the descriptions.

Status of This Memo

Copyright Notice

Table of Contents


```
   1.  Introduction  . . . . . . . . . . . . . . . . . . . . . . . .   3
     1.1.  What MUD Doesn't Do . . . . . . . . . . . . . . . . . . .   5
     1.2.  A Simple Example  . . . . . . . . . . . . . . . . . . . .   5
     1.3.  Terminology . . . . . . . . . . . . . . . . . . . . . . .   5
     1.4.  Determining Intended Use  . . . . . . . . . . . . . . . .   6
     1.5.  Finding A Policy: The MUD URL . . . . . . . . . . . . . .   6
     1.6.  Processing of the MUD URL . . . . . . . . . . . . . . . .   7
     1.7.  Types of Policies . . . . . . . . . . . . . . . . . . . .   8
     1.8.  The Manufacturer Usage Description Architecture . . . . .  10
     1.9.  Order of operations . . . . . . . . . . . . . . . . . . .  11
   2.  The MUD Model and Semantic Meaning  . . . . . . . . . . . . .  12
     2.1.  The IETF-MUD YANG Module  . . . . . . . . . . . . . . . .  13
   3.  MUD model definitions for the root mud container  . . . . . .  14
     3.1.  mud-version . . . . . . . . . . . . . . . . . . . . . . .  14
     3.2.  mud-url . . . . . . . . . . . . . . . . . . . . . . . . .  15
     3.3.  to-device-policy and from-device-policy containers  . . .  15
     3.4.  last-update . . . . . . . . . . . . . . . . . . . . . . .  15
     3.5.  cache-validity  . . . . . . . . . . . . . . . . . . . . .  15
     3.6.  is-supported  . . . . . . . . . . . . . . . . . . . . . .  15
     3.7.  systeminfo  . . . . . . . . . . . . . . . . . . . . . . .  15
     3.8.  mfg-name, software-rev, model-name firmware-rev . . . . .  16
     3.9.  extensions  . . . . . . . . . . . . . . . . . . . . . . .  16
   4.  Augmentation to the ACL Model . . . . . . . . . . . . . . . .  16
     4.1.  manufacturer  . . . . . . . . . . . . . . . . . . . . . .  16
     4.2.  same-manufacturer . . . . . . . . . . . . . . . . . . . .  16
     4.3.  documentation . . . . . . . . . . . . . . . . . . . . . .  17
     4.4.  model . . . . . . . . . . . . . . . . . . . . . . . . . .  17
     4.5.  local-networks  . . . . . . . . . . . . . . . . . . . . .  17
     4.6.  controller  . . . . . . . . . . . . . . . . . . . . . . .  17
     4.7.  my-controller . . . . . . . . . . . . . . . . . . . . . .  18
     4.8.  direction-initiated . . . . . . . . . . . . . . . . . . .  18
   5.  Processing of the MUD file  . . . . . . . . . . . . . . . . .  18
   6.  What does a MUD URL look like?  . . . . . . . . . . . . . . .  18
   7.  The MUD YANG Model  . . . . . . . . . . . . . . . . . . . . .  19
   8.  The Domain Name Extension to the ACL Model  . . . . . . . . .  25
     8.1.  src-dnsname . . . . . . . . . . . . . . . . . . . . . . .  26
     8.2.  dst-dnsname . . . . . . . . . . . . . . . . . . . . . . .  26
     8.3.  The ietf-acldns Model . . . . . . . . . . . . . . . . . .  26
   9.  MUD File Example  . . . . . . . . . . . . . . . . . . . . . .  28
```

10. The MUD URL DHCP Option 30
10.1. Client Behavior 31
10.2. Server Behavior 31
10.3. Relay Requirements 32
11. The Manufacturer Usage Description (MUD) URL X.509 Extension 32
12. The Manufacturer Usage Description LLDP extension 34
13. Creating and Processing of Signed MUD Files 35
13.1. Creating a MUD file signature 36
13.2. Verifying a MUD file signature 36
14. Extensibility 37
15. Deployment Considerations 37
16. Security Considerations 38
17. IANA Considerations 40
17.1. YANG Module Registrations 41
17.2. DHCPv4 and DHCPv6 Options 41
17.3. PKIX Extensions 41
17.4. MIME Media-type Registration for MUD files 42
17.5. LLDP IANA TLV Subtype Registry 42
17.6. The MUD Well Known Universal Resource Name (URNs) . . . 43
17.7. Extensions Registry 43
18. Acknowledgments 43
19. References 44
19.1. Normative References 44
19.2. Informative References 47
Appendix A. Changes from Earlier Versions 49
Appendix B. Default MUD nodes 52
Appendix C. A Sample Extension: DETNET-indicator 57
Authors' Addresses 60

1. Introduction

The Internet has largely been constructed for general purpose computers, those devices that may be used for a purpose that is specified by those who own 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 intended to be used for general purpose computing tasks. These devices, which this memo refers to as Things, have a specific purpose. By definition, therefore, all other uses are not intended. If a small number of communication patterns follows from those small number of uses, 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. MUD primarily addresses threats to the device rather than the device as a threat. In some circumstances, however, MUD may offer some protection in the latter case, depending on the MUD-URL is communicated, and how devices and their communications are authenticated.

We use the notion of "manufacturer" loosely in this context to refer to the entity or organization that will state how a device is intended to be used. For example, 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 an integrator of that device. The key points are that the device itself is assumed to serve a limited purpose, and that there exists an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The intent of MUD is to provide the following:

- Substantially reduce the threat surface on a device to those communications intended by the manufacturer.

- Provide a means to scale network policies to the ever-increasing number of types of devices in the network.

- Provide a means to address at least some vulnerabilities in a way that is faster than the time it might take to update systems. This will be particularly true for systems that are no longer supported.

- Keep the cost of implementation of such a system to the bare minimum.

- Provide a means of extensibility for manufacturers to express other device capabilities or requirements.

MUD consists of three architectural building blocks:

- A URL that can be used to locate a description;

- The description itself, including how it is interpreted, and;

- A means for local network management systems to retrieve the description.

MUD is most effective when the network is able to identify in some way the remote endpoints that Things will talk to.

In this specification we describe each of these building blocks and how they are intended to be used together. However, they may also be used separately, independent of this specification, by local deployments for their own purposes.

1.1. What MUD Doesn't Do

MUD is not intended to address network authorization of general purpose computers, as their manufacturers cannot envision a specific communication pattern to describe. In addition, even those devices that have a single or small number of uses might have very broad communication patterns. MUD on its own is not for them either.

Although MUD can provide network administrators with some additional protection when device vulnerabilities exist, it will never replace the need for manufacturers to patch vulnerabilities.

Finally, no matter what the manufacturer specifies in a MUD file, these are not directives, but suggestions. How they are instantiated locally will depend on many factors and will be ultimately up to the local network administrator, who must decide what is appropriate in a given circumstances.

1.2. 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 application on a smart phone. 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 social networking 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.

1.3. Terminology

MUD: manufacturer usage description.

MUD file: a file containing YANG-based JSON that describes a Thing and associated suggested specific network behavior.

MUD file server: a web server that hosts a MUD file.

MUD manager: the system that requests and receives the MUD file from the MUD server. After it has processed a MUD file, it may direct changes to relevant network elements.

MUD controller: a synonym that has been used in the past for MUD manager.

MUD URL: a URL that can be used by the MUD manager to receive the MUD file.

Thing: the device emitting a MUD URL.

Manufacturer: the entity that configures the Thing to emit the MUD URL and the one who asserts a recommendation in a MUD file. The manufacturer might not always be the entity that constructs a Thing. It could, for instance, be a systems integrator, or even a component provider.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.4. 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]. Profiling systems that make use of heuristics to identify types of systems have existed for years as well.

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

1.5. Finding A Policy: The MUD URL

Our work begins with the device emitting a Universal Resource Locator (URL) [RFC3986]. This URL serves both to classify the device type and to provide a means to locate a policy file.

MUD URLs MUST use the HTTPS scheme [RFC7230].

In this memo three means are defined to emit the MUD URL, as follows:

- o A DHCP option[RFC2131],[RFC3315] that the DHCP client uses to inform the DHCP server. The DHCP server may take further actions, such as act as the MUD manager or otherwise pass the MUD URL along to the MUD manager.

- o An X.509 constraint. The IEEE has developed [IEEE8021AR] that provides a certificate-based approach to communicate device characteristics, which itself relies on [RFC5280]. The MUD URL extension is non-critical, as required by IEEE 802.1AR. Various means may be used to communicate that certificate, including Tunnel Extensible Authentication Protocol (TEAP) [RFC7170].

- o Finally, a Link Layer Discovery Protocol (LLDP) frame is defined [IEEE8021AB].

It is possible that there may be other means for a MUD URL to be learned by a network. For instance, some devices may already be fielded or have very limited ability to communicate a MUD URL, and yet can be identified through some means, such as a serial number or a public key. In these cases, manufacturers may be able to map those identifiers to particular MUD URLs (or even the files themselves). Similarly, there may be alternative resolution mechanisms available for situations where Internet connectivity is limited or does not exist. Such mechanisms are not described in this memo, but are possible. Implementors are encouraged to allow for this sort of flexibility of how MUD URLs may be learned.

1.6. Processing of the MUD URL

MUD managers that are able to do so SHOULD retrieve MUD URLs and signature files as per [RFC7230], using the GET method [RFC7231]. They MUST validate the certificate using the rules in [RFC2818], Section 3.1.

Requests for MUD URLs SHOULD include an "Accept" header ([RFC7231], Section 5.3.2) containing "application/mud+json", an "Accept-Language" header field ([RFC7231], Section 5.3.5), and a "User-Agent" header ([RFC7231], Section 5.5.3).

MUD managers SHOULD automatically process 3xx response status codes.

If a MUD manager is not able to fetch a MUD URL, other means MAY be used to import MUD files and associated signature files. So long as the signature of the file can be validated, the file can be used. In such environments, controllers SHOULD warn administrators when cache-validity expiry is approaching so that they may check for new files.

It may not be possible for a MUD manager to retrieve a MUD file at any given time. Should a MUD manager fail to retrieve a MUD file, it SHOULD consider the existing one safe to use, at least for a time. After some period, it SHOULD log that it has been unable to retrieve the file. There may be very good reasons for such failures, including the possibility that the MUD manager is in an off-line environment, the local Internet connection has failed, or the remote Internet connection has failed. It is also possible that an attacker is attempting to interfere with the deployment of a device. It is a local decision as to how to handle such circumstances.

1.7. Types of Policies

When the MUD URL is resolved, the MUD manager retrieves a file that describes what sort of communications a device is designed to have. The manufacturer may specify either specific hosts for cloud based services or certain classes for access within an operational network. An example of a class might be "devices of a specified manufacturer type", where the manufacturer type itself is indicated simply by the authority component (e.g, the domain name) of the MUD URL. Another example might be to allow or disallow local access. Just like other policies, these may be combined. For example:

- o Allow access to devices of the same manufacturer
- o Allow access to and from controllers via Constrained Application Protocol (COAP)[RFC7252]
- o Allow access to local DNS/NTP
- o Deny all other access

A printer might have a description that states:

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

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

The files that are retrieved are intended to be closely aligned to existing network architectures so that they are easy to deploy. We make use of YANG [RFC7950] because it provides accurate and adequate models for use by network devices. JSON[RFC8259] is used as a serialization format for compactness and readability, relative to XML. Other formats may be chosen with later versions of MUD.

While the policy examples given here focus on access control, this is not intended to be the sole focus. By structuring the model described in this document with clear extension points, other descriptions could be included. One that often comes to mind is quality of service.

The YANG modules specified here are extensions of [I-D.ietf-netmod-acl-model]. The extensions to this model allow for a manufacturer to express classes of systems that a manufacturer would find necessary for the proper function of the device. Two modules are specified. The first module specifies a means for domain names to be used in ACLs so that devices that have their controllers in the cloud may be appropriately authorized with domain names, where the mapping of those names to addresses may rapidly change.

The other module abstracts away IP addresses into certain classes that are instantiated into actual IP addresses through local processing. Through these classes, manufacturers can specify how the device is designed to communicate, so that network elements can be configured by local systems that have local topological knowledge. That is, the deployment populates the classes that the manufacturer specifies. The abstractions below map to zero or more hosts, as follows:

Manufacturer: A device made by a particular manufacturer, as identified by the authority component of its MUD URL

same-manufacturer: Devices that have the same authority component of their MUD URL.

controller: Devices that the local network administrator admits to the particular class.

my-controller: Devices intended to serve as controllers for the MUD-URL that the Thing emitted.

local: The class of IP addresses that are scoped within some administrative boundary. By default it is suggested that this be the local subnet.

The "manufacturer" classes can be easily specified by the manufacturer, whereas controller classes are initially envisioned to be specified by the administrator.

Because manufacturers do not know who will be using their devices, it is important for functionality referenced in usage descriptions to be relatively ubiquitous and mature. For these reasons the YANG-based configuration in a MUD file is limited to either the modules specified or referenced in this document, or those specified in documented extensions.

1.8. The Manufacturer Usage Description Architecture

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

```
 .......................................
 .                      ____________   .           _____________
 .                     |            |  .          |             |
 .                     |    MUD     |-->get URL-->|    MUD      |
 .                     |  Manager   |  .(https)   | File Server |
 .  End system network |____________|<-MUD file<-<|_____________|
 .                             .       .
 .                             .       .
 . _______                 _________   .
 .|       |  (dhcp et al) | router  |  .
 .| Thing |---->MUD URL-->|   or    |  .
 .|_______|               | switch  |  .
 .                        |_________|  .
 .......................................
```

Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URLs and forwards them to the MUD manager (a network management system) for processing. This happens in different ways, depending on how the URL is communicated. For instance, in the case of DHCP, the DHCP server might receive the URL and then process it. In the case of IEEE 802.1X [IEEE8021X], the switch would carry the URL via a certificate to the authentication server via EAP over Radius[RFC3748], which would then process it. One method to do this is TEAP, described in [RFC7170]. The certificate extension is described below.

The information returned by the MUD file server is valid for as long as the Thing is connected. There is no expiry. However, if the MUD manager has detected that the MUD file for a Thing has changed, it SHOULD update the policy expeditiously, taking into account whatever approval flow is required in a deployment. In this way, new recommendations from the manufacturer can be processed in a timely fashion.

The information returned by the MUD file server (a web server) is valid for the duration of the Thing's connection, or as specified in the description. Thus if the Thing is disconnected, any associated configuration in the switch can be removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web server is typically run by or on behalf of the manufacturer. Its domain name is that of the authority found in the MUD URL. For legacy cases where Things cannot emit a URL, if the switch is able to determine the appropriate URL, it may proxy it. In the trivial case it may hardcode MUD-URL on a switch port or a map from some available identifier such as an L2 address or certificate hash to a MUD-URL.

The role of the MUD manager in this environment is to do the following:

- receive MUD URLs,
- fetch MUD files,
- translate abstractions in the MUD files to specific network element configuration,
- maintain and update any required mappings of the abstractions, and
- update network elements with appropriate configuration.

A MUD manager may be a component of a AAA or network management system. Communication within those systems and from those systems to network elements is beyond the scope of this memo.

1.9. Order of operations

As mentioned above, MUD contains architectural building blocks, and so order of operation may vary. However, here is one clear intended example:

1. Thing emits URL.
2. That URL is forwarded to a MUD manager by the nearest switch (how this happens depends on the way in which the MUD URL is emitted).
3. The MUD manager retrieves the MUD file and signature from the MUD file server, assuming it doesn't already have copies. After validating the signature, it may test the URL against a web or domain reputation service, and it may test any hosts within the file against those reputation services, as it deems fit.

4. The MUD manager may query the administrator for permission to add the Thing and associated policy. If the Thing is known or the Thing type is known, it may skip this step.

5. The MUD manager instantiates local configuration based on the abstractions defined in this document.

6. The MUD manager configures the switch nearest the Thing. Other systems may be configured as well.

7. When the Thing disconnects, policy is removed.

2. The MUD Model and Semantic Meaning

A MUD file consists of a YANG model instance that has been serialized in JSON [RFC7951]. For purposes of MUD, the nodes that can be modified are access lists as augmented by this model. The MUD file is limited to the serialization of only the following YANG schema:

- ietf-access-control-list [I-D.ietf-netmod-acl-model]

- ietf-mud (this document)

- ietf-acldns (this document)

Extensions may be used to add additional schema. This is described further on.

To provide the widest possible deployment, publishers of MUD files SHOULD make use of the abstractions in this memo and avoid the use of IP addresses. A MUD manager SHOULD NOT automatically implement any MUD file that contains IP addresses, especially those that might have local significance. The addressing of one side of an access list is implicit, based on whether it is applied as to-device-policy or from-device-policy.

With the exceptions of "name" of the ACL, "type", "name" of the ACE, and TCP and UDP source and destination port information, publishers of MUD files SHOULD limit the use of ACL model leaf nodes expressed to those found in this specification. Absent any extensions, MUD files are assumed to implement only the following ACL model features:

- match-on-ipv4, match-on-ipv6, match-on-tcp, match-on-udp, match-on-icmp

Furthermore, only "accept" or "drop" actions SHOULD be included. A MUD manager MAY choose to interpret "reject" as "drop". A MUD manager SHOULD ignore all other actions. This is because manufacturers do not have sufficient context within a local deployment to know whether reject is appropriate. That is a decision that should be left to a network administrator.

Given that MUD does not deal with interfaces, the support of the "ietf-interfaces" module [RFC8343] is not required. Specifically, the support of interface-related features and branches (e.g., interface-attachment and interface-stats) of the ACL YANG module is not required.

In fact, MUD managers MAY ignore any particular component of a description or MAY ignore the description in its entirety, and SHOULD carefully inspect all MUD descriptions. Publishers of MUD files MUST NOT include other nodes except as described in Section 3.9. See that section for more information.

2.1. The IETF-MUD YANG Module

This module is structured into three parts:

- o The first component, the "mud" container, holds information that is relevant to retrieval and validity of the MUD file itself, as well as policy intended to and from the Thing.

- o The second component augments the matching container of the ACL model to add several nodes that are relevant to the MUD URL, or otherwise abstracted for use within a local environment.

- o The third component augments the tcp-acl container of the ACL model to add the ability to match on the direction of initiation of a TCP connection.

A valid MUD file will contain two root objects, a "mud" container and an "acls" container. Extensions may add additional root objects as required. As a reminder, when parsing acls, elements within a "match" block are logically ANDed. In general, a single abstraction in a match statement should be used. For instance, it makes little sense to match both "my-controller" and "controller" with an argument, since they are highly unlikely to be the same value.

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is explained in [RFC8340].

```
module: ietf-mud
  +--rw mud!
     +--rw mud-version           uint8
     +--rw mud-url               inet:uri
     +--rw last-update           yang:date-and-time
     +--rw mud-signature?        inet:uri
     +--rw cache-validity?       uint8
     +--rw is-supported          boolean
     +--rw systeminfo?           string
     +--rw mfg-name?             string
     +--rw model-name?           string
     +--rw firmware-rev?         string
     +--rw software-rev?         string
     +--rw documentation?        inet:uri
     +--rw extensions*           string
     +--rw from-device-policy
     |  +--rw acls
     |     +--rw access-list* [name]
     |        +--rw name    -> /acl:acls/acl/name
     +--rw to-device-policy
        +--rw acls
           +--rw access-list* [name]
              +--rw name    -> /acl:acls/acl/name

  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw mud
       +--rw manufacturer?        inet:host
       +--rw same-manufacturer?   empty
       +--rw model?               inet:uri
       +--rw local-networks?      empty
       +--rw controller?          inet:uri
       +--rw my-controller?       empty
  augment
    /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches
       /acl:l4/acl:tcp/acl:tcp:
    +--rw direction-initiated?   direction
```

3. MUD model definitions for the root mud container

3.1. mud-version

This node specifies the integer version of the MUD specification. This memo specifies version 1.

3.2. mud-url

This URL identifies the MUD file. This is useful when the file and associated signature are manually uploaded, say, in an offline mode.

3.3. to-device-policy and from-device-policy containers

[I-D.ietf-netmod-acl-model] describes access-lists. In the case of MUD, a MUD file must be explicit in describing the communication pattern of a Thing, and that includes indicating what is to be permitted or denied in either direction of communication. Hence each of these containers indicates the appropriate direction of a flow in association with a particular Thing. They contain references to specific access-lists.

3.4. last-update

This is a date-and-time value of when the MUD file was generated. This is akin to a version number. Its form is taken from [RFC6991] which, for those keeping score, in turn was taken from Section 5.6 of [RFC3339], which was taken from [ISO.8601.1988].

3.5. cache-validity

This uint8 is the period of time in hours that a network management station MUST wait since its last retrieval before checking for an update. It is RECOMMENDED that this value be no less than 24 and MUST NOT be more than 168 for any Thing that is supported. This period SHOULD be no shorter than any period determined through HTTP caching directives (e.g., "cache-control" or "Expires"). N.B., expiring of this timer does not require the MUD manager to discard the MUD file, nor terminate access to a Thing. See Section 16 for more information.

3.6. is-supported

This boolean is an indication from the manufacturer to the network administrator as to whether or not the Thing is supported. In this context a Thing is said to not be supported if the manufacturer intends never to issue a firmware or software update to the Thing or never update the MUD file. A MUD manager MAY still periodically check for updates.

3.7. systeminfo

This is a textual UTF-8 description of the Thing to be connected. The intent is for administrators to be able to see a brief displayable description of the Thing. It SHOULD NOT exceed 60 characters worth of display space.

3.8. mfg-name, software-rev, model-name firmware-rev

These optional fields are filled in as specified by [RFC8348]. Note that firmware-rev and software-rev MUST NOT be populated in a MUD file if the device can be upgraded but the MUD-URL cannot be. This would be the case, for instance, with MUD-URLs that are contained in 802.1AR certificates.

3.9. extensions

This optional leaf-list names MUD extensions that are used in the MUD file. Note that MUD extensions MUST NOT be used in a MUD file without the extensions being declared. Implementations MUST ignore any node in this file that they do not understand.

Note that extensions can either extend the MUD file as described in the previous paragraph, or they might reference other work. An extension example can be found in Appendix C.

4. Augmentation to the ACL Model

Note that in this section, when we use the term "match" we are referring to the ACL model "matches" node.

4.1. manufacturer

This node consists of a hostname that would be matched against the authority component of another Thing’s MUD URL. In its simplest form "manufacturer" and "same-manufacturer" may be implemented as access-lists. In more complex forms, additional network capabilities may be used. For example, if one saw the line "manufacturer" : "flobbidy.example.com", then all Things that registered with a MUD URL that contained flobbity.example.com in its authority section would match.

4.2. same-manufacturer

This null-valued node is an equivalent for when the manufacturer element is used to indicate the authority that is found in another Thing’s MUD URL matches that of the authority found in this Thing’s MUD URL. For example, if the Thing’s MUD URL were https://b1.example.com/ThingV1, then all devices that had MUD URL with an authority section of b1.example.com would match.

4.3. documentation

This URI consists of a URL that points to documentation relating to the device and the MUD file. This can prove particularly useful when the "controller" class is used, so that its use can be explained.

4.4. model

This string matches the entire MUD URL, thus covering the model that is unique within the context of the authority. It may contain not only model information, but versioning information as well, and any other information that the manufacturer wishes to add. The intended use is for devices of this precise class to match, to permit or deny communication between one another.

4.5. local-networks

This null-valued node expands to include local networks. Its default expansion is that packets must not traverse toward a default route that is received from the router. However, administrators may expand the expression as is appropriate in their deployments.

4.6. controller

This URI specifies a value that a controller will register with the MUD manager. The node then is expanded to the set of hosts that are so registered. This node may also be a URN. In this case, the URN describes a well known service, such as DNS or NTP, that has been standardized. Both of those URNs may be found in Section 17.6.

When "my-controller" is used, it is possible that the administrator will be prompted to populate that class for each and every model. Use of "controller" with a named class allows the user to populate that class only once for many different models that a manufacturer may produce.

Controller URIs MAY take the form of a URL (e.g. "http[s]://"). However, MUD managers MUST NOT resolve and retrieve such files, and it is RECOMMENDED that there be no such file at this time, as their form and function may be defined at a point in the future. For now, URLs should serve simply as class names and may be populated by the local deployment administrator.

Great care should be taken by MUD managers when invoking the controller class in the form of URLs. For one thing, it requires some understanding by the administrator as to when it is appropriate. Pre-registration in such classes by controllers with the MUD server is encouraged. The mechanism to do that is beyond the scope of this work.

4.7. my-controller

This null-valued node signals to the MUD manager to use whatever mapping it has for this MUD URL to a particular group of hosts. This may require prompting the administrator for class members. Future work should seek to automate membership management.

4.8. direction-initiated

This MUST only be applied to TCP. This matches the direction in which a TCP connection is initiated. When direction initiated is "from-device", packets that are transmitted in the direction of a thing MUST be dropped unless the thing has first initiated a TCP connection. By way of example, this node may be implemented in its simplest form by looking at naked SYN bits, but may also be implemented through more stateful mechanisms.

When applied this matches packets when the flow was initiated in the corresponding direction. [RFC6092] specifies IPv6 guidance best practices. While that document is scoped specifically to IPv6, its contents are applicable for IPv4 as well.

5. Processing of the MUD file

To keep things relatively simple in addition to whatever definitions exist, we also apply two additional default behaviors:

- o Anything not explicitly permitted is denied.
- o Local DNS and NTP are, by default, permitted to and from the Thing.

An explicit description of the defaults can be found in Appendix B. These are applied AFTER all other explicit rules. Thus, a default behavior can be changed with a "drop" action.

6. What does a MUD URL look like?

MUD URLs are required to use the HTTPS scheme, in order to establish the MUD file server’s identity and assure integrity of the MUD file.

Any "https://" URL can be a MUD URL. For example:

```
     https://things.example.org/product_abc123/v5
     https://www.example.net/mudfiles/temperature_sensor/
     https://example.com/lightbulbs/colour/v1
```

A manufacturer may construct a MUD URL in any way, so long as it makes use of the "https" schema.

7. The MUD YANG Model

```
   <CODE BEGINS>file "ietf-mud@2018-06-15.yang"
   module ietf-mud {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-mud";
     prefix ietf-mud;

     import ietf-access-control-list {
       prefix acl;
     }
     import ietf-yang-types {
       prefix yang;
     }
     import ietf-inet-types {
       prefix inet;
     }

     organization
       "IETF OPSAWG (Ops Area) Working Group";
     contact
       "WG Web: http://tools.ietf.org/wg/opsawg/
        WG List: opsawg@ietf.org
        Author: Eliot Lear
        lear@cisco.com
        Author: Ralph Droms
        rdroms@gmail.com
        Author: Dan Romascanu
        dromasca@gmail.com

       ";
     description
       "This YANG module defines a component that augments the
        IETF description of an access list.  This specific module
        focuses on additional filters that include local, model,
        and same-manufacturer.

        This module is intended to be serialized via JSON and stored
        as a file, as described in RFC XXXX [RFC Editor to fill in with
        this document #].
```

```
        Copyright (c) 2016,2017 IETF Trust and the persons
        identified as the document authors.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD
        License set forth in Section 4.c of the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2018-06-15 {
       description
         "Initial proposed standard.";
       reference
         "RFC XXXX: Manufacturer Usage Description
          Specification";
     }

     typedef direction {
       type enumeration {
         enum to-device {
           description
             "packets or flows destined to the target
              Thing";
         }
         enum from-device {
           description
             "packets or flows destined from
              the target Thing";
         }
       }
       description
         "Which way are we talking about?";
     }

     container mud {
       presence "Enabled for this particular MUD URL";
       description
         "MUD related information, as specified
          by RFC-XXXX [RFC Editor to fill in].";
       uses mud-grouping;
     }

     grouping mud-grouping {
       description
         "Information about when support end(ed), and
          when to refresh";
```

```
       leaf mud-version {
         type uint8;
         mandatory true;
         description
           "This is the version of the MUD
            specification.  This memo specifies version 1.";
       }
       leaf mud-url {
         type inet:uri;
         mandatory true;
         description
           "This is the MUD URL associated with the entry found
            in a MUD file.";
       }
       leaf last-update {
         type yang:date-and-time;
         mandatory true;
         description
           "This is intended to be when the current MUD file
            was generated.  MUD Managers SHOULD NOT check
            for updates between this time plus cache validity";
       }
       leaf mud-signature {
         type inet:uri;
         description
           "A URI that resolves to a signature as
            described in this specification.";
       }
       leaf cache-validity {
         type uint8 {
           range "1..168";
         }
         units "hours";
         default "48";
         description
           "The information retrieved from the MUD server is
            valid for these many hours, after which it should
            be refreshed.  N.B. MUD manager implementations
            need not discard MUD files beyond this period.";
       }
       leaf is-supported {
         type boolean;
         mandatory true;
         description
           "This boolean indicates whether or not the Thing is
            currently supported by the manufacturer.";
       }
       leaf systeminfo {
```

```
         type string;
         description
           "A UTF-8 description of this Thing.  This
            should be a brief description that may be
            displayed to the user to determine whether
            to allow the Thing on the
            network.";
       }
       leaf mfg-name {
         type string;
         description
           "Manufacturer name, as described in
            the ietf-hardware YANG module.";
       }
       leaf model-name {
         type string;
         description
           "Model name, as described in the
            ietf-hardware YANG module.";
       }
       leaf firmware-rev {
         type string;
         description
           "firmware-rev, as described in the
            ietf-hardware YANG module.  Note this field MUST
            NOT be included when the device can be updated
            but the MUD-URL cannot.";
       }
       leaf software-rev {
         type string;
         description
           "software-rev, as described in the
            ietf-hardware YANG module.  Note this field MUST
            NOT be included when the device can be updated
            but the MUD-URL cannot.";
       }
       leaf documentation {
         type inet:uri;
         description
           "This URL points to documentation that
            relates to this device and any classes that it uses
            in its MUD file.  A caution: MUD managers need
            not resolve this URL on their own, but rather simply
            provide it to the administrator.  Parsing HTML is
            not an intended function of a MUD manager.";
       }
       leaf-list extensions {
         type string {
```

```
           length "1..40";
         }
         description
           "A list of extension names that are used in this MUD
            file.  Each name is registered with the IANA and
            described in an RFC.";
       }
       container from-device-policy {
         description
           "The policies that should be enforced on traffic
            coming from the device. These policies are not
            necessarily intended to be enforced at a single
            point, but may be rendered by the controller to any
            relevant enforcement points in the network or
            elsewhere.";
         uses access-lists;
       }
       container to-device-policy {
         description
           "The policies that should be enforced on traffic
            going to the device. These policies are not
            necessarily intended to be enforced at a single
            point, but may be rendered by the controller to any
            relevant enforcement points in the network or
            elsewhere.";
         uses access-lists;
       }
     }

     grouping access-lists {
       description
         "A grouping for access lists in the context of device
          policy.";
       container access-lists {
         description
           "The access lists that should be applied to traffic
              to or from the device.";
         list access-list {
           key "name";
           description
             "Each entry on this list refers to an ACL that
                should be present in the overall access list
                data model. Each ACL is identified by name and
                type.";
           leaf name {
             type leafref {
               path "/acl:acls/acl:acl/acl:name";
             }
```

```
             description
               "The name of the ACL for this entry.";
           }
         }
       }
     }

     augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
       description
         "adding abstractions to avoid need of IP addresses";
       container mud {
         description
           "MUD-specific matches.";
         leaf manufacturer {
           type inet:host;
           description
             "A domain that is intended to match the authority
              section of the MUD URL. This node is used to specify
              one or more manufacturers a device should
              be authorized to access.";
         }
         leaf same-manufacturer {
           type empty;
           description
             "This node matches the authority section of the MUD URL
              of a Thing.  It is intended to grant access to all
              devices with the same authority section.";
         }
         leaf model {
           type inet:uri;
           description
             "Devices of the specified model type will match if
              they have an identical MUD URL.";
         }
         leaf local-networks {
           type empty;
           description
             "IP addresses will match this node if they are
              considered local addresses.  A local address may be
              a list of locally defined prefixes and masks
              that indicate a particular administrative scope.";
         }
         leaf controller {
           type inet:uri;
           description
             "This node names a class that has associated with it
              zero or more IP addresses to match against.  These
              may be scoped to a manufacturer or via a standard
```

```
              URN.";
         }
         leaf my-controller {
           type empty;
           description
             "This node matches one or more network elements that
              have been configured to be the controller for this
              Thing, based on its MUD URL.";
         }
       }
     }
     augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
       "/acl:l4/acl:tcp/acl:tcp" {
       description
         "add direction-initiated";
       leaf direction-initiated {
         type direction;
         description
           "This node matches based on which direction a
            connection was initiated. The means by which that
            is determined is discussed in this document.";
       }
     }
   }
   <CODE ENDS>
```

8. The Domain Name Extension to the ACL Model

This module specifies an extension to IETF-ACL model such that domain names may be referenced by augmenting the "matches" node. Different implementations may deploy differing methods to maintain the mapping between IP address and domain name, if indeed any are needed. However, the intent is that resources that are referred to using a name should be authorized (or not) within an access list.

The structure of the change is as follows:

```
module: ietf-acldns
  augment /acl:acls/acl:acl/acl:aces/acl:ace/
    acl:matches/acl:l3/acl:ipv4/acl:ipv4:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
  augment /acl:acls/acl:acl/acl:aces/acl:ace/
    acl:matches/acl:l3/acl:ipv6/acl:ipv6:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
```

The choice of these particular points in the access-list model is based on the assumption that we are in some way referring to IP-related resources, as that is what the DNS returns. A domain name in our context is defined in [RFC6991]. The augmentations are replicated across IPv4 and IPv6 to allow MUD file authors the ability to control the IP version that the Thing may utilize.

The following node are defined.

8.1. src-dnsname

The argument corresponds to a domain name of a source as specified by inet:host. A number of means may be used to resolve hosts. What is important is that such resolutions be consistent with ACLs required by Things to properly operate.

8.2. dst-dnsname

The argument corresponds to a domain name of a destination as specified by inet:host See the previous section relating to resolution.

Note when using either of these with a MUD file, because access is associated with a particular Thing, MUD files MUST NOT contain either a src-dnsname in an ACL associated with from-device-policy or a dst-dnsname associated with to-device-policy.

8.3. The ietf-acldns Model

```
<CODE BEGINS>file "ietf-acldns@2018-06-15.yang"
module ietf-acldns {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acldns";
  prefix ietf-acldns;

  import ietf-access-control-list {
    prefix acl;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact
    "WG Web: http://tools.ietf.org/wg/opsawg/
     WG List: opsawg@ietf.org
     Author: Eliot Lear
```

```
           lear@cisco.com
           Author: Ralph Droms
           rdroms@gmail.com
           Author: Dan Romascanu
           dromasca@gmail.com
          ";
        description
          "This YANG module defines a component that augments the
           IETF description of an access list to allow DNS names
           as matching criteria.";

        revision 2018-06-15 {
          description
            "Base version of dnsname extension of ACL model";
          reference
            "RFC XXXX: Manufacturer Usage Description
             Specification";
        }

        grouping dns-matches {
          description
            "Domain names for matching.";
          leaf src-dnsname {
            type inet:host;
            description
              "domain name to be matched against";
          }
          leaf dst-dnsname {
            type inet:host;
            description
              "domain name to be matched against";
          }
        }

        augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
         "/acl:l3/acl:ipv4/acl:ipv4" {
          description
            "Adding domain names to matching";
          uses dns-matches;
        }
        augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
         "/acl:l3/acl:ipv6/acl:ipv6" {
          description
            "Adding domain names to matching";
          uses dns-matches;
        }
      }
      <CODE ENDS>
```

9. MUD File Example

This example contains two access lists that are intended to provide outbound access to a cloud service on TCP port 443.


```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.example.com/lightbulb2000",
    "last-update": "2018-03-02T11:20:51+01:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "The BMS Example Lightbulb",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname": "test.example.com",
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated": "from-device",
```

```
                     "source-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         },
         {
           "name": "mud-76100-v6fr",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "cl0-frdev",
                 "matches": {
                   "ipv6": {
                     "ietf-acldns:dst-dnsname": "test.example.com",
                     "protocol": 6
                   },
                   "tcp": {
                     "ietf-mud:direction-initiated": "from-device",
                     "destination-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         }
       ]
     }
   }
```


In this example, two policies are declared, one from the Thing and the other to the Thing. Each policy names an access list that applies to the Thing, and one that applies from. Within each access list, access is permitted to packets flowing to or from the Thing that can be mapped to the domain name of "service.bms.example.com". For each access list, the enforcement point should expect that the Thing initiated the connection.

10. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:

```
+------+-----+------------------------------
| code | len |  MUDstring
+------+-----+------------------------------
```

Code OPTION_MUD_URL_V4 (161) is assigned by IANA. len is a single octet that indicates the length of MUD string in octets. The MUD string is defined as follows:

```
MUDstring = mudurl [ " " reserved ]
mudurl = URI; a URL [RFC3986] that uses the "https" schema [RFC7230]
reserved = 1*( OCTET ) ; from [RFC5234]
```

The entire option MUST NOT exceed 255 octets. If a space follows the MUD URL, a reserved string that will be defined in future specifications follows. MUD managers that do not understand this field MUST ignore it.

The IPv6 MUD URL client option has the following format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_MUD_URL_V6     |        option-length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            MUDstring                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

OPTION_MUD_URL_V6 (112; assigned by IANA).

option-length contains the length of the MUDstring, as defined above, in octets.

The intent of this option is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this option is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

10.1. Client Behavior

A DHCPv4 client MAY emit a DHCPv4 option and a DHCPv6 client MAY emit DHCPv6 option. These options are singletons, as specified in [RFC7227]. Because clients are intended to have at most one MUD URL associated with them, they may emit at most one MUD URL option via DHCPv4 and one MUD URL option via DHCPv6. In the case where both v4 and v6 DHCP options are emitted, the same URL MUST be used.

10.2. Server Behavior

A DHCP server may ignore these options or take action based on receipt of these options. When a server consumes this option, it will either forward the URL and relevant client information (such as the gateway address or giaddr and requested IP address, and lease length) to a network management system, or it will retrieve the usage description itself by resolving the URL.

DHCP servers may implement MUD functionality themselves or they may pass along appropriate information to a network management system or MUD manager. A DHCP server that does process the MUD URL MUST adhere to the process specified in [RFC2818] and [RFC5280] to validate the TLS certificate of the web server hosting the MUD file. Those servers will retrieve the file, process it, create and install the necessary configuration on the relevant network element. Servers SHOULD monitor the gateway for state changes on a given interface. A DHCP server that does not provide MUD functionality and has forwarded a MUD URL to a MUD manager MUST notify the MUD manager of any corresponding change to the DHCP state of the client (such as expiration or explicit release of a network address lease).

Should the DHCP server fail, in the case when it implements the MUD manager functionality, any backup mechanisms SHOULD include the MUD state, and the server SHOULD resolve the status of clients upon its restart, similar to what it would do, absent MUD manager functionality. In the case where the DHCP server forwards information to the MUD manager, the MUD manager will either make use of redundant DHCP servers for information, or otherwise clear state based on other network information, such as monitoring port status on a switch via SNMP, Radius accounting, or similar mechanisms.

10.3. Relay Requirements

There are no additional requirements for relays.

11. The Manufacturer Usage Description (MUD) URL X.509 Extension

This section defines an X.509 non-critical certificate extension that contains a single Uniform Resource Locator (URL) that points to an on-line Manufacturer Usage Description concerning the certificate subject. URI must be represented as described in Section 7.4 of [RFC5280].

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in Section 3.1 of [RFC3987] before they are placed in the certificate extension.

The semantics of the URL are defined Section 6 of this document.

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280]:

> These extensions may be used to direct applications to on-line information about the issuer or the subject.

The MUD URL is precisely that: online information about the particular subject.

In addition, a separate new extension is defined as id-pe-mudsigner. This contains the subject field of the signing certificate of the MUD file. Processing of this field is specified in Section 13.2.

The purpose of this signature is to make a claim that the MUD file found on the server is valid for a given device, independent of any other factors. There are several security considerations below in Section 16.

A new content-type id-ct-mud is also defined. While signatures are detached today, should a MUD file be transmitted as part of a CMS message, this content-type SHOULD be used.

The new extension is identified as follows:

```
<CODE BEGINS>
   MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
                internet(1) security(5) mechanisms(5) pkix(7)
                id-mod(0) id-mod-mudURLExtn2016(88) }
    DEFINITIONS IMPLICIT TAGS ::= BEGIN
```

```
      -- EXPORTS ALL --

     IMPORTS

       -- RFC 5912
       EXTENSION
       FROM PKIX-CommonTypes-2009
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5) mechanisms(5) pkix(7) id-mod(0)
              id-mod-pkixCommon-02(57) }

       -- RFC 5912
       id-ct
       FROM PKIXCRMF-2009
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5)  mechanisms(5) pkix(7) id-mod(0)
              id-mod-crmf2005-02(55) }

       -- RFC 6268
       CONTENT-TYPE
       FROM CryptographicMessageSyntax-2010
         { iso(1) member-body(2) us(840) rsadsi(113549)
            pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

       -- RFC 5912
       id-pe, Name
       FROM PKIX1Explicit-2009
             { iso(1) identified-organization(3) dod(6) internet(1)
               security(5) mechanisms(5) pkix(7) id-mod(0)
               id-mod-pkix1-explicit-02(51) } ;

      --
      -- Certificate Extensions
      --

      MUDCertExtensions EXTENSION ::=
         { ext-MUDURL | ext-MUDsigner, ... }

      ext-MUDURL EXTENSION ::=
         { SYNTAX MUDURLSyntax IDENTIFIED BY id-pe-mud-url }

      id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }

      MUDURLSyntax ::= IA5String

      ext-MUDsigner EXTENSION ::=
         { SYNTAX MUDsignerSyntax IDENTIFIED BY id-pe-mudsigner }
```

```
      id-pe-mudsigner OBJECT IDENTIFIER ::= { id-pe TBD1 }

      MUDsignerSyntax ::= Name

      --
      -- CMS Content Types
      --

      MUDContentTypes CONTENT-TYPE ::=
         { ct-mud, ... }

       ct-mud CONTENT-TYPE ::=
         { -- directly include the content
           IDENTIFIED BY id-ct-mudtype }
         -- The binary data that is in the form
         -- 'application/mud+json" is directly encoded as the
         -- signed data.  No additional ASN.1 encoding is added.

      id-ct-mudtype OBJECT IDENTIFIER ::= { id-ct TBD2 }

      END
  <CODE ENDS>
```

While this extension can appear in either an 802.AR manufacturer certificate (IDevID) or deployment certificate (LDevID), of course it is not guaranteed in either, nor is it guaranteed to be carried over. It is RECOMMENDED that MUD manager implementations maintain a table that maps a Thing to its MUD URL based on IDevIDs.

12. The Manufacturer Usage Description LLDP extension

The IEEE802.1AB Link Layer Discovery Protocol (LLDP) is a one hop vendor-neutral link layer protocol used by end hosts network Things for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network. Its Type-Length-Value (TLV) design allows for 'vendor-specific' extensions to be defined. IANA has a registered IEEE 802 organizationally unique identifier (OUI) defined as documented in [RFC7042]. The MUD LLDP extension uses a subtype defined in this document to carry the MUD URL.

The LLDP vendor specific frame has the following format:

```
+--------+--------+----------+---------+--------------
|TLV Type|  len   |   OUI    |subtype  | MUDString
|  =127  |        |= 00 00 5E|  = 1    |
|(7 bits)|(9 bits)|(3 octets)|(1 octet)|(1-255 octets)
+--------+--------+----------+---------+--------------
```

where:

- o TLV Type = 127 indicates a vendor-specific TLV
- o len - indicates the TLV string length
- o OUI = 00 00 5E is the organizationally unique identifier of IANA
- o subtype = 1 (to be assigned by IANA for the MUD URL)
- o MUD URL - the length MUST NOT exceed 255 octets

The intent of this extension is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this extension is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

Hosts, routers, or other network elements that implement this option are intended to have at most one MUD URL associated with them, so they may transmit at most one MUD URL value.

Hosts, routers, or other network elements that implement this option may ignore these options or take action based on receipt of these options. For example they may fill in information in the respective extensions of the LLDP Management Information Base (LLDP MIB). LLDP operates in a one-way direction. LLDPDUs are not exchanged as information requests by one Thing and response sent by another Thing. The other Things do not acknowledge LLDP information received from a Thing. No specific network behavior is guaranteed. When a Thing consumes this extension, it may either forward the URL and relevant remote Thing information to a MUD manager, or it will retrieve the usage description by resolving the URL in accordance with normal HTTP semantics.

13. Creating and Processing of Signed MUD Files

Because MUD files contain information that may be used to configure network access lists, they are sensitive. To ensure that they have not been tampered with, it is important that they be signed. We make use of DER-encoded Cryptographic Message Syntax (CMS) [RFC5652] for this purpose.

13.1. Creating a MUD file signature

A MUD file MUST be signed using CMS as an opaque binary object. In order to make successful verification more likely, intermediate certificates SHOULD be included. The signature is stored at the location specified in the MUD file. Signatures are transferred using content-type "application/pkcs7-signature".

For example:

```
% openssl cms -sign -signer mancertfile -inkey mankey \
              -in mudfile -binary -outform DER -binary \
              -certfile intermediatecert -out mudfile.p7s
```

Note: A MUD file may need to be re-signed if the signature expires.

13.2. Verifying a MUD file signature

Prior to processing the rest of a MUD file, the MUD manager MUST retrieve the MUD signature file by retrieving the value of "mud-signature" and validating the signature across the MUD file. The Key Usage Extension in the signing certificate MUST be present and have the bit digitalSignature(0) set. When the id-pe-mudsigner extension is present in a device's X.509 certificate, the MUD signature file MUST have been generated by a certificate whose subject matches the contents of that id-pe-mudsigner extension. If these conditions are not met, or if it cannot validate the chain of trust to a known trust anchor, the MUD manager MUST cease processing the MUD file until an administrator has given approval.

The purpose of the signature on the file is to assign accountability to an entity, whose reputation can be used to guide administrators on whether or not to accept a given MUD file. It is already common place to check web reputation on the location of a server on which a file resides. While it is likely that the manufacturer will be the signer of the file, this is not strictly necessary, and may not be desirable. For one thing, in some environments, integrators may install their own certificates. For another, what is more important is the accountability of the recommendation, and not just the relationship between the Thing and the file.

An example:

```
% openssl cms -verify -in mudfile.p7s -inform DER -content mudfile
```

Note the additional step of verifying the common trust root.

14. Extensibility

One of our design goals is to see that MUD files are able to be understood by as broad a cross-section of systems as is possible. Coupled with the fact that we have also chosen to leverage existing mechanisms, we are left with no ability to negotiate extensions and a limited desire for those extensions in any event. A such, a two-tier extensibility framework is employed, as follows:

1. At a coarse grain, a protocol version is included in a MUD URL. This memo specifies MUD version 1. Any and all changes are entertained when this version is bumped. Transition approaches between versions would be a matter for discussion in future versions.

2. At a finer grain, only extensions that would not incur additional risk to the Thing are permitted. Specifically, adding nodes to the mud container is permitted with the understanding that such additions will be ignored by unaware implementations. Any such extensions SHALL be standardized through the IETF process, and MUST be named in the "extensions" list. MUD managers MUST ignore YANG nodes they do not understand and SHOULD create an exception to be resolved by an administrator, so as to avoid any policy inconsistencies.

15. Deployment Considerations

Because MUD consists of a number of architectural building blocks, it is possible to assemble different deployment scenarios. One key aspect is where to place policy enforcement. In order to protect the Thing from other Things within a local deployment, policy can be enforced on the nearest switch or access point. In order to limit unwanted traffic within a network, it may also be advisable to enforce policy as close to the Internet as possible. In some circumstances, policy enforcement may not be available at the closest hop. At that point, the risk of lateral infection (infection of devices that reside near one another) is increased to the number of Things that are able to communicate without protection.

A caution about some of the classes: admission of a Thing into the "manufacturer" and "same-manufacturer" class may have impact on access of other Things. Put another way, the admission may grow the access-list on switches connected to other Things, depending on how access is managed. Some care should be given on managing that access-list growth. Alternative methods such as additional network segmentation can be used to keep that growth within reason.

Because as of this writing MUD is a new concept, one can expect a great many devices to not have implemented it. It remains a local deployment decision as to whether a device that is first connected should be allowed broad or limited access. Furthermore, as mentioned in the introduction, a deployment may choose to ignore a MUD policy in its entirety, but simply taken into account the MUD URL as a classifier to be used as part of a local policy decision.

Finally, please see directly below regarding device lifetimes and use of domain names.

16. Security Considerations

Based on how a MUD URL is emitted, a Thing may be able to lie about what it is, thus gaining additional network access. This can happen in a number of ways when a device emits a MUD URL using DHCP or LLDP, such as being inappropriately admitted to a class such as "same-manufacturer", given access to a device such as "my-controller", or being permitted access to an Internet resource, where such access would otherwise be disallowed. Whether that is the case will depend on the deployment. Implementations SHOULD be configurable to disallow additive access for devices using MUD-URLs that are not emitted in a secure fashion such as in a certificate. Similarly, implementations SHOULD NOT grant elevated permissions (beyond those of devices presenting no MUD policy) to devices which do not strongly bind their identity to their L2/L3 transmissions. When insecure methods are used by the MUD Manager, the classes SHOULD NOT contain devices that use both insecure and secure methods, in order to prevent privilege escalation attacks, and MUST NOT contain devices with the same MUD-URL that are derived from both strong and weak authentication methods.

Devices may forge source (L2/L3) information. Deployments should apply appropriate protections to bind communications to the authentication that has taken place. For 802.1X authentication, IEEE 802.1AE (MACsec) [IEEE8021AE] is one means by which this may happen. A similar approach can be used with 802.11i (WPA2) [IEEE80211i]. Other means are available with other lower layer technologies. Implementations using session-oriented access that is not cryptographically bound should take care to remove state when any form of break in the session is detected.

A rogue CA may sign a certificate that contains the same subject name as is listed in the MUDsigner field in the manufacturer certificate, thus seemingly permitting a substitute MUD file for a device. There are two mitigations available: first, if the signer changes, this may be flagged as an exception by the MUD manager. If the MUD file also changes, the MUD manager SHOULD seek administrator approval (it should do this in any case). In all circumstances, the MUD manager MUST maintain a cache of trusted CAs for this purpose. When such a rogue is discovered, it SHOULD be removed.

Additional mitigations are described below.

When certificates are not present, Things claiming to be of a certain manufacturer SHOULD NOT be included in that manufacturer grouping without additional validation of some form. This will be relevant whenthe MUD manager makes use of primitives such as "manufacturer" for the purpose of accessing Things of a particular type. Similarly, network management systems may be able to fingerprint the Thing. In such cases, the MUD URL can act as a classifier that can be proven or disproven. Fingerprinting may have other advantages as well: when 802.1AR certificates are used, because they themselves cannot change, fingerprinting offers the opportunity to add artifacts to the MUD string in the form of the reserved field discussed in Section 10. The meaning of such artifacts is left as future work.

MUD managers SHOULD NOT accept a usage description for a Thing with the same MAC address that has indicated a change of the URL authority without some additional validation (such as review by a network administrator). New Things that present some form of unauthenticated MUD URL SHOULD be validated by some external means when they would be be given increased network access.

It may be possible for a rogue manufacturer to inappropriately exercise the MUD file parser, in order to exploit a vulnerability. There are three recommended approaches to address this threat. The first is to validate that the signer of the MUD file is known to and trusted by the MUD manager. The second is to have a system do a primary scan of the file to ensure that it is both parseable and believable at some level. MUD files will likely be relatively small, to start with. The number of ACEs used by any given Thing should be relatively small as well. It may also be useful to limit retrieval of MUD URLs to only those sites that are known to have decent web or domain reputations.

Use of a URL necessitates the use of domain names. If a domain name changes ownership, the new owner of that domain may be able to provide MUD files that MUD managers would consider valid. There are a few approaches that can mitigate this attack. First, MUD managers SHOULD cache certificates used by the MUD file server. When a new certificate is retrieved for whatever reason, the MUD manager should check to see if ownership of the domain has changed. A fair programmatic approximation of this is when the name servers for the domain have changed. If the actual MUD file has changed, the MUD manager MAY check the WHOIS database to see if registration ownership of a domain has changed. If a change has occurred, or if for some reason it is not possible to determine whether ownership has changed, further review may be warranted. Note, this remediation does not take into account the case of a Thing that was produced long ago and only recently fielded, or the case where a new MUD manager has been installed.

The release of a MUD URL by a Thing reveals what the Thing is, and provides an attacker with guidance on what vulnerabilities may be present.

While the MUD URL itself is not intended to be unique to a specific Thing, the release of the URL may aid an observer in identifying individuals when combined with other information. This is a privacy consideration.

In addressing both of these concerns, implementors should take into account what other information they are advertising through mechanisms such as mDNS[RFC6872], how a Thing might otherwise be identified, perhaps through how it behaves when it is connected to the network, whether a Thing is intended to be used by individuals or carry personal identifying information, and then apply appropriate data minimization techniques. One approach is to make use of TEAP [RFC7170] as the means to share information with authorized components in the network. Network elements may also assist in limiting access to the MUD URL through the use of mechanisms such as DHCPv6-Shield [RFC7610].

There is the risk of the MUD manager itself being spied on to determine what things are connected to the network. To address this risk, MUD managers may choose to make use of TLS proxies that they trust that would aggregate other information.

Please note that the security considerations mentioned in Section 4.7 of [I-D.ietf-netmod-rfc6087bis] are not applicable in this case because the YANG serialization is not intended to be accessed via NETCONF. However, for those who try to instantiate this model in a network element via NETCONF, all objects in each model in this draft exhibit similar security characteristics as [I-D.ietf-netmod-acl-model]. The basic purpose of MUD is to configure access, and so by its very nature can be disruptive if used by unauthorized parties.

17. IANA Considerations

[There was originally a registry entry for .well-known suffixes. This has been removed from the draft and may be marked as deprecated in the registry. RFC Editor: please remove this comment.]

17.1. YANG Module Registrations

The following YANG modules are requested to be registered in the "IANA Module Names" registry:

The ietf-mud module:

- Name: ietf-mud
- URN: urn:ietf:params:xml:ns:yang:ietf-mud
- Prefix: ietf-mud
- Registrant conact: The IESG
- Reference: [RFCXXXX]

The ietf-acldns module:

- Name: ietf-acldns
- URI: urn:ietf:params:xml:ns:yang:ietf-acldns
- Prefix: ietf-acldns
- Registrant: the IESG
- Reference: [RFCXXXX]

17.2. DHCPv4 and DHCPv6 Options

The IANA has allocated option 161 in the Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) Parameters registry for the MUD DHCPv4 option, and option 112 for DHCPv6, as described in Section 10.

17.3. PKIX Extensions

IANA is kindly requested to make the following assignments for:

- The MUDURLExtnModule-2016 ASN.1 module in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0).
- id-pe-mud-url object identifier from the "SMI Security for PKIX Certificate Extension" registry (1.3.6.1.5.5.7.1).
- id-pe-mudsigner object identifier from the "SMI Security for PKIX Certificate Extension" registry (TBD1).

o id-ct-mudtype object identifier from the "SMI Security for S/MIME CMS Content Type" registry (TBD2).

The use of these values is specified in Section 11.

17.4. MIME Media-type Registration for MUD files

The following media-type is defined for transfer of MUD file:

- o Type name: application
- o Subtype name: mud+json
- o Required parameters: n/a
- o Optional parameters: n/a
- o Encoding considerations: 8bit; application/mud+json values are represented as a JSON object; UTF-8 encoding MUST be employed. [RFC3629]
- o Security considerations: See Security Considerations of RFCXXXX and [RFC8259] Section 12.
- o Interoperability considerations: n/a
- o Published specification: [RFCXXXX]
- o Applications that use this media type: MUD managers as specified by [RFCXXXX].
- o Fragment identifier considerations: n/a
- o Additional information:

  Magic number(s): n/a
  File extension(s): n/a
  Macintosh file type code(s): n/a

- o Person & email address to contact for further information: Eliot Lear <lear@cisco.com>, Ralph Droms <rdroms@gmail.com>
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author:
  Eliot Lear <lear@cisco.com>
  Ralph Droms <rdroms@gmail.com>
- o Change controller: IESG
- o Provisional registration? (standards tree only): No.

17.5. LLDP IANA TLV Subtype Registry

IANA is requested to create a new registry for IANA Link Layer Discovery Protocol (LLDP) TLV subtype values. The recommended policy for this registry is Expert Review. The maximum number of entries in the registry is 256.

IANA is required to populate the initial registry with the value:

LLDP subtype value = 1 (All the other 255 values should be initially marked as 'Unassigned'.)

Description = the Manufacturer Usage Description (MUD) Uniform Resource Locator (URL)

Reference = < this document >

17.6. The MUD Well Known Universal Resource Name (URNs)

The following parameter registry is requested to be added in accordance with [RFC3553]

```
Registry name: "urn:ietf:params:mud" is requested.
Specification: this document
Repository: this document
Index value:  Encoded identically to a TCP/UDP port service
              name, as specified in Section 5.1 of [RFC6335]
```

The following entries should be added to the "urn:ietf:params:mud" name space:

"urn:ietf:params:mud:dns" refers to the service specified by [RFC1123]. "urn:ietf:params:mud:ntp" refers to the service specified by [RFC5905].

17.7. Extensions Registry

The IANA is requested to establish a registry of extensions as follows:

```
Registry name: MUD extensions registry
Registry policy: Standards action
Standard reference: document
Extension name: UTF-8 encoded string, not to exceed 40 characters.
```

Each extension MUST follow the rules specified in this specification. As is usual, the IANA issues early allocations based in accordance with [RFC7120].

18. Acknowledgments

The authors would like to thank Einar Nilsen-Nygaard, who singlehandedly updated the model to match the updated ACL model, Bernie Volz, Tom Gindin, Brian Weis, Sandeep Kumar, Thorsten Dahm, John Bashinski, Steve Rich, Jim Bieda, Dan Wing, Joe Clarke, Henk Birkholz, Adam Montville, Jim Schaad, and Robert Sparks for their valuable advice and reviews. Russ Housley entirely rewrote

Section 11 to be a complete module. Adrian Farrel provided the basis for privacy considerations text. Kent Watsen provided a thorough review of the architecture and the YANG model. The remaining errors in this work are entirely the responsibility of the authors.

19. References

19.1. Normative References

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-19 (work in progress), April 2018.

[IEEE8021AB]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", n.d..

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

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

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[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, <https://www.rfc-editor.org/info/rfc3986>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/info/rfc3987>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/info/rfc5911>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <https://www.rfc-editor.org/info/rfc7227>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <https://www.rfc-editor.org/info/rfc7610>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

19.2. Informative References

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

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20 (work in progress), March 2018.

[IEEE80211i]
Institute for Electrical and Electronics Engineers, "IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11- Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications- Amendment 6- Medium Access Control (MAC) Security Enhancements", 2004.

[IEEE8021AE]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks- Media Access Control (MAC) Security", 2006.

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

[IEEE8021X]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", 2010.

[ISO.8601.1988]
International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, June 1988.

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <https://www.rfc-editor.org/info/rfc3553>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <https://www.rfc-editor.org/info/rfc6092>.

[RFC6872] Gurbani, V., Ed., Burger, E., Ed., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Information Model", RFC 6872, DOI 10.17487/RFC6872, February 2013, <https://www.rfc-editor.org/info/rfc6872>.

[RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <https://www.rfc-editor.org/info/rfc7042>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/info/rfc7170>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <https://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, <https://www.rfc-editor.org/info/rfc7488>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

Appendix A. Changes from Earlier Versions

RFC Editor to remove this section prior to publication.

Draft -19: * Edits after discussion with apps area to address reserved field for the future. * Correct systeminfo to be utf8. * Remove "hardware-rev" from list.

Draft -18: * Correct an error in the augment statement * Changes to the ACL model re ports.

Draft -17:

- One editorial.

Draft -16

- add mud-signature element based on review comments
- redo mud-url
- make clear that systeminfo uses UTF8

Draft -13 to -14:

- Final WGLC comments and review comments
- Move version from MUD-URL to Model
- Have MUD-URL in model
- Update based on update to draft-ietf-netmod-acl-model
- Point to tree diagram draft instead of 6087bis.

Draft -12 to -13:

- Additional WGLC comments

Draft -10 to -12:

These are based on WGLC comments:

- Correct examples based on ACL model changes.
- Change ordering nodes.
- Additional explanatory text around systeminfo.
- Change ordering in examples.
- Make it VERY VERY VERY VERY clear that these are recommendations, not mandates.
- DHCP -> NTP in some of the intro text.
- Remove masa-server
- "Things" to "network elements" in a few key places.
- Reference to JSON YANG RFC added.

Draft -10 to -11:

- Example corrections
- Typo
- Fix two lists.
- Addition of 'any-acl' and 'mud-acl' in the list of allowed features.
- Clarification of what should be in a MUD file.

Draft -09 to -10:

- AD input.
- Correct dates.
- Add compliance sentence as to which ACL module features are implemented.

Draft -08 to -09:

- o Resolution of Security Area review, IoT directorate review, GenART review, YANG doctors review.
- o change of YANG structure to address mandatory nodes.
- o Terminology cleanup.
- o specify out extra portion of MUD-URL.
- o consistency changes.
- o improved YANG descriptions.
- o Remove extra revisions.
- o Track ACL model changes.
- o Additional cautions on use of ACL model; further clarifications on extensions.

Draft -07 to -08:

- o a number of editorials corrected.
- o definition of MUD file tweaked.

Draft -06 to -07:

- o Examples updated.
- o Additional clarification for direction-initiated.
- o Additional implementation guidance given.

Draft -06 to -07:

- o Update models to match new ACL model
- o extract directionality from the ACL, introducing a new device container.

Draft -05 to -06:

- o Make clear that this is a component architecture (Polk and Watson)
- o Add order of operations (Watson)

- Add extensions leaf-list (Pritikin)
- Remove previous-mud-file (Watson)
- Modify text in last-update (Watson)
- Clarify local networks (Weis, Watson)
- Fix contact info (Watson)
- Terminology clarification (Weis)
- Advice on how to handle LDevIDs (Watson)
- Add deployment considerations (Watson)
- Add some additional text about fingerprinting (Watson)
- Appropriate references to 6087bis (Watson)
- Change systeminfo to a URL to be referenced (Lear)

Draft -04 to -05: * syntax error correction

Draft -03 to -04: * Re-add my-controller

Draft -02 to -03: * Additional IANA updates * Format correction in YANG. * Add reference to TEAP.

Draft -01 to -02: * Update IANA considerations * Accept Russ Housley rewrite of X.509 text * Include privacy considerations text * Redo the URL limit. Still 255 bytes, but now stated in the URL definition. * Change URI registration to be under urn:ietf:params

Draft -00 to -01: * Fix cert trust text. * change supportInformation to meta-info * Add an informational element in. * add urn registry and create first entry * add default elements

Appendix B. Default MUD nodes

What follows is the portion of a MUD file that permits DNS traffic to a controller that is registered with the URN "urn:ietf:params:mud:dns" and traffic NTP to a controller that is registered "urn:ietf:params:mud:ntp". This is considered the default behavior and the ACEs are in effect appended to whatever other "ace" entries that a MUD file contains. To block DNS or NTP one repeats the matching statement but replaces the "forwarding" action "accept" with "drop". Because ACEs are processed in the order they are received, the defaults would not be reached. A MUD manager might further decide to optimize to simply not include the defaults when they are overridden.

Four "acl" list entries that implement default MUD nodes are listed below. Two are for IPv4 and two are for IPv6 (one in each direction for both versions of IP). Note that neither access-list name nor ace name need be retained or used in any way by local implementations, but are simply there for completeness' sake.


```
 "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-59776-v4to",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "ent0-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:dns"
                },
                "ipv4": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            },
            {
              "name": "ent1-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:ntp"
                },
                "ipv4": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
```

```
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-59776-v4fr",
          "type": "ipv4-acl-type",
          "aces": {
            "ace": [
              {
                "name": "ent0-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "ipv4": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
                      "operator": "eq",
                      "port": 53
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              },
              {
                "name": "ent1-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv4": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
```

```
                    "operator": "eq",
                    "port": 123
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-59776-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "ent0-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:dns"
                },
                "ipv6": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            },
            {
              "name": "ent1-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:ntp"
                },
                "ipv6": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
```

```
                       "operator": "eq",
                       "port": 123
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         },
         {
           "name": "mud-59776-v6fr",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "ent0-frdev",
                 "matches": {
                   "ietf-mud:mud": {
                     "controller": "urn:ietf:params:mud:dns"
                   },
                   "ipv6": {
                     "protocol": 17
                   },
                   "udp": {
                     "destination-port": {
                       "operator": "eq",
                       "port": 53
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               },
               {
                 "name": "ent1-frdev",
                 "matches": {
                   "ietf-mud:mud": {
                     "controller": "urn:ietf:params:mud:ntp"
                   },
                   "ipv6": {
                     "protocol": 17
                   },
                   "udp": {
                     "destination-port": {
```

```
                       "operator": "eq",
                       "port": 123
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         }
       ]
     }
```


Appendix C. A Sample Extension: DETNET-indicator

In this sample extension we augment the core MUD model to indicate whether the device implements DETNET. If a device claims not to use DETNET, but then later attempts to do so, a notification or exception might be generated. Note that this example is intended only for illustrative purposes.

Extension Name: "Example-Extension" (to be used in the extensions list)
Standard: this document (but do not register the example)

This extension augments the MUD model to include a single node, using the following sample module that has the following tree structure:

```
module: ietf-mud-detext-example
  augment /ietf-mud:mud:
    +--rw is-detnet-required?   boolean
```

The model is defined as follows:

```
<CODE BEGINS>file "ietf-mud-detext-example@2018-06-15.yang"
module ietf-mud-detext-example {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-detext-example";
  prefix ietf-mud-detext-example;

  import ietf-mud {
    prefix ietf-mud;
  }
```

```
     organization
       "IETF OPSAWG (Ops Area) Working Group";
     contact
       "WG Web: http://tools.ietf.org/wg/opsawg/
        WG List: opsawg@ietf.org
        Author: Eliot Lear
        lear@cisco.com
        Author: Ralph Droms
        rdroms@gmail.com
        Author: Dan Romascanu
        dromasca@gmail.com

       ";
     description
       "Sample extension to a MUD module to indicate a need
        for DETNET support.";

     revision 2018-06-15 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Manufacturer Usage Description
          Specification";
     }

     augment "/ietf-mud:mud" {
       description
         "This adds a simple extension for a manufacturer
           to indicate whether DETNET is required by a
          device.";
       leaf is-detnet-required {
         type boolean;
         description
           "This value will equal true if a device requires
            detnet to properly function";
       }
     }
   }
   <CODE ENDS>
```

Using the previous example, we now show how the extension would be expressed:


```
   {
     "ietf-mud:mud": {
       "mud-version": 1,
       "mud-url": "https://lighting.example.com/lightbulb2000",
       "last-update": "2018-03-02T11:20:51+01:00",
```

```
       "cache-validity": 48,
       "extensions": [
           "ietf-mud-detext-example"
        ],
       "ietf-mud-detext-example:is-detnet-required": "false",
       "is-supported": true,
       "systeminfo": "The BMS Example Lightbulb",
       "from-device-policy": {
         "access-lists": {
           "access-list": [
             {
               "name": "mud-76100-v6fr"
             }
           ]
         }
       },
       "to-device-policy": {
         "access-lists": {
           "access-list": [
             {
               "name": "mud-76100-v6to"
             }
           ]
         }
       }
     },
     "ietf-access-control-list:acls": {
       "acl": [
         {
           "name": "mud-76100-v6to",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "cl0-todev",
                 "matches": {
                   "ipv6": {
                     "ietf-acldns:src-dnsname": "test.example.com",
                     "protocol": 6
                   },
                   "tcp": {
                     "ietf-mud:direction-initiated": "from-device",
                     "source-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   }
                 },
```

```
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         },
         {
           "name": "mud-76100-v6fr",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "cl0-frdev",
                 "matches": {
                   "ipv6": {
                     "ietf-acldns:dst-dnsname": "test.example.com",
                     "protocol": 6
                   },
                   "tcp": {
                     "ietf-mud:direction-initiated": "from-device",
                     "destination-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         }
       ]
     }
   }
```


Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

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

Ralph Droms
Google
355 Main St., 5th Floor
Cambridge

Phone: +1 978 376 3731
Email: rdroms@gmail.com

Dan Romascanu

Phone: +972 54 5555347
Email: dromasca@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 30, 2018

C. Li
C. Xie
China Telecom
R. Kumar
R. Lohiya
Juniper Networks
G. Fioccola
Telecom Italia
W. Xu
W. Liu
Huawei Technologies
D. Ma
ZDNS
J. Bi
Tsinghua University
June 28, 2018

Coordinated Address Space Management architecture
draft-li-opsawg-address-pool-management-arch-01

Abstract

IP addresses work as a basic element for providing broadband network services. However, the increase in number, diversity and complexity of modern network devices and services creates unprecedented challenges for the currently prevailing approach of manual IP address management. Manually maintaining IP addresses could always be sub-optimal for IP resource utilization. Besides, it requires heavy human effort from network operators. To achieve high utilization and flexible scheduling of IP network addresses, it is necessary to automate the address scheduling process. This document describes an architecture for the IP address space management. It includes architectural concepts and components used in the CASM (Coordinated Address Space Management), with a focus on those interfaces to be standardized in the IETF.

Status of This Memo

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 December 30, 2018.

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 4
3. CASM Reference architecture 4
4. The overall procedure of CASM 7
5. CASM Interface and operation 8
5.1. CASM App-facing Interface 8
5.1.1. Functional requirements 8
5.1.2. Interface modeling requirements 9
5.2. CASM device-facing Interface 9
5.2.1. Functional requirements 10
5.2.2. Interface modeling requirements/Initial Address Pool Configuration 10
5.2.3. Interface modeling requirements/Address Pool Status Report 12
5.2.4. Interface modeling requirements/Address Pool Status Query 13
5.2.5. Interface modeling requirements/Address Exhaustion . 13
5.2.6. Interface modeling requirements / Address Pool Release 14
6. Services SDN Management Use Cases 15
7. Security Considerations 16
8. Acknowledgements 16
9. References 16
9.1. Normative References 16

9.2. Informative References 17
Authors' Addresses 17

1. Introduction

The address space management is an integral part of any network management solution. However, the increase in number, diversity and complexity of modern network devices and services creates unprecedented challenges for the currently prevailing approach of manual IP address management. Manually maintaining IP addresses could always be sub-optimal for IP resource utilization. Besides, it requires heavy human effort from network operators.

Another factor which drive this work is that tThe network architectures are rapidly changing with the migration toward private and public clouds. At the same time, application architectures are also evolving with a shift toward micro-services and multi-tiered approach.

There is a pressing need to define a new address management system which can meet these diverse set of requirements. To achieve high utilization and flexible scheduling of IP network addresses, Such a system should be capable of automating the address scheduling process. Such a system must be built with well-defined interfaces so users can easily migrate from one vendor to another without rewriting their network management systems.

This document defines a reference architecture that should become the basis to develop a new address management system. This system is called Coodinated Address Space Management (CSAM) system.

A series of use cases are defined in "Use Case Draft". For example, Broadband Network Gateway (BNG), which manages a routable IP address on behalf of each subscriber, should be configured with the IP address pools allocated to subscribers. However, currently operators are facing with the address shortage problem, the remaining IPv4 address pools are usually quite scattered, no more than /24 per address pool in many cases. Therefore, it is complicated to manually configure the address pools on lots of Broadband Network Gateway (BNG) for operators. For large scale Metro Area Network (MAN), the number of BNGs can be up to over one hundred. Manual configuration on all the BNGs statically will not only greatly increase the workload, but also decrease the utilization efficiency of the address pools when the number of subscribers changes over time in the future.

Above is one example of use case, there are other devices which may need to configure address pools as well. In this document, we propose a general mechanism to manage the address pools coordinately, which can be used in multiple use cases. With this approach, operators do not need to configure the address pools one by one manually and it also helps to use the address pools more efficiently.

2. Terminology

The following terms are used in this document:

CASM: Coordinated Address Space Management, a newly-defined general architecture which can automate IP address management for wide-variety of use cases

IPAM: IP Address Management, a means of planning, tracking, and managing the Internet Protocol address space used in a network

DA: A device agent within the device, which contacts with CASM Coordinator to manipulate address pool

CASM Coordinator: A management system which has a database manage the overall address pools and allocate address pools to devices.

3. CASM Reference architecture

The figure below shows the reference architecture for CASM. This figure covers the various possible scenarios that can exist in future network.

```
          +-------------+       +-------------+       +-------------+
          |     CASM    |       |     CASM    |       |     CASM    |
          |application 1|       |application j|       |application n|
          +------/------+       +------/------+       +------/------+
                 |                     |                     |
                 |                     |                     |
                 |                     |                     |
                 |                     |                     |
         +-------\---------------------\---------------------\-------+
         |    Coordinated Address Space Management System (CASM)     |
         |                       Coordinator                         |
         |    +-------------+  +-------------+  +-------------+      |
         |    |     Pool    |  |   Address   |  |   Address   |      |
         |    |  Management |  | Management  |  |  Database   |      |
         |    +-------------+  +-------------+  +-------------+      |
         |                                                           |
         +---.-------------------------.--------------------------.--+
             |                         |                          |
             |                         |                          |
             |                         |                          |
             |                         |                          |
             |                         |                          |
  +----------\--------+      +---------\---------+       +--------\----------+
  |                   |      |                   |       |                   |
  |  +-------------+  |      |  +-------------+  |       |  +-------------+  |
  |  |      DA     |  |      |  |     DA      |  |       |  |     DA      |  |
  |  +-------------+  |      |  +-------------+  |       |  +-------------+  |
  |                   |      |                   |       |                   |
  |  +-------------+  |      |  +-------------+  |       |  +-------------+  |
  |  |     CASM    |  |      |  |     CASM    |  |       |  |     CASM    |  |
  |  | Distributor |  |      |  | Distributor |  |       |  | Distributor |  |
  |  +-------------+  |      |  +-------------+  |       |  +-------------+  |
  |      Device 1     |      |      Device 2     |       |      Device m     |
  +-------------------+      +-------------------+       +-------------------+
```

Figure 1: CASM reference architecture

Each component of CASM is introduced as below,

1) CASM Application

The CASM Application is a functional entity which usually has the requirements of centralized address management to realize its specific upper-layer functions. In order to achieve this goal, it needs to manage, operate and maintain the CASM Coordinator. For example, an operator or external user can manage the address pool in the CASM Coordinator, as well as access log, address allocation records, etc.

2) CASM Coordinator

The CASM Coordinator is a coordinated address management coordinator for the CASM Application to maintain overall address pools, addresses, address properties, etc. It maintains an address database including the overall address pools (OAP) and the address pool status (APS). CASM Applications can maintain their remaining address pools in the OAP. They can also reserve some address pools for special purposes. The address pool status is to reflect the current usage of address pools for different devices. The CASM Coordinator also has the capability to maintain the address pools to different devices dynamically.

3) CASM Device

A CASM Device is responsible for distributing or allocating addresses from local address pools received from the CASM Coordinator. CASM has two components in devices. The first one is Device Agent (DA), which resides in a CASM Device through which the device can contact with the CASM Coordinator. On behalf of the device, the agent initiates the address pool allocation requests, passes the address pools to local instances, detect the availability of address pools or report the status of local address pool usage and update the address pool requests, etc. For some devices, e.g. IPv6 transition and VPN, additional routing modules are needed to update the routing table accordingly.

The CASM Distributor is another component in a CASM device. The DHCP server is a typical distributor that can assign IP addresses to client hosts, and the DHCP protocol is usually used for this task. The address assignment procedure between the CASM Distributor and the client host is out of the scope of this document.

The device determines whether the usage status of the IP address pool resource within the device satisfies the condition. When the IP address pool resource in the device is insufficient or excessive, the device will obtain IP address pool resource request, and sends the request to the CASM Coordinator. The device receives a resource response with IP address pools allocated for it, then it use these address pools to assign IP addresses to end users. Typical CASM Devices include BNGs, BRASes, CGNs, DHCP Servers, NATs, IPv6 Transitions, DNS Servers, etc.

The form of devices is diverse, it can be physical or virtual, and it can be box-integrated with a control plane and a user plane, or a separated control plane remote from the box, where one or more devices share the centralized control plane. In the latter case, the control plane will manage multiple user plane devices. A number of devices that are subordinate to the control plane will jointly share the address pools to make address utilization much higher.

4. The overall procedure of CASM

1. Operators configure remaining address pools centrally in the CASM Coordinator. There are multiple address pools that can be configured. The CASM Coordinator server then divides the address pools into addressing units (AUs) which would be allocated to device agents by default.

2. The agent will initiate an AddressPool request to the CASM Coordinator. It can carry its desired size of address pool with the request, or just use a default value. The address pool size in the request is only used as a hint. The actual size of the address pool is totally determined by the CASM Coordinator. It would also carry the DA's identification and the type of the address pool.

3. The CASM Coordinator looks up remaining address pools in its local database, and then allocates a set of address pools to the DA. Each address pool has a lifetime.

4. The DA receives the AddressPool reply and uses it for its purpose.

5. If the lifetime of the address pool is going to expire, the DA should issue an AddressPoolRenew request to extend it, including IPv4, IPv6, port numbers, etc.

6. The AddressPoolReport module keeps monitoring and reports the usage of all current address pools for each transition mechanism. If it is running out of address pools, it can renew the AddressPoolRequest for a newly allocated one. It can also release and recycle an existing address pool if that address pool has not been used for a specific and configurable time.

7. When the connection of the CASM Coordinator is lost or it needs the status information of certain applications, it may pre-actively query the DA for its status information.

Currently, the CASM system focuses on the coordination of IP address resources. This Solution should be extended to handle containers, VLAN assignments, etc. These are subject for future work.

5. CASM Interface and operation

5.1. CASM App-facing Interface

The CASM architecture consists of three major distinct entities: CASM Application, CASM Coordinator and network device with a device Agent (DA). In order to provide address space and pools resource that CASM Coordinator can centrally maintain, there is an interface between CASM Applications and CASM Coordinator. The CASM Application can manage the address space and pool in the CASM Coordinator, and the get address allocation records, logs from CASM Coordinator.

5.1.1. Functional requirements

The CASM should support following functionality for it to be adopted for wide variety of use cases.

1. Address pools requirements

A CASM system should allow ability to manage different kind of address pools. The following pools should be considered for implementation; this is not mandatory or exhaustive by any means but given here as most commonly used in networks. The CASM system should allow user-defined pools with any address objects.

Unicast address pool:

- Private IPv4 addresses
- Public IPv4 addresses
- IPv6 addresses
- MAC Addresses

Multicast address pool:

- IPv4 address
- IPv6 address

2. Pool management requirements

There should be a rich set of functionality as defined in this section for operation of a given pool.

Address management:

- o Address allocation either as single or block
- o Address reservation
- o Allocation logic such as mapping schemes or algorithm per pool
- o

General management:

- o Pool initializing, resizing, threshold markings for resource monitoring
- o Pool attributes such as used to automatically create DNS record
- o Pool priority for searching across different pools
- o Pool fragmentation rules, such as how pool can be sub-divided
- o Pool lease rules for allocation requests

5.1.2. Interface modeling requirements

There are three broad categories for CASM interface definition:

Pool management interface: Interface to external user or applications such as SDN controller to manage addresses

Log interface: Interface to access log and records such as DHCP, DNS, NAT Integration interface: Interface to address services such as DHCP, DNS, NAT

5.2. CASM device-facing Interface

In order to provide address pool manipulations between CASM Coordinator and device, the CASM architecture calls for well-defined protocols for interfacing between them. Protocol such as radius can be used to compatible with legacy network equipment. And in more modern network system, network device acts as NETCONF/RESTCONF server side, device like CASM Coordinator act as client side. The network device sends address pool request message carrying the requested resource information to the CASM Coordinator, the CASM Coordinator send response message to the network device, where the response message includes address pool resource information allocated to the network device, and network device receives the response message and retrieve the allocated address pool resource information carried in the response message.

5.2.1. Functional requirements

In order to build a complete address management system, it is important that CASM should be able to integrate with other address services. This will provide a complete solution to network operators without requiring any manual or proprietary workflows.

DHCP server:

- Interface to initialize address pools on DHCP server

- Notification interface whenever an address lease is modified

- Interface to access address lease records from DHCP server

- Ability to store lease records and play back to DHCP server on reboot

DNS server:

- Interface to create DNS records on DNS server based on DHCP server events

NAT device:

- Interface to initialize NAT pools

- Interface to access NAT records from NAT device

- Ability to store NAT records and play back to NAT device on reboot

5.2.2. Interface modeling requirements/Initial Address Pool Configuration

```
          +--------------+                           +-----------------+
          |   Device     |                           |       CASM      |
          |   Agent      |                           |   Coordinator   |
          +------+-------+                           +--------+--------+
                 |                                            |
        +--------+-------+                                    |
        |1.DA start-up   |                                    |
        +---------+------+                                    |
                  |            2.Address Pool Request         |
                  |------------------------------------------>|
                  |                                           |
                  |                                  +--------+-------+
                  |                                  |  3. Check      |
                  |                                  |   address pool |
                  |                                  +--------+-------+
                  |          4.Address Pool Reply             |
                  |<------------------------------------------|
                  |                                           |
```

Figure 2: Initial Address Pool Configuration

As shown in Figure 2, the procedure is as follows:

1. The DA checks whether there is already address pool configured in the local site when it starts up.

2. The DA will initiate Address Pool request to the CASM Coordinator. It can carry its desired size of address pool in the request, or just use a default value. The address pool size in the DA’s request is only used as a hint. The actual size of the address pool is totally determined by CASM Coordinator. It will also carry the DA’s identification, the type of transition mechanism and the indication of port allocation support.

3. The CASM Coordinator determines the address pool allocated for the DA based on the parameters received.

4. The CASM Coordinator sends the Address Pool Reply to the DA. It will also distribute the routing entry of the address pool automatically. In particular, if the newly received address pool can be aggregated to an existing one, the routing should be aggregated accordingly.

5.2.3. Interface modeling requirements/Address Pool Status Report

```
        +--------------+                             +-----------------+
        |   Device     |                             |      CASM       |
        |   Agent      |                             |   Coordinator   |
        +------+-------+                             +--------+--------+
               |                                              |
      +--------+-------+                                      |
      |1.Monitor and   |                                      |
      |count the status|                                      |
      +--------+-------+                                      |
               |        2.Address Pool Status Report          |
               |--------------------------------------------->|
               |                                     +--------+-------+
               |                                     |  3. Record     |
               |                                     |   address pool |
               |                                     +--------+-------+
               |       4.Address Pool Report Confirm          |
               |<---------------------------------------------|
               |                                              |
               |                                              |
```

Figure 3: Address Pool Status Report

Figure 3 illustrates the active address pool status report procedure:

1. The DA will monitor and count the usage status of the local address pool. The DA counts the address usage status in one month, one week and one day, which includes the local address, address usage ratio (peak and average values), and the port usage ratio (peak and average values).

2. The DA reports the address pool usage status to the CASM Coordinator. For example, it will report the address usage status in one day, which contains the IP address, NAT44, address list: 30.14.44.0/28, peak address value 14, average address usage ratio 90%, TCP port usage ratio 20%, UDP port usage ratio 30% and etc.

3. The CASM Coordinator records the status and compares with the existing address information to determine whether additional address pool is needed.

4. The CASM Coordinator will confirm the address pool status report request to the DA. It will keep sending the address pool status

report request to the CASM Coordinator if no confirm message is received.

5.2.4. Interface modeling requirements/Address Pool Status Query

When the status of CASM Coordinator is lost or the CASM Coordinator needs the status information of the DAs, the CASM Coordinator may actively query the TD for the status information, as shown in step 1 of Figure 4. The following steps 2,3,4,5 are the same as the Address Pool Status Report procedure.

```
        +--------------+                             +-----------------+
        |   Device     |                             |      CASM       |
        |   Agent      |                             |   Coordinator   |
        +------+-------+                             +--------+--------+
               |                                              |
               |                                              |
               |         1.Address Pool Status Query          |
               |<---------------------------------------------|
               |                                              |
      +--------+-------+                                      |
      |2.Monitor and   |                                      |
      |count the status|                                      |
      +--------+-------+                                      |
               |        3.Address Pool Status Report          |
               |--------------------------------------------->|
               |                                     +--------+-------+
               |                                     |  4. Record     |
               |                                     |   address pool |
               |                                     +--------+-------+
               |       5.Address Pool Report Confirm          |
               |<---------------------------------------------|
               |                                              |
               |                                              |
```

Figure 4: Address Pool Status Query

5.2.5. Interface modeling requirements/Address Exhaustion

When the addresses used by the DA reaches a certain usage threshold, the DA will renew the address pool request to the CASM Coordinator for an additional address pool. The procedure is the same as the initial address pool request.

5.2.6. Interface modeling requirements / Address Pool Release

```
        +--------------+                             +-----------------+
        |   Device     |                             |       CASM      |
        |   Agent      |                             |    Coordinator  |
        +------+-------+                             +--------+--------+
               |                                              |
      +--------+-------+                                      |
      |1.Address pools |                                      |
      |  not used for a|                                      |
      |   long time    |                                      |
      +--------+-------+                                      |
               |        2.Address Pool Release Request        |
               |--------------------------------------------->|
               |                                     +--------+-------+
               |                                     |3. Update       |
               |                                     |   address pool |
               |                                     |   database     |
               |                                     +--------+-------+
               |       4.Address Pool Release Notification    |
               |<---------------------------------------------|
      +--------+-------+                                      |
      |5. Reduce       |                                      |
      |  address pool  |                                      |
      +--------+-------+                                      |
               |         6.Address Pool Release Confirm       |
               |--------------------------------------------->|
               |                                              |
               |                                              |
```

Figure 5: Address Pool Release

Figure 5 illustrates the address pool release procedure:

1. The counting module in the DA checks if the usage threshold of address pool reaches a certain condition;

2. The DA sends the address pool release request to the CASM Coordinator to ask the release of those addresses;

3. The CASM Coordinator updates the local address pool information to add the new addressed released;

4. The CASM Coordinator notifies the TD that the addresses have been release successfully;

5. The DA will update the local address pool. If no Address Pool Release Notification is received, the DA will repeat step 2;

6. Optionally, the DA confirms with the CASM Coordinator that the address pool has been released successfully.

6. Services SDN Management Use Cases

```
                          -------------
                         |    CASM     |
                         | Application |
                          -------------
                                :
                       ------------------
                      |     Provider     |
                      |   Orchestrator   |
                      |                  |
                      .------------------.
                     .          :         .
                    .           :          .
           ------------     ------------     ------------
          |            |   |            |   |            |
          | Controller |   | Controller |   | Controller |
          |            |   |            |   |            |
           ------------     ------------     ------------
               :              .       .               :
               :             .         .              :
               :            .           .             :
          ---------      ---------   ---------     ---------
         | Network |    | Network | | Network |   | Network |
         | Element |    | Element | | Element |   | Element |
          ---------      ---------   ---------     ---------
```

Figure 6: L3 and L2 Services Orchestration

Network Operators need to manage addressing of undelay network elements in order to build end-to-end services and private or public clouds. So address management of customer equipments, provider edges, but also of virtual machines, virtual functions and overlay networks is a very important task. In general the SDN Orchestrators and other management systems must coordinate addressing schemes to ensure network operation. There is need for one address management system that would meet the requirements of such a network deployment. The SDN Orchestrator manages IPv4, IPv6 addresses and also MAC addresses to assign to network interfaces in order to install end-to-end services, and this task can be achieved by the CASM coordination.

A typical use case is the application to the Service provisioning of L3VPN and L2VPN by the SDN orchestration level. For example the architecture presented in [RFC8309] and, more in general in every SDN architecture, could be integrated with CASM. It is important to mention also the possibility of Multi-Provider services, and in this case the two CASM coordinators of the two involved Providers should synchronize. The following Figure shows how CASM Application can communicate with both the Network Operator Orchestrator and, in case of Multi-Provider Service, with another Network Operator Orchestrator too.

7. Security Considerations

8. Acknowledgements

N/A.

9. References

9.1. Normative References

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <https://www.rfc-editor.org/info/rfc2132>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <https://www.rfc-editor.org/info/rfc6888>.

Authors' Addresses

Chen Li
China Telecom
No.118 Xizhimennei street, Xicheng District
Beijing 100035
P.R. China

Email: lichen@ctbri.com.cn

Chongfeng Xie
China Telecom
No.118 Xizhimennei street, Xicheng District
Beijing 100035
P.R. China

Email: xiechf.bri@chinatelecom.cn

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale CA 94089
US

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale CA 94089
US

Email: alohiya@juniper.net

Giuseppe Fioccola
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

Weiping Xu
Huawei Technologies
Bantian, Longgang District
shenzhen 518129
P.R. China

Email: xuweiping@huawei.com

Will(Shucheng) Liu
Huawei Technologies
Bantian, Longgang District
shenzhen 518129
P.R. China

Email: liushucheng@huawei.com

Di Ma
ZDNS
4 South 4th St. Zhongguancun
Beijing 100190
P.R. China

Email: madi@zdns.cn

Jun Bi
Tsinghua University
3-212, FIT Building, Tsinghua University, Haidian District
Beijing 100084
P.R. China

Email: junbi@tsinghua.edu.cn

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

H. Song, Ed.
T. Zhou
ZB. Li
Huawei
G. Fioccola
Telecom Italia
ZQ. Li
China Mobile
P. Martinez-Julia
NICT
L. Ciavaglia
Nokia
A. Wang
China Telecom
July 2, 2018

Toward a Network Telemetry Framework
draft-song-ntf-02

Abstract

This document suggests the necessity of an architectural framework for network telemetry in order to meet the current and future network operation requirements. The defining characteristics of network telemetry shows a clear distinction from the conventional network OAM concept; hence the network telemetry demands new techniques and protocols. This document clarifies the terminologies and classifies the categories and components of a network telemetry framework. The requirements, challenges, existing solutions, and future directions are discussed for each category. The network telemetry framework and the taxonomy help to set a common ground for the collection of related works and put future technique and standard developments into perspective.

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].

Status of This Memo

Copyright Notice

Table of Contents

1. Motivation 3
1.1. Use Cases 3
1.2. Challenges 5
1.3. Glossary 5
1.4. Network Telemetry 6
2. The Necessity of a Network Telemetry Framework 8
3. Network Telemetry Framework 9
3.1. Existing Works Mapped in the Framework 11
3.2. Management Plane Telemetry 12
3.2.1. Requirements and Challenges 12
3.2.2. Push Extensions for NETCONF 13
3.2.3. gRPC Network Management Interface 13
3.3. Control Plane Telemetry 14
3.3.1. Requirements and Challenges 14
3.3.2. BGP Monitoring Protocol 14
3.4. Data Plane Telemetry 15
3.4.1. Requirements and Challenges 15
3.4.2. Technique Classification 16
3.4.3. The IPFPM technology 16
3.4.4. Dynamic Network Probe 18
3.4.5. IP Flow Information Export (IPFIX) protocol 18

3.4.6. In-Situ OAM 18
3.5. External Data and Event Telemetry 19
3.5.1. Requirements and Challenges 19
4. Security Considerations 20
5. IANA Considerations 20
6. Contributors 20
7. Acknowledgments 20
8. References 20
8.1. Normative References 20
8.2. Informative References 20
Authors' Addresses 23

1. Motivation

The advance of AI/ML technologies gives networks an unprecedented opportunity to realize network autonomy with closed control loops. An intent-driven autonomous network is the logical next step for network evolution following SDN, aiming to reduce (or even eliminate) human labor, make the most efficient use of network resources, and provide better services more aligned with customer requirements. Although we still have a long way to reach the ultimate goal, the journey has started nevertheless.

The storage and computing technologies are already mature enough to be able to retain and process a huge amount of data and make real-time inference. Tools based on machine learning technologies and big data analytics are powerful in detecting and reacting on network faults, anomalies, and policy violations. In turn, the network policy updates for planning, intrusion prevention, optimization, and self-healing can be applied. Some tools can even predict future events based on historical data.

However, the networks fail to keep pace with such data need. The current network architecture, protocol suite, and system design are not ready yet to provide enough quality data. In the remaining of this section, first we identify a few key network operation use cases that network operators need the most. These use cases are also the essential functions of the future autonomous networks. Next, we show why the current network OAM techniques and protocols are not sufficient to meet the requirements of these use cases. The discussion underlines the need of a new brood of techniques and protocols which we put under an umbrella term - network telemetry.

1.1. Use Cases

All these use cases involves the data extracted from the network data plane and sometimes from the network control plane and management plane.

Intent and Policy Compliance: Network policies are the rules that constraint the services for network access, provide differentiate within a service, or enforce specific treatment on the traffic. For example, a service function chain is a policy that requires the selected flows to pass through a set of network functions in order. An intents is a high-level abstract policy which requires a complex translation and mapping process before being applied on networks. While a policy is enforced, the compliance needs to be verified and monitored continuously.

SLA Compliance: A Service-Level Agreement (SLA) defines the level of service a user expects from a network operator, which include the metrics for the service measurement and remedy/penalty procedures when the service level misses the agreement. Users need to check if they get the service as promised and network operators need to evaluate how they can deliver the services that can meet the SLA.

Root Cause Analysis: Network failure often involves a sequence of chained events and the source of the failure is not straightforward to identify, especially when the failure is sporadic. While machine learning or other data analytics technologies can be used for root cause analysis, it up to the network to provide all the relevant data for analysis.

Load Balancing, Traffic Engineering, and Network Planning: Network operators are motivated to optimize their network utilization for better ROI or lower CAPEX, as well as differentiation across services and/or users of a given service. The first step is to know the real-time network conditions before applying policies to steer the user traffic or adjust the load balancing algorithm. In some cases network micro-bursts need to be detected in a very short time-frame so that fine grained traffic control can be applied to avoid possible network congestion. The long term network capacity planning and topology augmentation also rely on the accumulated data of the network operation.

Event Tracking and Prediction: Network visibility is critical for a healthy network operation. Numerous network events are of interest to network operators. For example, Network operators always want to learn where and why packets are dropped for an application flow. They also want to be warned by some early signs that some component is going to fail so the proper fix or replacement can be made in time.

1.2. Challenges

The conventional OAM techniques, as described in [RFC7276], are not sufficient to support the above use cases for the following reasons:

- o Most use cases need to continuously monitor the network and dynamically refine the data collection in real-time and interactively. The poll-based low-frequency data collection is ill-suited for these applications. Streaming data directly pushed from the data source is preferred.

- o Various data is needed from any place ranging from the packet processing engine to the QoS traffic manager. Traditional data plane devices cannot provide the necessary probes. An open and programmable data plane is therefore needed.

- o Many application scenarios need to correlate data from multiple sources (e.g., from distributed nodes or from different network plane). A piecemeal solution is often lacking the capability to consolidate the data from multiple sources. The composition of a complete solution, as partly proposed by ARCA [I-D.pedro-nmrg-anticipated-adaptation], will be empowered and guided by a comprehensive framework.

- o The passive measurement techniques can either consume too much network resources and render too much redundant data, or lead to inaccurate results. The active measurement techniques are indirect, and they can interfere with the user traffic. We need techniques that can collect direct and on-demand data from user traffic.

1.3. Glossary

Before further discussion, we list some key terminology and acronyms used in this documents. We make an intended distinction between network telemetry and network OAM.

AI: Artificial Intelligence. Use machine-learning based technologies to automate network operation.

BMP: BGP Monitoring Protocol

DNP: Dynamic Network Probe

DPI: Deep Packet Inspection

gNMI: gPRC Network Management Interface

gRPC: gRPC Remote Procedure Call

IDN: Intent-Driven Network

IPFIX: IP Flow Information Export Protocol

IPFPM: IP Flow Performance Measurement

IOAM: In-situ OAM

NETCONF: Network Configuration Protocol

Network Telemetry: A general term for a new brood of network visibility techniques and protocols, with the characteristics defined in this document. Network telemetry enables smooth evolution toward intent-driven autonomous networks.

NMS: Network Management System

OAM: Operations, Administration, and Maintenance. A group of network management functions that provide network fault indication, fault localization, performance information, and data and diagnosis functions. Most conventional network monitoring techniques and protocols belong to network OAM.

SNMP: Simple Network Management Protocol

YANG: A data modeling language for NETCONF

YANG FSM: A YANG model to define device side finite state machine

YANG PUSH: A method to subscribe pushed data from remote YANG datastore

1.4. Network Telemetry

For a long time, network operators have relied upon protocols such as SNMP [RFC1157] to monitor the network. SNMP can only provide limited information about the network. Since SNMP is poll-based, it incurs low data rate and high processing overhead. Such drawbacks make SNMP unsuitable for today’s automatic network applications.

Network telemetry has emerged as a mainstream technical term to refer to the newer techniques of data collection and consumption, distinguishing itself form the convention techniques for network OAM. It is expected that network telemetry can provide the necessary network visibility for autonomous networks, address the shortcomings of conventional OAM techniques, and allow for the emergence of new techniques bearing certain characterisitcs.

One key difference between the network telemetry and the network OAM is that the network telemetry assumes an intelligent machine in the center of a closed control loop, while the network OAM assumes the human network operators in the middle of an open control loop. The network telemetry can directly trigger the automated network operation; The conventional OAM tools only help human operators to monitor and diagnose the networks and guide manual network operations. The different assumptions lead to very different techniques.

Although the network telemetry techniques are just emerging and subject to continuous evolution, several defining characteristics of network telemetry have been well accepted:

- o Push and Streaming: Instead of polling data from network devices, the telemetry collector subscribes to the streaming data pushed from the data source in network devices.

- o Volume and Velocity: The telemetry data is intended to be consumed by machine rather than by human. Therefore, the data volume is huge and the processing is often in realtime.

- o Normalization and Unification: Telemetry aims to address the overall network automation needs. The piecemeal solutions offered by the conventional OAM approach are no longer suitable. Efforts need to be made to normalize the data representation and unify the protocols.

- o Model-based: The data is model-based which allows applications to configure and consume data with ease.

- o Data Fusion: The data for a single application can come from multiple data sources (e.g., cross domain, cross device, and cross layer) and needs to be correlated to take effect.

- o Dynamic and Interactive: Since the network telemetry means to be used in a closed control loop for network automation, it needs to run continuously and adapt to the dynamic and interactive queries from the network operation controller.

In addition, the ideal network telemetry solution should also support the following features:

- o In-Network Customization: The data can be customized in network at run-time to cater to the specific need of applications. This needs the support of a programmable data plane which allows probes to be deployed at flexible locations.

- Direct Data Plane Export: The data originated from data plane can be directly exported to the data consumer for efficiency, especially when the data bandwidth is large and the real-time processing is required.

- In-band Data Collection: In addition to the passive and active data collection approaches, the new hybrid approach allows to directly collect data for any target flow on its entire forwarding path.

- Non-intrusive: The telemetry system should not fall into the trap of the "observer effect". That is, it should not change the network behavior or affect the forwarding performance.

2. The Necessity of a Network Telemetry Framework

Big data analytics and machine-learning based AI technologies are applied for network operation automation, relying on abundant data from networks. The single-sourced and static data acquisition cannot meet the data requirements. It is desirable to have a framework that integrates multiple telemetry approaches from different layers, and allows flexible combinations for different applications. The framework will benefit application development for the following reasons.

- The future autonomous networks will require a holistic view on network visibility. All the use cases and applications need to be supported uniformly and coherently under a single intelligent agent. Therefore, the protocols and mechanisms should be consolidated into a minimum yet comprehensive set. A telemetry framework can help to normalize the technique developments.

- Network visibility presents multiple viewpoints. For example, the device viewpoint takes the network infrastructure as the monitoring object from which the network topology and device status can be acquired; the traffic viewpoint takes the flows or packets as the monitoring object from which the traffic quality and path can be acquired. An application may need to switch its viewpoint during operation. It may also need to correlate a service and its network experience to acquire the comprehensive information.

- Applications require network telemetry to be elastic in order to efficiently use the network resource and reduce the performance impact. Routine network monitoring covers the entire network with low data sampling rate. When issues arise or trends emerge, the telemetry data source can be modified and the data rate can be boosted.

- Efficient data fusion is critical for applications to reduce the overall quantity of data and improve the accuracy of analysis.

So far, some telemetry related work has been done within IETF. However, this work is fragmented and scattered in different working groups. The lack of coherence makes it difficult to assemble a comprehensive network telemetry system and causes repetitive and redundant work.

A formal network telemetry framework is needed for constructing a working system. The framework should cover the concepts and components from the standardization perspective. This document clarifies the layers on which the telemetry is exerted and decomposes the telemetry system into a set of distinct components that the existing and future work can easily map to.

3. Network Telemetry Framework

Telemetry can be applied on the data plane, the control plane, and the management plane in a network, as well as other sources out of the network, as shown in Figure 1.

```
+------------------------------+
|                              |
|       Network Operation      |<-------+
|          Applications        |        |
|                              |        |
+------------------------------+        |
     ^      ^           ^               |
     |      |           |               |
     V      |           V               V
+-----------|---+--------------+  +-----------+
|           |   |              |  |           |
| Control Pl|ane|              |  | External  |
| Telemetry | <--->            |  | Data and  |
|           |   |              |  | Event     |
|      ^    V   |  Management  |  | Telemetry |
+------|--------+  Plane       |  |           |
|      V        |  Telemetry   |  +-----------+
|               |              |
| Data Plane  <--->            |
| Telemetry     |              |
|               |              |
+---------------+--------------+
```

Figure 1: Layer Category of the Network Telemetry Framework

Note that the interaction with the network operation applications can be indirect. For example, in the management plane telemetry, the management plane may need to acquire data from the data plane. On the other hand, an application may involve more than one plane simultaneously. For example, an SLA compliance application may require both the data plane telemetry and the control plane telemetry.

At each plane, the telemetry can be further partitioned into five distinct components:

Data Source: Determine where the original data is acquired. The data source usually just provides raw data which needs further processing. A data source can be considered a probe. A probe can be statically installed or dynamically installed.

Data Subscription: Determine the protocol and channel for applications to acquire desired data. Data subscription is also responsible to define the desired data that might not be directly available form data sources. The subscription data can be described by a model. The model can be statically installed or dynamically installed.

Data Generation: The original data needs to be processed, encoded, and formatted in network devices to meet application subscription requirements. This may involve in-network computing and processing on either the fast path or the slow path in network devices.

Data Export: Determine how the ready data are delivered to applications.

Data Analysis and Storage: In this final step, data is consumed by applications or stored for future reference. Data analysis can be interactive. It may initiate further data subscription.

```
+-------------------------------+
|                               |
|    Data Analysis/Storage      |
|                               |
+-------------------------------+
        |               ^
        |               |
        V               |
+---------------+---------------+
|               |               |
| Data          | Data          |
| Subscription  | Export        |
|               |               |
+---------------+---------------|
|                               |
|       Data Generation         |
|                               |
+-------------------------------|
|                               |
|       Data Source             |
|                               |
+-------------------------------+
```

Figure 2: Components in the Network Telemetry Framework

Since most existing standard-related work belongs to the first four components, in the remainder of the document, we focus on these components only.

3.1. Existing Works Mapped in the Framework

The following table provides a non-exhaustive list of existing works (mainly published in IETF and with the emphasis on the latest new technologies) and shows their positions in the framework.

	Management Plane	Control Plane	Data Plane
Data Source	YANG Data Store	Control Proto. Network State	Flow/Packet Statistics States DPI
Data Subscribe	gPRC YANG PUSH	NETCONF/YANG BGP	NETCONF/YANG YANG FSM
Data Generation	Soft DNP	Soft DNP	In-situ OAM IPFPM Hard DNP
Data Export	gRPC YANG PUSH UDP	BMP	IPFIX UDP

Figure 3: Existing Work

3.2. Management Plane Telemetry

3.2.1. Requirements and Challenges

The management plane of the network element interacts with the Network Management System (NMS), and provides information such as performance data, network logging data, network warning and defects data, and network statistics and state data. Some legacy protocols are widely used for the management plane, such as SNMP and Syslog, but these protocols do not meet the requirements of the automatic network operation applications.

New management plane telemetry protocols should consider the following requirements:

Convenient Data Subscription: An application should have the freedom to choose the data export means such as the data types and the export frequency.

Structured Data: For automatic network operation, machines will replace human for network data comprehension. The schema languages such as YANG can efficiently describe structured data and normalize data encoding and transformation.

High Speed Data Transport: In order to retain the information, a server needs to send a large amount of data at high frequency. Compact encoding formats are needed to compress the data and improve the data transport efficiency. The push mode, by replacing the poll mode, can also reduce the interactions between clients and servers, which help to improve the server's efficiency.

3.2.2. Push Extensions for NETCONF

NETCONF [RFC6241] is one popular network management protocol, which is also recommended by IETF. Although it can be used for data collection, NETCONF is good at configurations. YANG Push [I-D.ietf-netconf-yang-push] extends NETCONF and enables subscriber applications to request a continuous, customized stream of updates from a YANG datastore. Providing such visibility into changes made upon YANG configuration and operational objects enables new capabilities based on the remote mirroring of configuration and operational state. Moreover, distributed data collection mechanism [I-D.zhou-netconf-multi-stream-originators] via UDP based publication channel [I-D.ietf-netconf-udp-pub-channel] provides enhanced efficiency for the NETCONF based telemetry.

3.2.3. gRPC Network Management Interface

gRPC Network Management Interface (gNMI) [I-D.openconfig-rtgwg-gnmi-spec] is a network management protocol based on the gRPC [I-D.kumar-rtgwg-grpc-protocol] RPC (Remote Procedure Call) framework. With a single gRPC service definition, both configuration and telemetry can be covered. gRPC is an HTTP/2 [RFC7540] based open source micro service communication framework. It provides a number of capabilities that makes it well-suited for network telemetry, including:

- o Full-duplex streaming transport model combined with a binary encoding mechanism provided further improved telemetry efficiency.

- o gRPC provides higher-level features consistency across platforms that common HTTP/2 libraries typically do not. This characteristic is especially valuable for the fact that telemetry data collectors normally reside on a large variety of platforms.

- o The built-in load-balancing and failover mechanism.

3.3. Control Plane Telemetry

3.3.1. Requirements and Challenges

The control plane telemetry refers to the health condition monitoring of different network protocols, which covers Layer 2 to Layer 7. Keeping track of the running status of these protocols is beneficial for detecting, localizing, and even predicting various network issues, as well as network optimization, in real-time and in fine granularities.

One of the most challenging problems for the control plane telemetry is how to correlate the E2E Key Performance Indicators (KPI) to a specific layer's KPIs. For example, an IPTV user may describe his User Experience (UE) by the video fluency and definition. Then in case of an unusually poor UE KPI or a service disconnection, it is non-trivial work to delimit and localize the issue to the responsible protocol layer (e.g., the Transport Layer or the Network Layer), the responsible protocol (e.g., ISIS or BGP at the Network Layer), and finally the responsible device(s) with specific reasons.

Traditional OAM-based approaches for control plane KPI measurement include PING (L3), Tracert (L3), Y.1731 (L2) and so on. One common issue behind these methods is that they only measure the KPIs instead of reflecting the actual running status of these protocols, making them less effective or efficient for control plane troubleshooting and network optimization. An example of the control plane telemetry is the BGP monitoring protocol (BMP), it is currently used to monitoring the BGP routes and enables rich applications, such as BGP peer analysis, AS analysis, prefix analysis, security analysis, and so on. However, the monitoring of other layers, protocols and the cross-layer, cross-protocol KPI correlations are still in their infancies (e.g., the IGP monitoring is missing), which require substantial further research.

3.3.2. BGP Monitoring Protocol

BGP Monitoring Protocol (BMP) [RFC7854] is used to monitor BGP sessions and intended to provide a convenient interface for obtaining route views.

The BGP routing information is collected from the monitored device(s) to the BMP monitoring station by setting up the BMP TCP session. The BGP peers are monitored by the BMP Peer Up and Peer Down Notifications. The BGP routes (including Adjacency_RIB_In [RFC7854], Adjacency_RIB_out [I-D.ietf-grow-bmp-adj-rib-out], and Local_Rib [I-D.ietf-grow-bmp-local-rib] are encapsulated in the BMP Route Monitoring Message and the BMP Route Mirroring Message, in the form of both initial table dump and real-time route update. In addition, BGP statistics are reported through the BMP Stats Report Message, which could be either timer triggered or event driven. More BMP extensions can be explored to enrich the applications of BGP monitoring.

3.4. Data Plane Telemetry

3.4.1. Requirements and Challenges

An effective data plane telemetry system relies on the data that the network device can expose. The data's quality, quantity, and timeliness must meet some stringent requirements. This raises some challenges to the network data plane devices where the first hand data originate.

- A data plane device's main function is user traffic processing and forwarding. While supporting network visibility is important, the telemetry is just an auxiliary function and it should not impede normal traffic processing and forwarding (i.e., the performance is not lowered and the behavior is not altered due to the telemetry functions).

- The network operation applications requires end-to-end visibility from various sources, which results in a huge volume of data. However, the sheer data quantity should not stress the network bandwidth, regardless of the data delivery approach (i.e., through in-band or out-of-band channels).

- The data plane devices must provide the data in a timely manner with the minimum possible delay. Long processing, transport, storage, and analysis delay can impact the effectiveness of the control loop and even render the data useless.

- The data should be structured and labeled, and easy for applications to parse and consume. At the same time, the data types needed by applications can vary significantly. The data plane devices need to provide enough flexibility and programmability to support the precise data provision for applications.

- The data plane telemetry should support incremental deployment and work even though some devices are unaware of the system. This challenge is highly relevant to the standards and legacy networks.

The industry has agreed that the data plane programmability is essential to support network telemetry. Newer data plane chips are all equipped with advanced telemetry features and provide flexibility to support customized telemetry functions.

3.4.2. Technique Classification

There can be multiple possible dimensions to classify the data plane telemetry techniques.

Active and Passive: The active and passive methods (as well as the hybrid types) are well documented in [RFC7799]. The passive methods include TCPDUMP, IPFIX [RFC7011], sflow, and traffic mirror. These methods usually have low data coverage. The bandwidth cost is very high in oreder to improve the data coverage. On the other hand, the active methods include Ping, Traceroute, OWAMP [RFC4656], and TWAMP [RFC5357]. These methods are intrusive and only provide indirect network measurement results. The hybrid methods, including in-situ OAM [I-D.brockners-inband-oam-requirements], IPFPM [RFC8321], and Multipoint Alternate Marking [I-D.fioccola-ippm-multipoint-alt-mark], provide a well-balanced and more flexible approach. However, these methods are also more complex to implement.

In-Band and Out-of-Band: The telemetry data, before being exported to some collector, can be carried in user packets. Such methods are considered in-band (e.g., in-situ OAM [I-D.brockners-inband-oam-requirements]). If the telemetry data is directly exported to some collector without modifying the user packets, Such mothods are considered out-of-band (e.g., postcard-based INT). It is possible to have hybrid methods. For example, only the telemetry instruction or partitial data is carried by user packets (e.g., IPFPM [RFC8321]).

E2E and In-Network: Some E2E methods start from and end at the network end hosts (e.g., Ping). The other methods work in networks and are transparent to end hosts. However, if needed, the in-network methods can be easily extended into end hosts.

Flow, Path, and Node: Depending on the telemetry objective, the methods can be flow-based (e.g., in-situ OAM [I-D.brockners-inband-oam-requirements]), path-based (e.g., Traceroute), and node-based (e.g., IPFIX [RFC7011]).

3.4.3. The IPFPM technology

The Alternate Marking method is efficient to perform packet loss, delay, and jitter measurements both in an IP and Overlay Networks, as presented in IPFPM [RFC8321] and [I-D.fioccola-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point and multipoint-to-multipoint flows. Alternate Marking creates batches of packets by alternating the value of 1 bit (or a label) of the packet header. These batches of packets are unambiguously recognized over the network and the comparison of packet counters for each batch allows the packet loss calculation. The same idea can be applied to delay measurement by selecting ad hoc packets with a marking bit dedicated for delay measurements.

Alternate Marking method needs two counters each marking period for each flow under monitor. For instance, by considering n measurement points and m monitored flows, the order of magnitude of the packet counters for each time interval is n*m*2 (1 per color).

Since networks offer rich sets of network performance measurement data (e.g packet counters), traditional approaches run into limitations. One reason is the fact that the bottleneck is the generation and export of the data and the amount of data that can be reasonably collected from the network. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.

Multipoint Alternate Marking approach, described in [I-D.fioccola-ippm-multipoint-alt-mark], aims to resolve this issue and makes the performance monitoring more flexible in case a detailed analysis is not needed.

An application orchestrates network performance measurements tasks across the network to allow an optimized monitoring and it can calibrate how deep can be obtained monitoring data from the network by configuring measurement points roughly or meticulously.

Using Alternate Marking, it is possible to monitor a Multipoint Network without examining in depth by using the Network Clustering (subnetworks that are portions of the entire network that preserve the same property of the entire network, called clusters). So in case there is packet loss or the delay is too high the filtering criteria could be specified more in order to perform a detailed analysis by using a different combination of clusters up to a per-flow measurement as described in IPFPM [RFC8321].

In summary, an application can configure initially an end to end monitoring between ingress points and egress points of the network. If the network does not experiment issues, this approximate monitoring is good enough and is very cheap in terms of network resources. But, in case of problems, the application becomes aware of the issues from this approximate monitoring and, in order to localize the portion of the network that has issues, configures the measurement points more exhaustively. So a new detailed monitoring is performed. After the detection and resolution of the problem the initial approximate monitoring can be used again.

3.4.4. Dynamic Network Probe

Hardware based Dynamic Network Probe (DNP) [I-D.song-opsawg-dnp4iq] provides a programmable means to customize the data that an application collects from the data plane. A direct benefit of DNP is the reduction of the exported data. A full DNP solution covers several components including data source, data subscription, and data generation. The data subscription needs to define the custom data which can be composed and derived from the raw data sources. The data generation takes advantage of the moderate in-network computing to produce the desired data.

While DNP can introduce unforeseeable flexibility to the data plane telemetry, it also faces some challenges. It requires a flexible data plane that can be dynamically reprogrammed at run-time. The programming API is yet to be defined.

3.4.5. IP Flow Information Export (IPFIX) protocol

Traffic on a network can be seen as a set of flows passing through network elements. IP Flow Information Export (IPFIX) [RFC7011] provides a means of transmitting traffic flow information for administrative or other purposes. A typical IPFIX enabled system includes a pool of Metering Processes collects data packets at one or more Observation Points, optionally filters them and aggregates information about these packets. An Exporter then gathers each of the Observation Points together into an Observation Domain and sends this information via the IPFIX protocol to a Collector.

3.4.6. In-Situ OAM

Traditional passive and active monitoring and measurement techniques are either inaccurate or resource-consuming. It is preferable to directly acquire data associated with a flow’s packets when the packets pass through a network. In-situ OAM (iOAM) [I-D.brockners-inband-oam-requirements], a data generation technique, embeds a new instruction header to user packets and the instruction directs the network nodes to add the requested data to the packets. Thus, at the path end the packet’s experience on the entire forwarding path can be collected. Such firsthand data is invaluable to many network OAM applications.

However, iOAM also faces some challenges. The issues on performance impact, security, scalability and overhead limits, encapsulation difficulties in some protocols, and cross-domain deployment need to be addressed.

3.5. External Data and Event Telemetry

Events that occur outside the boundaries of the network system are another important source of telemetry information. Correlating both internal telemetry data and external events with the requirements of network systems, as presented in Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems [I-D.pedro-nmrg-anticipated-adaptation], provides a strategic and functional advantage to management operations.

3.5.1. Requirements and Challenges

As with other sources of telemetry information, the data and events must meet strict requirements, especially in terms of timeliness, which is essential to properly incorporate external event information to management cycles. Thus, the specific challenges are described as follows:

- The role of external event detector can be played by multiple elements, including hardware (e.g. physical sensors, such as seismometers) and software (e.g. Big Data sources that analyze streams of information, such as Twitter messages). Thus, the transmitted data must support different shapes but, at the same time, follow a common but extensible ontology.

- Since the main function of the external event detectors is actually to perform the notifications, their timeliness is assumed. However, once messages have been dispatched, they must be quickly collected and inserted into the control plane with variable priority, which will be high for important sources and/or important events and low for secondary ones.

- The ontology used by external detectors must be easily adopted by current and future devices and applications. Therefore, it must be easily mapped to current information models, such as in terms of YANG.

Organizing together both internal and external telemetry information will be key for the general exploitation of the management possibilities of current and future network systems, as reflected in the incorporation of cognitive capabilities to new hardware and software (virtual) elements.

4. Security Considerations

TBD

5. IANA Considerations

This document includes no request to IANA.

6. Contributors

The other main contributors of this document are listed as follows.

- James N. Guichard, Huawei
- Yunan Gu, Huawei

7. Acknowledgments

We would like to thank Victor Liu and others who have provided helpful comments and suggestions to improve this document.

8. References

8.1. Normative References

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

8.2. Informative References

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

[I-D.fioccola-ippm-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-fioccola-ippm-multipoint-alt-mark-04 (work in progress), June 2018.

[I-D.ietf-grow-bmp-adj-rib-out]
Evens, T., Bayraktar, S., Lucente, P., Mi, K., and S. Zhuang, "Support for Adj-RIB-Out in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-adj-rib-out-01 (work in progress), March 2018.

[I-D.ietf-grow-bmp-local-rib]
Evens, T., Bayraktar, S., Bhardwaj, M., and P. Lucente, "Support for Local RIB in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-local-rib-01 (work in progress), February 2018.

[I-D.ietf-netconf-udp-pub-channel]
Zheng, G., Zhou, T., and A. Clemm, "UDP based Publication Channel for Streaming Telemetry", draft-ietf-netconf-udp-pub-channel-03 (work in progress), July 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-17 (work in progress), July 2018.

[I-D.kumar-rtgwg-grpc-protocol]
Kumar, A., Kolhe, J., Ghemawat, S., and L. Ryan, "gRPC Protocol", draft-kumar-rtgwg-grpc-protocol-00 (work in progress), July 2016.

[I-D.openconfig-rtgwg-gnmi-spec]
Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack, C., and C. Morrow, "gRPC Network Management Interface (gNMI)", draft-openconfig-rtgwg-gnmi-spec-01 (work in progress), March 2018.

[I-D.pedro-nmrg-anticipated-adaptation]
Martinez-Julia, P., "Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems", draft-pedro-nmrg-anticipated-adaptation-02 (work in progress), June 2018.

[I-D.song-opsawg-dnp4iq]
Song, H. and J. Gong, "Requirements for Interactive Query with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01 (work in progress), June 2017.

[I-D.zhou-netconf-multi-stream-originators]
Zhou, T., Zheng, G., Voit, E., Clemm, A., and A. Bierman, "Subscription to Multiple Stream Originators", draft-zhou-netconf-multi-stream-originators-02 (work in progress), May 2018.

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <https://www.rfc-editor.org/info/rfc1157>.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <https://www.rfc-editor.org/info/rfc4656>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/info/rfc5357>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[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, <https://www.rfc-editor.org/info/rfc7276>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

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

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

Authors' Addresses

Haoyu Song (editor)
Huawei
2330 Central Expressway
Santa Clara
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: zhoutianran@huawei.com

Zhenbin Li
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: lizhenbin@huawei.com

Giuseppe Fioccola
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

Zhenqiang Li
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing, 100032
P.R. China

Email: lizhenqiang@chinamobile.com

Pedro Martinez-Julia
NICT
4-2-1, Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Phone: +81 42 327 7293
Email: pedro@nict.go.jp

Laurent Ciavaglia
Nokia
Villarceaux 91460
France

Email: laurent.ciavaglia@nokia.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, 102209
P.R. China

Email: wangaj.bri@chinatelecom.cn

OPSAWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2018

Q. Wu
M. Wang
Huawei
M. Boucadair
Orange

March 2, 2018

A YANG Data Module for Network Virtualization Overlay Resource Management

draft-wu-opsawg-network-overlay-resource-model-00

Abstract

This document defines a YANG data module for Network Virtualization Overlay Resource Management. It is a resource facing model independent of control plane protocols and captures topological and resource related information pertaining to Network Virtualization Overlay.

This module enables clients, which interact with a network orchestrator or controller via a REST interface, for Network Virtualization Overlay topology related operations such as obtaining and allocating the relevant topology resource information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Conventions used in this document 3
3. Overview of Network Virtualization Overlay Resource Management Model 4
  3.1. VN Service Configuration 6
    3.1.1. VN and Network Access Association Configuration . . . 6
    3.1.2. Traffic Performance Requirements Configuration . . . 7
  3.2. VN Service Topology Resource Distribution configuration . 10
4. RPC Definitions for Computation of TE Path Element List and Network Access Connectivity List 11
5. Data Hierarchy 13
6. Network Virtualization Overlay Management YANG Module 17
7. Security Considerations 33
8. IANA Considerations 34
9. References 34
  9.1. Normative References 34
  9.2. Informative References 35
Authors' Addresses 35

1. Introduction

[RFC8299] defines customer service model for L3VPN service that can be used to describe a service as offered or delivered to a customer by a network operator. As described in [RFC8309], a customer service model is not resource facing model and does not describes how a network operator realizes and delivers the service described by the module since it is not used to directly configure network devices, protocols, or functions or something sent to network devices (i.e., routers or switches) for processing.

This document defines a YANG module for Network Virtualization Overlay Management. It is a resource facing model independent of control plane protocols and captures topological and resource related information pertaining to Network Virtualization Overlay.

This module enables clients to interact with a network orchestrator or controller via a RESTful interface, for providing connectivity services over a Network Virtualization Overlay topology. In particular, this module supports operations such as exposing abstract service topology, retrieving, and allocating the relevant topology resource information.

As a reminder, and as defined in [RFC7297], the IP connectivity service is the IP transfer capability characterized by a (Source Nets, Destination Nets, Guarantees, Scope) tuple where "Source Nets" is a group of unicast IP addresses, "Destination Nets" is a group of IP unicast and/or multicast addresses, and "Guarantees" reflects the guarantees (expressed in terms of Quality Of Service (QoS), performance, and availability, for example) to properly forward traffic to the said "Destination". Finally, the "Scope" denotes the (network) perimeter (e.g., between Provider Edge (PE) routers or Customer Nodes) where the said guarantees need to be provided. These requirements include: reachability scope (e.g., limited scope, Internet-wide), direction (in/ou), bandwidth requirements, QoS parameters (e.g., one-way delay [RFC7679], loss [RFC7680], or one-way delay variation (jitter) [RFC3393]), protection, and high-availability guidelines (e.g., restoration in less than 50 ms, 100 ms, or 1 second).

The module includes flow identification and classification rules that are required for traffic conformance purposes.

How the data captured using this YANG module is tranlated into network-spefic clauses is out of scope.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

The following notations are used within the data tree and carry the meaning as below.

Each node is printed as:

<status> <flags> <name> <opts> <type>

<status> is one of:
+ for current

<flags> is one of:

rw for configuration data
ro for non-configuration data
-x for rpcs
-n for notifications
-w for writable

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>.

<opts> is one of:

? for an optional leaf or choice
! for a presence container
* for a leaf-list or list
[<keys>] for a list's keys
(choice)/:(case) Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":")
<type> is the name of the type for leafs and leaf-lists

3. Overview of Network Virtualization Overlay Resource Management Model

```
      ----------- l3vpn-svc
                    Model   |
    Customer      l2vpn-svc |
    Facing Model    Model   |
                     +----------------------+
      ---------------|   Service component  |
                     +-----------+----------+
                                 |
                      VN Overlay |
                      Resource   |
    Resource          Model      |
    Facing Model                 |
                                 |
                                 |
                      +----------+-----------+
   ------------  +----|   Config component   |-------+
                /     +----------------------+        \   Network
               /             /            \            \  Configuration
              /             /              \             \ models
             /             /                \             \
     +------+   Bearer    +------+           +------+      +------+
     | CE A + ----------- + PE A |           | PE B + ---- + CE B |
     +------+  Connection +------+           +------+      +------+

                Site A                                Site B
```

L3VPN and L2VPN service models provide an abstracted view of the Layer 3 and Layer 2 VPN service configuration components. Services are built from a combination of network elements and protocols configuration, but are specified for service users in more abstract terms, e.g., these models will specify where to create site and establish site-network-access of a particular site to the provider network (e.g., PE, aggregation switch) and what service requirements of each site-network-access are.

Site location can be determined based on proposed location parameters and constraints in these service models and service requirements of each site-network-access can be determined based on traffic performance metrics (e.g., one-way delay, one-way delay variation, bandwidth) of each PE-CE link connectivity and traffic performance metrics of each service flow or application. The management system will use service models as an input to select appropriate PEs and CEs, allocate interface on the node, generate PE and CE configuration associated with each PE-CE link.

Based on selected PE and CE configuration on each site-network-access of a particular site, the management system can use L3VPN service model and L2VPN service model as inputs and translate it into resource facing model, i.e., the network virtualization overlay resource model.

This resource facing model can be seen as the projection model of L3VPN service and L2VPN service model and is used to compute path elements and the network access connectivity list when two sites belonging to one VPN spanning across several domains. It also can be combined with other performance measurement or warning models to expose abstract service topology and resource distribution in the network re-optimization cases.

3.1. VN Service Configuration

The YANG module is divided into two main containers: "vn-services" and "sites".

The "vn-service" list under the vn-services container defines global parameters for the VN service for a specific customer. The "vn-id" provided in the vn-service list refers to an internal reference for this VN service, while the customer name refers to a more-explicit reference to the customer. The "vn-type" in the vn-service list refers to a set of basic VPN type. In addition, each "vn-service" also include a list of "site-network-access".

The service requirements on each "site-network-access" or site to site service requirements is specified in details in the service container under "sites/site" or "sites/site/site-network-access".

3.1.1. VN and Network Access Association Configuration

Within a given VN service there can be one or more VN and Network Access Associations(VNAAs). VNAAs are represented as a list and indexed by the vn-id and vn-type.

```
  module: ietf-vn-rsc
 +--rw vn-rsc
  +--rw vn-services
   | +--rw vn-service* [vn-id]
   |   +--rw vn-id          svc-id
   |   +--rw vn-type        identityref
   .
   .
   |   +--rw site-network-accesses
   |   +--rw site-network-access* [site-network-access-id]
   |     +--rw site-network-access-id   svc-id
```

Snippet of data hierarchy related to VN and Network Access Associations (VNAA)

3.1.2. Traffic Performance Requirements Configuration

3.1.2.1. Per-Site Network Access Requirements

Per-Site network access traffic performance requirements are represented as a list within the data hierarchy and indexed by the key site-network-access-id.

Traffic Performance requirements include latency, jitter, and bandwidth utilization. Upload bandwidth and download bandwidth are performance parameters associated each domain-network-access.

Latency, jitter, and bandwidth utilization are performance requirements associated with each service flow or application.

```
module: ietf-vn-rsc
   +--rw site-network-accesses
     +--rw site-network-access* [site-network-access-id]
      +--rw site-network-access-id   leafref
      +--rw device-id   leafref
      +--rw access-diversity {site-diversity}?
      | +--rw groups
      | | +--rw group* [group-id]
      | |   +--rw group-id  string
      | +--rw constraints
      |   +--rw constraint* [constraint-type]
      |    +--rw constraint-type  identityref
      |    +--rw target
      |      +--rw (target-flavor)?
      |       +--:(id)
      |       | +--rw group* [group-id]
      |       |    ...
      |       +--:(all-accesses)
      |       | +--rw all-other-accesses?  empty
      |       +--:(all-groups)
      |         +--rw all-other-groups?   empty
      +--rw service
      | +--rw svc-input-bandwidth?  uint32
      | +--rw svc-output-bandwidth?  uint32
      | +--rw svc-mtu?        uint16
      | +--rw qos {qos}?
      | | +--rw qos-classification-policy
      | | | +--rw rule* [id]
      | | |   +--rw id          uint16
      | | |   +--rw (match-type)?
      | | |   | +--:(match-flow)
      | | |   | | +--rw match-flow
      | | |   | |    ...
      | | |   | +--:(match-application)
      | | |   |   +--rw match-application?  identityref
      | | |   +--rw target-class-id?   string
      | | +--rw qos-profile
      | |   +--rw (qos-profile)?
      | |    +--:(standard)
      | |    | +--rw profile?  string
      | |    +--:(custom)
      | |      +--rw classes {qos-custom}?
      | |       +--rw class* [class-id]
```

Snippet of data hierarchy related to Per Site network access QoS requirements

3.1.2.2. Site-to-Site Traffic Performance Requirements

QoS guarantees denote a set of transfer performance metrics that characterize the quality of the transfer treatment to be experienced (when crossing a transport infrastructure) by a flow issued from or forwarded to a (set of) sites.

Suppose one VPN has multiple sites and any two sites span across multiple domains, site-to-site network access QoS requirements can be used to describe QoS requirements across sites.

Site-to-site network access traffic performance requirements are represented as a list within the data hierarchy and indexed by the key 'site-id'. The source site is specified as 'site-id' under site list, the 'target-site' is specified under match-flow case.

Traffic performance requirements include latency, jitter, and bandwidth utilization.

Shaping/policing filters may be applied so as to assess whether traffic is within the capacity profile or out of profile. Out-of-profile traffic may be discarded or assigned another class.

```
         module: ietf-vn-rsc
         +--rw sites
           +--rw site* [site-id]
            +--rw site-id         svc-id
            +--rw service
            | +--rw qos {qos}?
            | | +--rw qos-classification-policy
            | | | +--rw rule* [id]
            | | |   +--rw id          uint16
            | | |   +--rw (match-type)?
            | | |   | +--:(match-flow)
            | | |   | | +--rw match-flow
            | | |   | |   +--rw target-sites*    svc-id
            | | |   +--rw target-class-id?   string
            | | +--rw qos-profile
            | |   +--rw (qos-profile)?
            | |    +--:(standard)
            | |    | +--rw profile?  string
            | |    +--:(custom)
            | |      +--rw classes {qos-custom}?
            | |       +--rw class* [class-id]
            | |         +--rw class-id   string
            | |         +--rw rate-limit?  uint8
            | |         +--rw latency
            | |         | +--rw (flavor)?
            | |         |     ...
            | |         +--rw jitter
            | |         | +--rw (flavor)?
            | |         |     ...
            | |         +--rw bandwidth
            | |          +--rw guaranteed-bw-percent?  uint8
            | |          +--rw end-to-end?       empty
```

Snippet of data hierarchy related to Site to Site QoS requirements

3.2. VN Service Topology Resource Distribution configuration

A ’site’ is composed of at least one "site-network-access" and, in the case of multihoming, may have multiple site-network-access points.

For each "site-network-access", the ingress device/customer device and/or egress device has been selected to connect to the provider network, ingress device list is specified under site and egress device is specified under vn-attachment container.

With selected ingress device and egress device and VN membership, VN service topology can be constructed. Resource allocation for Site to

Site connectivity or connectivity within site can be further calculated based on this VN service topology.

```
               VPN1-Site1                               VPN1-Site2
         +------------------------------------------------------------+
        /   [CE1]..                              [PE2______[CE3]     /
       /    /  \  :                              : \_       / :     /
      /    /    \  :                             :   \_    /  :    /
     /    /      \  :                           :      \  /   :   /
    /   [CE2]___[PE1]:                         :       [CE4]  :  /
   +------:-------:---:---------------------------------:-----:-+
            :        :   :         :           :     :
            :         :   :       :            :     :
            :  +-------:---:-----:-------------:-----:----+
            : /      [X1]__:___:___________[X2]   :     /
            :/         / \_  : :       _____/ /   :     /
            :         /    \_ :  _____/      /   :     /
           /:        /       \: /           /   :     /
          / :       /        [X5]          /   :     /
         /   :     /       __/ \__        /   :     /
        /     :   /    ___/       \__    /   :     /
       /       : / ___/              \  /   :     /
      /        [X4]__________________[X3]..:     /
     +------------------------------------------+
                                    L3 Topology
```

4. RPC Definitions for Computation of TE Path Element List and Network Access Connectivity List

The RPC model facilitates issuing commands to a NETCONF server (in this case to the device that need to execute the path computation API command or path computation algorithm) and obtain a response. RPC model defined here abstracts path computation specific commands in a technology independent manner.

There are two RPC commands defined for the purpose of computation of path element list and network access connectivity list respectively. In this section we present a snippet of the path element list computation command and network access connectivity list computation for illustration purposes. Please refer to Section 3.4 for the complete data hierarchy and Section 4 for the YANG model.

```
  rpcs:
    +---x vn-path-element-compute
    |  +---w input
    |  |  +---w vn-member-list* [vn-member-id]
    |  |     +---w vn-member-id          -> /vn-svc/vn-services/vn-service/vn-id
    |  |     +---w constraint
    |  |     |  +---w path-element* [path-element-id]
    |  |     |     +---w path-element-id
    |  |     |     +---w address?
    |  |     +---w objective-function?   identityref
    |  |     +---w metric* [metric-type]
    |  |        +---w metric-type     identityref
    |  |        +---w metric-value?   uint32
    |  +--ro output
    |     +--ro vn-member-list* [vn-member-id]
    |        +--ro vn-member-id    -> /vn-svc/vn-services/vn-service/vn-id
    |        +--ro metric* [metric-type]
    |        |  +--ro metric-type     identityref
    |        |  +--ro metric-value?   uint32
    |        +--ro path
    |           +--ro path-element* [path-element-id]
    |              +--ro path-element-id
    +---x vn-network-connectivity-stitch
       +---w input
       |  +---w vn-member-list* [vn-id]
       |     +---w vn-id            -> /vn-svc/vn-services/vn-service/vn-id
       |     +---w source-access* [access-id]
       |     |  +---w access-id
       |     |  +---w destination-access* [access-id]
       |     +---w objective-function?   identityref
       |     +---w metric* [metric-type]
       |        +---w metric-type     identityref
       |        +---w metric-value?   uint32
       +--ro output
          +--ro vn-access-list* [index]
             +--ro index        uint32
             +--ro source-access -> /vn-svc/sites/site/site-network-accesses/sit
e-network-access/site-network-access-id
             +--ro destination-access-> /vn-svc/sites/site/site-network-accesses
/site-network-access/site-network-access-id
             +--ro multi-domain-network-access-list * [domain-id]
                +--ro domain-id                 svc-id
                +--ro network-access-id         svc-id
```

With these two RPC commands, we can calculate

Path element list that is applied to network access connectivity within the site, or Site to Site connectivity or end to end connectivity.

Network access connectivity list that is applied to site to site connectivity and end to end connectivity spanning across multiple domains.

5. Data Hierarchy

The figure below describes the overall structure of the YANG module:

```
module: ietf-vn-rsc
    +--rw vn-rsc
       +--rw vn-services
       |  +--rw vn-service* [vn-id]
       |     +--rw vn-id                   svc-id
       |     +--rw customer-name?          string
       |     +--rw service-topology?       identityref
       |     +--rw site-network-accesses
       |        +--rw site-network-access* [site-network-access-id]
       |           +--rw site-network-access-id    svc-id
       +--rw sites
          +--rw site* [site-id]
             +--rw site-id                 svc-id
             +--rw cpe-devices
             |  +--rw cpe-device* [device-id]
             |     +--rw device-id        svc-id
             |     +--rw address-family?   address-family
             |     +--rw address?          inet:ip-address
             |     +--rw interfaces
             |        +--rw interface?        if:interface-ref
             |        +--rw sub-interfaces*   if:interface-ref
             +--rw service
             |  +--rw qos {qos}?
             |     +--rw qos-classification-policy
             |     |  +--rw rule* [id]
             |     |     +--rw id                  string
             |     |     +--rw (match-type)?
             |     |     |  +--:(match-flow)
             |     |     |  |  +--rw match-flow
             |     |     |  |     +--rw dscp?               inet:dscp
             |     |     |  |     +--rw dot1p?              uint8
             |     |     |  |     +--rw ipv4-src-prefix?    inet:ipv4-prefix
             |     |     |  |     +--rw ipv6-src-prefix?    inet:ipv6-prefix
             |     |     |  |     +--rw ipv4-dst-prefix?    inet:ipv4-prefix
             |     |     |  |     +--rw ipv6-dst-prefix?    inet:ipv6-prefix
             |     |     |  |     +--rw l4-src-port?        inet:port-number
             |     |     |  |     +--rw target-sites*       svc-id {target-site
s}?
             |     |     |  |     +--rw l4-src-port-range
             |     |     |  |     |  +--rw lower-port?   inet:port-number
             |     |     |  |     |  +--rw upper-port?   inet:port-number
```

```
             |     |     |  |     +--rw l4-dst-port?         inet:port-number
             |     |     |  |     +--rw l4-dst-port-range
             |     |     |  |     |  +--rw lower-port?   inet:port-number
             |     |     |  |     |  +--rw upper-port?   inet:port-number
             |     |     |  |     +--rw protocol-field?      union
             |     |     |  +--:(match-application)
             |     |     |     +--rw match-application?   identityref
             |     |     +--rw target-class-id?     string
             |     +--rw qos-profile
             |        +--rw (qos-profile)?
             |           +--:(standard)
             |           |  +--rw profile?
             |           |     -> /vn-svc/vpn-profiles/valid-provider-identifier
s/qos-profile-identifier/id
             |           +--:(custom)
             |              +--rw classes {qos-custom}?
             |                 +--rw class* [class-id]
             |                    +--rw class-id      string
             |                    +--rw direction?    identityref
             |                    +--rw rate-limit?   uint8
             |                    +--rw latency
             |                    |  +--rw (flavor)?
             |                    |     +--:(lowest)
             |                    |     |  +--rw use-lowest-latency?   empty
             |                    |     +--:(boundary)
             |                    |        +--rw latency-boundary?     uint16
             |                    +--rw jitter
             |                    |  +--rw (flavor)?
             |                    |     +--:(lowest)
             |                    |     |  +--rw use-lowest-jitter?   empty
             |                    |     +--:(boundary)
             |                    |        +--rw latency-boundary?    uint32
             |                    +--rw bandwidth
             |                       +--rw guaranteed-bw-percent    uint8
             |                       +--rw end-to-end?              empty
             +--rw site-network-accesses
                +--rw site-network-access* [site-network-access-id]
                   +--rw site-network-access-id
                           -> /vn-svc/vn-services/vn-service/site-network-access
es/site-network-access/site-network-access-id
                   +--rw ingress-device-id?              -> /vn-svc/sites/site
/cpe-devices/cpe-device/device-id
                   +--rw access-diversity {site-diversity}?
                   |  +--rw groups
                   |  |  +--rw group* [group-id]
                   |  |     +--rw group-id    string
                   |  +--rw constraints
                   |     +--rw constraint* [constraint-type]
                   |        +--rw constraint-type    identityref
                   |        +--rw target
                   |           +--rw (target-flavor)?
```

```
                   |              +--:(id)
                   |              |  +--rw group* [group-id]
                   |              |     +--rw group-id    string
                   |              +--:(all-accesses)
                   |              |  +--rw all-other-accesses?   empty
                   |              +--:(all-groups)
                   |                 +--rw all-other-groups?     empty
                   +--rw service
                   |  +--rw svc-input-bandwidth?    uint32
                   |  +--rw svc-output-bandwidth?   uint32
                   |  +--rw svc-mtu?                uint16
                   |  +--rw qos {qos}?
                   |     +--rw qos-classification-policy
                   |     |  +--rw rule* [id]
                   |     |     +--rw id                   string
                   |     |     +--rw (match-type)?
                   |     |     |  +--:(match-flow)
                   |     |     |  |  +--rw match-flow
                   |     |     |  |     +--rw dscp?                inet:dscp
                   |     |     |  |     +--rw dot1p?               uint8
                   |     |     |  |     +--rw ipv4-src-prefix?     inet:ipv4-pre
fix
                   |     |     |  |     +--rw ipv6-src-prefix?     inet:ipv6-pre
fix
                   |     |     |  |     +--rw ipv4-dst-prefix?     inet:ipv4-pre
fix
                   |     |     |  |     +--rw ipv6-dst-prefix?     inet:ipv6-pre
fix
                   |     |     |  |     +--rw l4-src-port?         inet:port-num
ber
                   |     |     |  |     +--rw target-sites*        svc-id {targe
t-sites}?
                   |     |     |  |     +--rw l4-src-port-range
                   |     |     |  |     |  +--rw lower-port?   inet:port-number
                   |     |     |  |     |  +--rw upper-port?   inet:port-number
                   |     |     |  |     +--rw l4-dst-port?         inet:port-num
ber
                   |     |     |  |     +--rw l4-dst-port-range
                   |     |     |  |     |  +--rw lower-port?   inet:port-number
                   |     |     |  |     |  +--rw upper-port?   inet:port-number
                   |     |     |  |     +--rw protocol-field?      union
                   |     |     |  +--:(match-application)
                   |     |     |     +--rw match-application?   identityref
                   |     |     +--rw target-class-id?     string
                   |     +--rw qos-profile
                   |        +--rw (qos-profile)?
                   |           +--:(standard)
                   |           |  +--rw profile?
                   |           |          -> /vn-svc/vpn-profiles/valid-provider
-identifiers/qos-profile-identifier/id
                   |           +--:(custom)
                   |              +--rw classes {qos-custom}?
                   |                 +--rw class* [class-id]
                   |                    +--rw class-id      string
                   |                    +--rw direction?    identityref
                   |                    +--rw rate-limit?   uint8
```

```
                   |                    +--rw latency
                   |                    |  +--rw (flavor)?
                   |                    |     +--:(lowest)
                   |                    |     |  +--rw use-lowest-latency?   emp
ty
                   |                    |     +--:(boundary)
                   |                    |        +--rw latency-boundary?     uin
t16
                   |                    +--rw jitter
                   |                    |  +--rw (flavor)?
                   |                    |     +--:(lowest)
                   |                    |     |  +--rw use-lowest-jitter?   empt
y
                   |                    |     +--:(boundary)
                   |                    |        +--rw latency-boundary?    uint
32
                   |                    +--rw bandwidth
                   |                       +--rw guaranteed-bw-percent    uint8
                   |                       +--rw end-to-end?              empty
                   +--rw vn-attachments
                      +--rw vn-attachment* [vn-id]
                         +--rw vn-id               svc-id
                         +--rw vn-type?            identityref
                         +--rw attachment-point
                            +--rw egress-device-id?     svc-id
                            +--rw address-family?   address-family
                            +--rw address?          inet:ip-address
                            +--rw interfaces
                               +--rw interface?        if:interface-ref
                               +--rw sub-interfaces*   if:interface-ref

  rpcs:
    +---x vn-path-element-compute
    |  +---w input
    |  |  +---w vn-member-list* [vn-member-id]
    |  |     +---w vn-member-id          -> /vn-svc/vn-services/vn-service/vn-id
    |  |     +---w src
    |  |     |  +---w src-address?              -> /vn-svc/sites/site/site-id
    |  |     |  +---w site-network-access-id?
                        -> /vn-svc/sites/site/site-network-accesses/site-network
-access/site-network-access-id
    |  |     +---w dst
    |  |     |  +---w dst-address?              -> /vn-svc/sites/site/site-id
    |  |     |  +---w site-network-access-id?
                        -> /vn-svc/sites/site/site-network-accesses/site-network
-access/site-network-access-id
    |  |     +---w constraint
    |  |     |  +---w path-element* [path-element-id]
    |  |     |     +---w path-element-id    -> /vn-svc/sites/site/site-network-a
ccesses/site-network-access/vn-attachments/vn-attachment/attachment-point/pe-dev
ice-id
    |  |     |     +---w address?           -> /vn-svc/sites/site/site-network-a
ccesses/site-network-access/vn-attachments/vn-attachment/attachment-point/addres
s
    |  |     +---w objective-function?   identityref
    |  |     +---w metric* [metric-type]
    |  |        +---w metric-type     identityref
    |  |        +---w metric-value?   uint32
```

```
    |  +--ro output
    |     +--ro vn-member-list* [vn-member-id]
    |        +--ro vn-member-id    uint32
    |        +--ro src
    |        |  +--ro src-address?              -> /vn-svc/sites/site/site-id
    |        |  +--ro site-network-access-id?   -> /vn-svc/sites/site/site-netwo
rk-accesses/site-network-access/site-network-access-id
    |        +--ro dst
    |        |  +--ro dst-address?              -> /vn-svc/sites/site/site-id
    |        |  +--ro site-network-access-id?   -> /vn-svc/sites/site/site-netwo
rk-accesses/site-network-access/site-network-access-id
    |        +--ro metric* [metric-type]
    |        |  +--ro metric-type     identityref
    |        |  +--ro metric-value?   uint32
    |        +--ro path
    |           +--ro path-element* [path-element-id]
    |              +--ro path-element-id    -> /vn-svc/sites/site/site-network-a
ccesses/site-network-access/vn-attachments/vn-attachment/attachment-point/pe-dev
ice-id
    |              +--ro index?            uint32
    |              +--ro address?          -> /vn-svc/sites/site/site-network-a
ccesses/site-network-access/vn-attachments/vn-attachment/attachment-point/addres
s
    |              +--ro hop-type?          identityref
    +---x vn-network-connectivity-stitch
       +---w input
       |  +---w vn-list* [vn-id]
       |     +---w vn-id                 -> /vn-svc/vn-services/vn-service/vn-id
       |     +---w source-access* [access-id]
       |     |  +---w access-id             -> /vn-svc/sites/site/site-network-a
ccesses/site-network-access/site-network-access-id
       |     |  +---w destination-access* [access-id]
       |     |     +---w access-id    -> /vn-svc/sites/site/site-network-accesse
s/site-network-access/site-network-access-id
       |     +---w objective-function?   identityref
       |     +---w metric* [metric-type]
       |        +---w metric-type     identityref
       |        +---w metric-value?   uint32
       +--ro output
          +--ro vn-access-list* [index]
             +--ro index        uint32
             +--ro source-access -> /vn-svc/sites/site/site-network-accesses/sit
e-network-access/site-network-access-id
             +--ro destination-access-> /vn-svc/sites/site/site-network-accesses
/site-network-access/site-network-access-id
             +--ro multi-domain-network-access-list *
                +--ro domain-id                 svc-id
                +--ro network-access-id         svc-id
```

6. Network Virtualization Overlay Management YANG Module

```
   <CODE BEGINS> file "ietf-vn-rsc@2018-02-03.yang"
   module ietf-vn-rsc {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-vn-rsc";
     prefix vnrsc;

     import ietf-inet-types {
```

```
        prefix inet;
      }
      import ietf-l3vpn-svc {
        prefix l3vpn-svc;
      }
      import ietf-interfaces{
        prefix if;
      }

      organization
        "IETF OPSAWG Working Group.";
      contact
        "WG List: foo@ietf.org
        Editor:  Qin Wu <mailto:bill.wu@huawei.com>
        Editor:  Zitao Wang <mailto:wangzitao@huawei.com>";

      description
        "The YANG module defines a generic service configuration
        model for Layer VN services common across all of the
        vendor implementations.";

    revision 2018-02-03{
    description
    "Initial revision";
    reference
    "A YANG Data Model for VN Service Delivery.";
    }
    /* Features */


    /* Typedefs */
    typedef svc-id {
     type string;
     description
     "Type definition for servicer identifier";
    }
     typedef address-family {
      type enumeration {
       enum ipv4 {
        description
         "IPv4 address family.";
       }
       enum ipv6 {
        description
         "IPv6 address family.";
       }
      }
      description
```

```
      "Defines a type for the address family.";
    }
    /*

   /* Identities */
    identity vn-type {
     description
     "Base identity for VN type";
    }
    identity l2vpn {
     base vn-type;
     description
     "Identity for Layer 2 vpn";
    }
    identity l3vpn {
     base vn-type;
     description
     "Identity for Layer 3 vpn";
    }
    identity evpn {
     base l2vpn;
     description
     "Identity for evpn";
    }
    identity vpls {
     base l2vpn;
     description
     "Identity for vpls";
    }
    identity vpw {
     base l2vpn;
     description
     "Identity for vpw";
    }
    identity vpn-topology {
     description
      "Base identity for VPN topology.";
    }
    identity any-to-any {
     base vpn-topology;
     description
      "Identity for any-to-any VPN topology.";
    }
    identity hub-spoke {
     base vpn-topology;
     description

      "Identity for Hub-and-Spoke VPN topology.";
```

```
    }
    identity hub-spoke-disjoint {
     base vpn-topology;
     description
      "Identity for Hub-and-Spoke VPN topology
       where Hubs cannot communicate with each other.";
    }

    identity objective-function{
     description
     "Identity for objective function";
    }

    identity metric-type{
     description
     "Identity for metric type";
    }

    identity hop-type{
     description
     "Identity for hop-type";
    }
    identity loose{
     base hop-type;
     description
     "loose hop in an explicit path";
    }
    identity strict{
     base hop-type;
     description
     "strict hop in an explicit path";
    }
   /* Grouping */
   grouping vn-service-list {
    list vn-service {
     key "vn-id";
     leaf vn-id {
      type svc-id;
      description
      "VN id";
     }
     leaf customer-name {
      type string;
      description
      "Customer name";
     }
     leaf service-topology {
      type identityref {
```

```
       base vpn-topology;
      }
      default any-to-any;
      description
       "VPN service topology.";
     }
     container site-network-accesses{
      list site-network-access{
       key "site-network-access-id";
       leaf site-network-access-id{
        type svc-id;
        description
        "Site network access identifier";
       }
       description
       "List for site-network access";
      }
      description
      "Container for site network accesses";
     }

     description
     "List for vn service";
    }
    description
    "Grouping for vn service list";
   }
   grouping vn-services-grouping{
    container vn-services{
     uses vn-service-list;
     description
     "Container for virtual network service";
    }
    description
    "Grouping for vn services";
   }

   grouping interfaces-grouping{
    container interfaces{
     leaf interface{
      type if:interface-ref;
      description
      "Base interface";
     }
     leaf-list sub-interfaces{
      type if:interface-ref;
      description
      "Sub interfaces";
```

```
     }
     description
     "Container for interfaces";
    }
    description
    "Grouping for interfaces";
   }

   grouping cpe-device-list{
    list cpe-device{
     key "device-id";
     leaf device-id {
      type svc-id;
      description
      "Device identifier";
     }
     leaf address-family{
      type address-family;
      description
       "Address family used for management. If address-family
       is specified, the address may or may not be specified
      (by the customer).";
     }
     leaf address{
      type inet:ip-address;
      description
      "IP address";
     }
     uses interfaces-grouping;
     description
     "List for devices";
    }
    description
    "Grouping for cpe device list";

   }
   grouping cpe-devices-grouping{
    container cpe-devices{
     uses cpe-device-list;
     description
     "Container for cpe devices";
    }
    description
    "grouping for cpe-devices-grouping";
   }

   grouping bandwidth-grouping {
    leaf svc-input-bandwidth{
```

```
     type uint32;
     description
     "Service input bandwidth";
    }
    leaf svc-output-bandwidth{
     type uint32;
     description
     "Service output bandwidth";
    }
    description
    "Grouping for bandwidth";
   }

   grouping attachment-point-grouping{
    container attachment-point{
      leaf pe-device-id {
      type svc-id;
      description
      "PE Device identifier";
     }
     leaf address-family{
      type address-family;
      description
       "Address family used for management. If address-family
       is specified, the address may or may not be specified
      (by the customer).";
     }
     leaf address{
      type inet:ip-address;
      description
      "IP address";
     }
     uses interfaces-grouping;
     description
     "Container for attachment point";
    }
    description
    "Grouping for attachment points";
   }

   grouping vn-attachment-list{
    list vn-attachment{
     key "vn-id";
     leaf vn-id{
      type svc-id;
      description
      "Virtual network identifier";
     }
```

```
     leaf vn-type{
      type identityref{
       base vn-type;
      }
      description
      "VN type";
     }
     uses attachment-point-grouping;
     description
     "List for VN attachments";
    }
    description
    "Grouping for VN attachment list";
   }

   grouping vn-attachments-grouping{
    container vn-attachments{
     uses vn-attachment-list;
     description
     "Container for VN attachments";
    }
    description
    "Grouping for VN attachments";
   }

   grouping site-network-access-list{
    list site-network-access{
     key "site-network-access-id";
     leaf site-network-access-id{
      type leafref{
       path "/vn-svc/vn-services/vn-service"
    +"/site-network-accesses/site-network-access"
    +"/site-network-access-id";
      }
      description
      "Site network access identifier";
     }
     leaf device-id {
      type leafref{
       path "/vn-svc/sites/site/cpe-devices"
    +"/cpe-device/device-id";
      }
      description
      "Device id";
     }
     uses l3vpn-svc:access-diversity;
     container service {
      uses bandwidth-grouping;
```

```
      leaf svc-mtu {
       type uint16;
    description
    "Service-mtu";
      }
      uses l3vpn-svc:site-service-qos-profile;
      description
      "Container for service";
     }
     uses vn-attachments-grouping;
     description
     "List for site-network access";


    }
    description
    "Grouping for site-network access list";
   }

   grouping site-network-accesses-grouping{
    container site-network-accesses{
     uses site-network-access-list;
     description
     "Container for site network accesses";
    }
    description
    "Grouping for site network accesses";
   }

   grouping site-list-grouping{
    list site {
     key "site-id";
     leaf site-id {
      type svc-id;
      description
      "Site identifier";
     }
     uses cpe-devices-grouping;
     container service {
      uses l3vpn-svc:site-service-qos-profile;
      description
      "Site service";
     }
     uses site-network-accesses-grouping;
     description
     "List for sites";
    }
    description
```

```
    "Grouping for site list";
   }

   grouping sites-grouping {
    container sites{
     uses site-list-grouping;
     description
     "Container for sites";
    }
    description
    "Grouping for sites";
   }

   grouping src-grouping{
      container src{
       leaf src-address{
     type leafref {
      path "/vn-svc/sites/site/site-id";
     }
     description
     "Leaf list for source address";
    }
    leaf site-network-access-id{
     type leafref {
      path "/vn-svc/sites/site/site-network-accesses"+
      "/site-network-access/site-network-access-id";
     }
     description
     "Leaf list for site-network-access id";
    }
       description
    "Container for source id";
      }
    description
    "Grouping for source site";
   }

   grouping dst-grouping{
     container dst{
       leaf dst-address{
     type leafref {
      path "/vn-svc/sites/site/site-id";
     }
     description
     "Leaf list for source address";
    }
    leaf site-network-access-id{
     type leafref {
```

```
      path "/vn-svc/sites/site/site-network-accesses"+
      "/site-network-access/site-network-access-id";
     }
     description
     "Leaf list for site-network-access id";
    }
       description
    "Container for destination id";
      }
    description
    "Grouping for source site";
   }

   grouping objective-function-group{
    leaf objective-function {
     type identityref{
      base objective-function;
     }
     description
     "operational state of the objective function";
     }
     description
     "Grouping for objective functions";
   }


   grouping path-element-list{
    list path-element{
     key "path-element-id";
     leaf path-element-id{
      type leafref{
      path "/vn-svc/sites/site/site-network-accesses"+
      "/site-network-access/vn-attachments/vn-attachment"+
      "/attachment-point/pe-device-id";
      }
      description
      "Path element identifier";
     }
     leaf address{
      type leafref{
      path "/vn-svc/sites/site/site-network-accesses"+
      "/site-network-access/vn-attachments/vn-attachment"+
      "/attachment-point/address";
      }
      description
      "Path element address";
     }
     description
```

```
     "List for path elements";
    }
    description
    "Grouping for path elements";
   }

   grouping constraint-grouping{
    container constraint{
     config false;
     uses path-element-list;
     description
     "Container for constraint";
    }
    description
    "Grouping for constraint";
   }

   grouping metric-grouping{
    list metric {
     key metric-type;
     leaf metric-type {
      type identityref{
       base metric-type;
      }
      description
      "Metric type";
     }
     leaf metric-value {
      type uint32;
      description
      "Metric value";
     }
     description
     "List for metric";
    }
    description
    "Grouping for metric";
   }

   grouping path-list{
    list path-element{
     key "path-element-id";
     leaf path-element-id{
      type leafref{
      path "/vn-svc/sites/site/site-network-accesses"+
      "/site-network-access/vn-attachments/vn-attachment"+
      "/attachment-point/pe-device-id";
      }
```

```
      description
      "Path element identifier";
     }
     leaf index{
      type uint32;
      description
      "Index";
     }
     leaf address{
      type leafref{
      path "/vn-svc/sites/site/site-network-accesses"+
      "/site-network-access/vn-attachments/vn-attachment"+
      "/attachment-point/address";
      }
      description
      "Path element address";
     }
     leaf hop-type{
      type identityref {
       base hop-type;
      }
      description
      "Hop type";
     }
     description
     "List for path elements";
    }
    description
    "Grouping for path list";
   }

   grouping path-grouping{
    container path{
    uses path-list;
    description
    "Container for path";
    }
    description
    "Grouping for path";
   }
   grouping access-grouping{
    list source-access{
     key "access-id";
     leaf access-id {
       type leafref{
     path "/vn-svc/sites/site/site-network-accesses"
     +"/site-network-access/site-network-access-id";
    }
```

```
      description
      "Access id";
     }
     list destination-access{
     key "access-id";
     leaf access-id {
       type leafref{
     path "/vn-svc/sites/site/site-network-accesses"
     +"/site-network-access/site-network-access-id";
    }
      description
      "Access id";
     }
     description
     "List for destination access id";
    }
    description
    "List for source access id";
    }
    description
    "Grouping for access";
   }
   /* .....................................*/

   container vn-svc{
    uses vn-services-grouping;
    uses sites-grouping;
    description
    "Container for vn service";
   }

   rpc vn-compute{
    description
    "RPC for VN compute";
    input {
     list vn-member-list {
      key "vn-member-id";
      leaf vn-member-id{
       type leafref{
     path "/vn-svc/vn-services/vn-service/vn-id";
    }
    description
    "VN member identifier";
      }
      uses src-grouping;
      uses dst-grouping;
      uses constraint-grouping;
      uses objective-function-group;
```

```
      uses metric-grouping;
      description
      "List for vn member";
     }

    }
    output{
     list vn-member-list {
      key "vn-member-id";
      leaf vn-member-id{
       type uint32;
    description
    "VN member identifier";
      }
      uses src-grouping;
      uses dst-grouping;
      uses metric-grouping;
      uses path-grouping;
      description
      "List for vn member";
     }
    }
   }

   rpc vn-stitch{
    description
    "RPC for VN compute";
    input {
     list vn-list {
      key "vn-id";
      leaf vn-id{
       type leafref{
     path "/vn-svc/vn-services/vn-service/vn-id";
    }
    description
    "VN identifier";
      }
      uses access-grouping;
      uses objective-function-group;
      uses metric-grouping;
      description
      "List for vn";
     }

    }
    output{
     list vn-access-list {
      key "index";
```

```
      leaf index{
       type uint32;
    description
    "Index for VN access";
      }
      leaf source-access {
       type leafref{
     path "/vn-svc/sites/site/site-network-accesses"
     +"/site-network-access/site-network-access-id";
    }
    description
    "Source Access ID";
      }
      leaf destination-access {
       type leafref{
     path "/vn-svc/sites/site/site-network-accesses"
     +"/site-network-access/site-network-access-id";
    }
    description
    "Destination Access ID";
      }
      list multi-domain-network-access-list {
       key "domain-id network-access-id";
       leaf domain-id {
       type string;
       description
       "Domain ID";
    }
    leaf network-access-id {
     type leafref{
     path "/vn-svc/sites/site/site-network-accesses"
     +"/site-network-access/site-network-access-id";
     }
      description
     "Network access ID";
    }
      description
      "List for multiple domain network access";
      }
      description
      "List for vn access";
     }
    }
   }
   }
   <CODE ENDS>
```

7. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol [RFC8040] or NETCONF protocol ([RFC6241]). The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH)[RFC6242] . The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /vn-svc/vn-services/vn-service

  The entries in this list include the whole vn service configurations to which the customer subscribed, and indirectly create or modify the egress and ingress device configurations. Unexpected changes to these entries could lead to the service disruption and/or network misbehavior.

- /vn-svc/sites/site

  The entries in this list include the customer site configurations. Unexpected changes to these entries could lead to the service disruption and/or network misbehavior.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /vn-svc/vn-services/vn-service

- /vn-svc/sites/site

The entries in these lists include customer-proprietary or confidential information, e.g., customer-name, site location, what service the customer subscribes.

8. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
--------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-vn-rsc

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.
--------------------------------------------------------------------
```

This document registers a YANG module in the YANG Module Names registry [RFC7950].

```
--------------------------------------------------------------------
   Name:         ietf-vn-rsc
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-vn-rsc
   Prefix:       vnrsc
   Reference:    RFC xxxx
--------------------------------------------------------------------
```

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6370] Bocci, M., Swallow, G., and E. Gray, "MPLS Transport Profile (MPLS-TP) Identifiers", RFC 6370, DOI 10.17487/RFC6370, September 2011, <https://www.rfc-editor.org/info/rfc6370>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

9.2. Informative References

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <https://www.rfc-editor.org/info/rfc3393>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <https://www.rfc-editor.org/info/rfc7297>.

[RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <https://www.rfc-editor.org/info/rfc7679>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <https://www.rfc-editor.org/info/rfc7680>.

Authors’ Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Michael Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com