

Mboned
Internet-Draft
Intended status: Best Current Practice
Expires: September 4, 2018

M. Abrahamsson
T-Systems
T. Chown
Jisc
L. Giuliano
Juniper Networks, Inc.
March 3, 2018

Deprecating ASM for Interdomain Multicast
draft-acg-mboned-deprecate-interdomain-asm-00

Abstract

This document recommends the deprecation of the use of Any-Source Multicast (ASM) for interdomain multicast. It therefore implicitly recommends the use of Source-Specific Multicast (SSM) for interdomain multicast applications, and that hosts and routers that are expected to handle such applications fully support SSM. The recommendations in this document do not preclude the continued use of ASM within a single organisation or domain.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 4, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Multicast routing protocols | 3 |
| 2.1. ASM routing protocols | 3 |
| 2.2. SSM Routing protocols | 4 |
| 3. Discussion | 4 |
| 3.1. Observations on ASM and SSM deployments | 4 |
| 3.2. Advantages of SSM for interdomain multicast | 5 |
| 4. Recommendations | 6 |
| 4.1. Deprecating use of ASM for interdomain multicast | 6 |
| 4.2. Including network support for IGMPv3 / MLDv2 | 6 |
| 4.3. Building application support for SSM | 7 |
| 4.4. Standardising an ASM/SSM protocol mapping mechanism | 7 |
| 4.5. Not filtering ASM addressing between domains | 8 |
| 4.6. Not precluding Intradomain ASM | 8 |
| 5. Security Considerations | 8 |
| 6. IANA Considerations | 8 |
| 7. Acknowledgments | 9 |
| 8. References | 9 |
| 8.1. Normative References | 9 |
| 8.2. Informative References | 10 |
| Authors' Addresses | 11 |

1. Introduction

IP Multicast has been deployed in various forms, both within private networks and on the wider Internet. While a number of service models have been published, and in many cases revised over time, there has been no strong recommendation made on the appropriateness of those models to certain scenarios. This document addresses this gap by making a BCP-level recommendation to deprecate the use of ASM for interdomain multicast, and thus implicitly also that all hosts and routers that are expected to support such multicast applications fully support SSM.

This document does not make any statement on the use of ASM within in a single domain or organisation, and therefore does not preclude its use. Indeed, there may be a number of application contexts for which ASM is currently still considered well-suited within a single domain.

2. Multicast routing protocols

The general IP multicast service model [RFC1112] is that sender(s) send to a multicast group address, receivers express an interest in traffic sent to a given multicast group address, and that routers use multicast routing protocols to determine how to deliver traffic from the sender(s) to the receivers.

Two high-level flavours of this service model have evolved over time. In Any-Source Multicast (ASM), any number of sources may transmit multicast packets, and those sources may come and go over the course of a multicast session without being known a priori. In ASM, receivers express interest only in a given multicast group address, and the multicast routing protocol facilitates source discovery at the network layer. In contrast, with Source-Specific Multicast (SSM) the specific source(s) that may send traffic to the group are known in advance, or may be determined during a session, typically through an out-of-band protocol sitting above the network layer. Thus in SSM, receivers express interest in both a multicast group address and specific associated source address(es).

IANA has reserved specific ranges of IPv4 and IPv6 address space for multicast addressing. Guidelines for IPv4 multicast address assignments can be found in [RFC5771], while guidelines for IPv6 multicast address assignments can be found in [RFC2375] and [RFC3307]. The IPv6 multicast address format is described in [RFC4291].

2.1. ASM routing protocols

The most commonly deployed ASM routing protocol is Protocol Independent Multicast - Sparse Mode, or PIM-SM, as detailed in [RFC7761]. PIM-SM, as the name suggests, was designed to be used in scenarios where the subnets with receivers are sparsely distributed throughout the network. Because it does not know sender addresses in advance, PIM-SM uses the concept of a Rendezvous Point (RP) to 'marry up' senders and receivers, where all routers in a PIM-SM domain are configured to use specific RP(s).

To enable PIM-SM to work between multiple domains, i.e. to allow an RP in one domain to learn the existence of a source in another domain, an inter-RP signalling protocol known as Multicast Source Discovery Protocol (MSDP) [RFC3618] is used. Deployment scenarios for MSDP are given in [RFC4611]. MSDP has remained an Experimental protocol since its publication in 2003, and was not replicated or carried forward for IPv6.

In the absence of MSDP, a new mechanism, Embedded-RP [RFC3956], was defined for IPv6 PIM-SM, which allows routers supporting the protocol to determine the RP for the group without any prior configuration, simply by observing the RP address that is embedded (included) in the IPv6 multicast group address. Embedded-RP allows PIM-SM operation across any IPv6 network in which there is an end-to-end path of routers supporting the protocol.

2.2. SSM Routing protocols

PIM-SSM is detailed in [RFC4607]. In contrast to PIM-SM, PIM-SSM benefits from sender source address(es) being known about in advance, i.e. a given source's IP address is known (by some out of band mechanism), and thus the receiver's router can send a PIM JOIN directly towards the sender, without needing to use an RP.

IPv4 addresses in the 232/8 (232.0.0.0 to 232.255.255.255) range are designated as source-specific multicast (SSM) destination addresses and are reserved for use by source-specific applications and protocols. For IPv6, the address prefix FF3x::/32 is reserved for source-specific multicast use.

3. Discussion

3.1. Observations on ASM and SSM deployments

In enterprise and campus scenarios, ASM in the form of PIM-SM is in relatively common use, and has generally replaced PIM-DM [RFC3973]. The configuration and management of an RP within a single domain is not onerous. However, if interworking with external PIM domains in IPv4 multicast deployments is needed, MSDP is required to exchange information between domain RPs about sources. MSDP remains an Experimental protocol, and can be a complex and fragile protocol to administer and troubleshoot.

PIM-SM is a general purpose protocol that can handle all use cases. In particular, it was designed for cases such as videoconferencing where multiple sources may come and go during a multicast session. But for cases where a single, persistent source is used, and receivers can be configured to know of that source, PIM-SM has unnecessary complexity.

MSDP was not taken forward to IPv6. Instead, IPv6 has Embedded-RP, which allows the RP address for a multicast group to be embedded in the group address, making RP discovery automatic, if all routers on the path between a receiver and a sender support the protocol. Embedded-RP can support lightweight ad-hoc deployments. However, it relies on a single RP for an entire group. Embedded-RP was run

successfully between European and US academic networks during the 6NET project in 2004/05. Its usage generally remains constrained to academic networks.

As stated in RFC 4607, SSM is particularly well-suited to dissemination-style applications with one or more senders whose identities are known (by some mechanism) before the application starts running. PIM-SSM is therefore very well-suited to applications such as classic linear broadcast TV over IP.

SSM requires hosts and their subnet routers using it support the new(er) IGMPv3 [RFC3376] and MLDv2 [RFC3810] protocols. While delayed delivery of support in some OSes has meant that adoption of SSM has also been slower than might have been expected, or hoped, and was a historical reason to use ASM rather than SSM, support for IGMPv3 and MLDv2 is now widespread in common OSes.

3.2. Advantages of SSM for interdomain multicast

A significant benefit of SSM is its reduced complexity through eliminating the network-based source discovery required in ASM. This means there are no RPs, shared trees, Shortest Path Tree (SPT) switchovers, PIM registers, MSDP or data-driven state creation elements to support. SSM is really just a small subset of PIM-SM, plus IGMPv3 / MLDv2.

This reduced complexity makes SSM radically simpler to manage, troubleshoot and operate, particularly for network backbone operators, and this is the main motivation for the recommendation to deprecate the use of ASM in interdomain scenarios. Interdomain ASM is widely viewed as complicated and fragile. By eliminating network-based source discovery for interdomain multicast, the vast majority of the complexity issues go away.

RFC 4607 details many benefits of SSM, including:

- "Elimination of cross-delivery of traffic when two sources simultaneously use the same source-specific destination address;

- Avoidance of the need for inter-host coordination when choosing source-specific addresses, as a consequence of the above;

- Avoidance of many of the router protocols and algorithms that are needed to provide the ASM service model."

Further discussion can also be found in [RFC3569].

SSM is considered more secure in that it supports access control, i.e. you only get packets from the sources you explicitly ask for, as opposed to ASM where anyone can decide to send traffic to a PIM-SM group address. This topic is expanded upon in [RFC4609].

4. Recommendations

4.1. Deprecating use of ASM for interdomain multicast

This document recommends that the use of ASM is deprecated for interdomain multicast, and thus implicitly that hosts and routers that are expected to support such interdomain applications fully support SSM. Best current practices for deploying interdomain multicast using SSM are documented in [RFC8313]

The recommendation applies to the use of ASM between domains where either MSDP (IPv4) or Embedded-RP (IPv6) is required for sharing knowledge of remote sources. It also recommends against the multi-domain use of an ASM group with a single RP in one domain, where multicast tunnels are used between domains.

While MSDP is an Experimental level standard, this document does not propose making MSDP Historic, given its use may be desirable for intradomain multicast use cases.

4.2. Including network support for IGMPv3 / MLDv2

This document recommends that all host and router platforms supporting multicast, and any security appliances that may handle multicast traffic, support IGMPv3 [RFC3376] and MLDv2 [RFC3810]. The updated IPv6 Node Requirements RFC [I-D.ietf-6man-rfc6434-bis] states that MLDv2 support is a MUST in all implementations. Such support is already widespread in common host and router platforms.

Further guidance on IGMPv3 and MLDv2 is given in [RFC4604].

It is sometimes desirable to limit the propagation of multicast messages in a layer 2 network, typically through a layer 2 switch device. In such cases multicast snooping can be used, by which the switch device observes the IGMP/MLD traffic passing through it, and then attempts to make intelligent decisions on which physical ports to forward multicast. Typically, ports that have not expressed an interest in receiving multicast for a given group would not have traffic for that group forwarded through them. Such snooping capability should support IGMPv3 and MLDv2. There is further discussion in [RFC4541].

4.3. Building application support for SSM

There will be a wide range of applications today that only support ASM, whether as software packages, or code embedded in devices such as set-top boxes.

The implicit recommendation to use SSM for interdomain multicast means that applications should use SSM, and operate correctly in an SSM environment, triggering IGMPv3/MLDv2 messages to signal use of SSM.

It is often thought that ASM is required for multicast applications where there are multiple sources. However, RFC 4607 also describes how SSM can be used instead of PIM-SM for multi-party applications:

"SSM can be used to build multi-source applications where all participants' identities are not known in advance, but the multi-source "rendezvous" functionality does not occur in the network layer in this case. Just like in an application that uses unicast as the underlying transport, this functionality can be implemented by the application or by an application-layer library."

Given all common OSes support SSM, it is then down to the programming language and APIs used as to whether the necessary SSM APIs are available. SSM support is generally quite ubiquitous, with the current exception of websockets used in web-browser based applications.

It is desirable that applications also support appropriate congestion control, as described in [RFC8085], with appropriate codecs, to achieve the necessary rate adaption.

Some useful considerations for multicast applications can still be found in the relatively old [RFC3170].

4.4. Standardising an ASM/SSM protocol mapping mechanism

In the case of existing ASM applications that cannot readily be ported to SSM, it may be possible to use some form of protocol mapping, i.e., to have a mechanism to translate a (*,G) join or leave to a (S,G) join or leave, for a specific source, S. The general challenge in performing such mapping is determining where the configured source address, S, comes from.

There are some existing vendor-specific mechanisms to achieve this function, but none are documented in IETF standards. This appears to be a useful area for the IETF to work on, but it should be noted that any such effort would only be an interim transition mechanism, and

such mappings do not remove the requirement for applications to be allocated ASM group addresses for the communications.

4.5. Not filtering ASM addressing between domains

A key benefit of SSM is that the multicast application does not need to be allocated a specific multicast group by the network, rather as SSM is inherently source-specific, it can use any group address, G, in the reserved range of IPv4 or IPv6 SSM addresses for its own source address, S.

In principle, if interdomain ASM is deprecated, backbone operators could begin filtering the ranges of group addresses used by ASM. In practice, this is not recommended given there will be a transition period from ASM to SSM, where some form of ASM-SSM mappings may be used, and filtering may preclude such operations.

4.6. Not precluding Intradomain ASM

The use of ASM within a single multicast domain, such as an enterprise or campus, with an RP for the site, is still relatively common today. The operators of such a site may choose to use Anycast-RP [RFC4610] or MSDP for internal RP resilience, at the expense of the extra complexity in managing that configuration.

This document does not preclude continued use of ASM in the intradomain scenario. If an organisation, or AS, wishes to use multiple multicast domains within its own network border, that is a choice for that organisation to make, and it may then use MSDP or Embedded-RP internally within its own network.

5. Security Considerations

This document adds no new security considerations. RFC 4609 describes the additional security benefits of using SSM instead of ASM.

6. IANA Considerations

This document makes no request of IANA.

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

7. Acknowledgments

The authors would like to thank members of the IETF mboned WG for discussions on the content of this document, with specific thanks to the following people for their contributions to the document: Hitoshi Asaeda, Dale Carder, Toerless Eckert, Jake Holland, Albert Manfredi, Mike McBride, Per Nihlen, Greg Shepherd, James Stevens, Stig Venaas, Nils Warnke, and Sandy Zhang.

8. References

8.1. Normative References

- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC2375] Hinden, R. and S. Deering, "IPv6 Multicast Address Assignments", RFC 2375, DOI 10.17487/RFC2375, July 1998, <<https://www.rfc-editor.org/info/rfc2375>>.
- [RFC3170] Quinn, B. and K. Almeroth, "IP Multicast Applications: Challenges and Solutions", RFC 3170, DOI 10.17487/RFC3170, September 2001, <<https://www.rfc-editor.org/info/rfc3170>>.
- [RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, DOI 10.17487/RFC3307, August 2002, <<https://www.rfc-editor.org/info/rfc3307>>.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, DOI 10.17487/RFC3569, July 2003, <<https://www.rfc-editor.org/info/rfc3569>>.
- [RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <<https://www.rfc-editor.org/info/rfc3618>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.

- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, DOI 10.17487/RFC3956, November 2004, <<https://www.rfc-editor.org/info/rfc3956>>.
- [RFC3973] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, DOI 10.17487/RFC3973, January 2005, <<https://www.rfc-editor.org/info/rfc3973>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol Independent Multicast (PIM)", RFC 4610, DOI 10.17487/RFC4610, August 2006, <<https://www.rfc-editor.org/info/rfc4610>>.
- [RFC5771] Cotton, M., Vegoda, L., and D. Meyer, "IANA Guidelines for IPv4 Multicast Address Assignments", BCP 51, RFC 5771, DOI 10.17487/RFC5771, March 2010, <<https://www.rfc-editor.org/info/rfc5771>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.

8.2. Informative References

- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.
- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <<https://www.rfc-editor.org/info/rfc4604>>.

- [RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, DOI 10.17487/RFC4609, October 2006, <<https://www.rfc-editor.org/info/rfc4609>>.
- [RFC4611] McBride, M., Meylor, J., and D. Meyer, "Multicast Source Discovery Protocol (MSDP) Deployment Scenarios", BCP 121, RFC 4611, DOI 10.17487/RFC4611, August 2006, <<https://www.rfc-editor.org/info/rfc4611>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8313] Tarapore, P., Ed., Sayko, R., Shepherd, G., Eckert, T., Ed., and R. Krishnan, "Use of Multicast across Inter-domain Peering Points", BCP 213, RFC 8313, DOI 10.17487/RFC8313, January 2018, <<https://www.rfc-editor.org/info/rfc8313>>.
- [I-D.ietf-6man-rfc6434-bis] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", draft-ietf-6man-rfc6434-bis-05 (work in progress), February 2018.

Authors' Addresses

Mikael Abrahamsson
T-Systems
Stockholm
Sweden

Email: mikael.abrahamsson@t-systems.se

Tim Chown
Jisc
Lumen House, Library Avenue
Harwell Oxford, Didcot OX11 0SG
United Kingdom

Email: tim.chown@jisc.ac.uk

Lenny Giuliano
Juniper Networks, Inc.
2251 Corporate Park Drive
Hemdon, Virginia 20171
United States

Email: lenny@juniper.net

MBONED
Internet-Draft
Intended status: Informational
Expires: September 1, 2018

M. McBride
Huawei
February 28, 2018

Multicast in the Data Center Overview
draft-ietf-mboned-dc-deploy-02

Abstract

There has been much interest in issues surrounding massive amounts of hosts in the data center. These issues include the prevalent use of IP Multicast within the Data Center. Its important to understand how IP Multicast is being deployed in the Data Center to be able to understand the surrounding issues with doing so. This document provides a quick survey of uses of multicast in the data center and should serve as an aid to further discussion of issues related to large amounts of multicast in the data center.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 1, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 1.1. Requirements Language | 3 |
| 2. Multicast Applications in the Data Center | 3 |
| 2.1. Client-Server Applications | 3 |
| 2.2. Non Client-Server Multicast Applications | 4 |
| 3. L2 Multicast Protocols in the Data Center | 5 |
| 4. L3 Multicast Protocols in the Data Center | 6 |
| 5. Challenges of using multicast in the Data Center | 7 |
| 6. Layer 3 / Layer 2 Topological Variations | 8 |
| 7. Address Resolution | 9 |
| 7.1. Solicited-node Multicast Addresses for IPv6 address resolution | 9 |
| 7.2. Direct Mapping for Multicast address resolution | 9 |
| 8. IANA Considerations | 10 |
| 9. Security Considerations | 10 |
| 10. Acknowledgements | 10 |
| 11. References | 10 |
| 11.1. Normative References | 10 |
| 11.2. Informative References | 10 |
| Author's Address | 10 |

1. Introduction

Data center servers often use IP Multicast to send data to clients or other application servers. IP Multicast is expected to help conserve bandwidth in the data center and reduce the load on servers. IP Multicast is also a key component in several data center overlay solutions. Increased reliance on multicast, in next generation data centers, requires higher performance and capacity especially from the switches. If multicast is to continue to be used in the data center, it must scale well within and between datacenters. There has been much interest in issues surrounding massive amounts of hosts in the data center. There was a lengthy discussion, in the now closed ARMD WG, involving the issues with address resolution for non ARP/ND multicast traffic in data centers. This document provides a quick survey of multicast in the data center and should serve as an aid to further discussion of issues related to multicast in the data center.

ARP/ND issues are not addressed in this document except to explain how address resolution occurs with multicast.

1.1. 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.

2. Multicast Applications in the Data Center

There are many data center operators who do not deploy Multicast in their networks for scalability and stability reasons. There are also many operators for whom multicast is a critical protocol within their network and is enabled on their data center switches and routers. For this latter group, there are several uses of multicast in their data centers. An understanding of the uses of that multicast is important in order to properly support these applications in the ever evolving data centers. If, for instance, the majority of the applications are discovering/signaling each other, using multicast, there may be better ways to support them than using multicast. If, however, the multicasting of data is occurring in large volumes, there is a need for good data center overlay multicast support. The applications either fall into the category of those that leverage L2 multicast for discovery or of those that require L3 support and likely span multiple subnets.

2.1. Client-Server Applications

IPTV servers use multicast to deliver content from the data center to end users. IPTV is typically a one to many application where the hosts are configured for IGMPv3, the switches are configured with IGMP snooping, and the routers are running PIM-SSM mode. Often redundant servers are sending multicast streams into the network and the network is forwarding the data across diverse paths.

Windows Media servers send multicast streaming to clients. Windows Media Services streams to an IP multicast address and all clients subscribe to the IP address to receive the same stream. This allows a single stream to be played simultaneously by multiple clients and thus reducing bandwidth utilization.

Market data relies extensively on IP multicast to deliver stock quotes from the data center to a financial services provider and then to the stock analysts. The most critical requirement of a multicast trading floor is that it be highly available. The network must be designed with no single point of failure and in a way the network can respond in a deterministic manner to any failure. Typically redundant servers (in a primary/backup or live live mode) are sending multicast streams into the network and the network is forwarding the

data across diverse paths (when duplicate data is sent by multiple servers).

With publish and subscribe servers, a separate message is sent to each subscriber of a publication. With multicast publish/subscribe, only one message is sent, regardless of the number of subscribers. In a publish/subscribe system, client applications, some of which are publishers and some of which are subscribers, are connected to a network of message brokers that receive publications on a number of topics, and send the publications on to the subscribers for those topics. The more subscribers there are in the publish/subscribe system, the greater the improvement to network utilization there might be with multicast.

2.2. Non Client-Server Multicast Applications

Routers, running Virtual Routing Redundancy Protocol (VRRP), communicate with one another using a multicast address. VRRP packets are sent, encapsulated in IP packets, to 224.0.0.18. A failure to receive a multicast packet from the master router for a period longer than three times the advertisement timer causes the backup routers to assume that the master router is dead. The virtual router then transitions into an unsteady state and an election process is initiated to select the next master router from the backup routers. This is fulfilled through the use of multicast packets. Backup router(s) are only to send multicast packets during an election process.

Overlays may use IP multicast to virtualize L2 multicasts. IP multicast is used to reduce the scope of the L2-over-UDP flooding to only those hosts that have expressed explicit interest in the frames. VXLAN, for instance, is an encapsulation scheme to carry L2 frames over L3 networks. The VXLAN Tunnel End Point (VTEP) encapsulates frames inside an L3 tunnel. VXLANs are identified by a 24 bit VXLAN Network Identifier (VNI). The VTEP maintains a table of known destination MAC addresses, and stores the IP address of the tunnel to the remote VTEP to use for each. Unicast frames, between VMs, are sent directly to the unicast L3 address of the remote VTEP. Multicast frames are sent to a multicast IP group associated with the VNI. Underlying IP Multicast protocols (PIM-SM/SSM/BIDIR) are used to forward multicast data across the overlay.

The Ganglia application relies upon multicast for distributed discovery and monitoring of computing systems such as clusters and grids. It has been used to link clusters across university campuses and can scale to handle clusters with 2000 nodes

Windows Server, cluster node exchange, relies upon the use of multicast heartbeats between servers. Only the other interfaces in the same multicast group use the data. Unlike broadcast, multicast traffic does not need to be flooded throughout the network, reducing the chance that unnecessary CPU cycles are expended filtering traffic on nodes outside the cluster. As the number of nodes increases, the ability to replace several unicast messages with a single multicast message improves node performance and decreases network bandwidth consumption. Multicast messages replace unicast messages in two components of clustering:

- o Heartbeats: The clustering failure detection engine is based on a scheme whereby nodes send heartbeat messages to other nodes. Specifically, for each network interface, a node sends a heartbeat message to all other nodes with interfaces on that network. Heartbeat messages are sent every 1.2 seconds. In the common case where each node has an interface on each cluster network, there are $N * (N - 1)$ unicast heartbeats sent per network every 1.2 seconds in an N-node cluster. With multicast heartbeats, the message count drops to N multicast heartbeats per network every 1.2 seconds, because each node sends 1 message instead of $N - 1$. This represents a reduction in processing cycles on the sending node and a reduction in network bandwidth consumed.
- o Regroup: The clustering membership engine executes a regroup protocol during a membership view change. The regroup protocol algorithm assumes the ability to broadcast messages to all cluster nodes. To avoid unnecessary network flooding and to properly authenticate messages, the broadcast primitive is implemented by a sequence of unicast messages. Converting the unicast messages to a single multicast message conserves processing power on the sending node and reduces network bandwidth consumption.

Multicast addresses in the 224.0.0.x range are considered link local multicast addresses. They are used for protocol discovery and are flooded to every port. For example, OSPF uses 224.0.0.5 and 224.0.0.6 for neighbor and DR discovery. These addresses are reserved and will not be constrained by IGMP snooping. These addresses are not to be used by any application.

3. L2 Multicast Protocols in the Data Center

The switches, in between the servers and the routers, rely upon igmp snooping to bound the multicast to the ports leading to interested hosts and to L3 routers. A switch will, by default, flood multicast traffic to all the ports in a broadcast domain (VLAN). IGMP snooping is designed to prevent hosts on a local network from receiving traffic for a multicast group they have not explicitly joined. It

provides switches with a mechanism to prune multicast traffic from links that do not contain a multicast listener (an IGMP client). IGMP snooping is a L2 optimization for L3 IGMP.

IGMP snooping, with proxy reporting or report suppression, actively filters IGMP packets in order to reduce load on the multicast router. Joins and leaves heading upstream to the router are filtered so that only the minimal quantity of information is sent. The switch is trying to ensure the router only has a single entry for the group, regardless of how many active listeners there are. If there are two active listeners in a group and the first one leaves, then the switch determines that the router does not need this information since it does not affect the status of the group from the router's point of view. However the next time there is a routine query from the router the switch will forward the reply from the remaining host, to prevent the router from believing there are no active listeners. It follows that in active IGMP snooping, the router will generally only know about the most recently joined member of the group.

In order for IGMP, and thus IGMP snooping, to function, a multicast router must exist on the network and generate IGMP queries. The tables (holding the member ports for each multicast group) created for snooping are associated with the querier. Without a querier the tables are not created and snooping will not work. Furthermore IGMP general queries must be unconditionally forwarded by all switches involved in IGMP snooping. Some IGMP snooping implementations include full querier capability. Others are able to proxy and retransmit queries from the multicast router.

In source-only networks, however, which presumably describes most data center networks, there are no IGMP hosts on switch ports to generate IGMP packets. Switch ports are connected to multicast source ports and multicast router ports. The switch typically learns about multicast groups from the multicast data stream by using a type of source only learning (when only receiving multicast data on the port, no IGMP packets). The switch forwards traffic only to the multicast router ports. When the switch receives traffic for new IP multicast groups, it will typically flood the packets to all ports in the same VLAN. This unnecessary flooding can impact switch performance.

4. L3 Multicast Protocols in the Data Center

There are three flavors of PIM used for Multicast Routing in the Data Center: PIM-SM [RFC4601], PIM-SSM [RFC4607], and PIM-BIDIR [RFC5015]. SSM provides the most efficient forwarding between sources and receivers and is most suitable for one to many types of multicast applications. State is built for each S,G channel therefore the more

sources and groups there are, the more state there is in the network. BIDIR is the most efficient shared tree solution as one tree is built for all S,G's, therefore saving state. But it is not the most efficient in forwarding path between sources and receivers. SSM and BIDIR are optimizations of PIM-SM. PIM-SM is still the most widely deployed multicast routing protocol. PIM-SM can also be the most complex. PIM-SM relies upon a RP (Rendezvous Point) to set up the multicast tree and then will either switch to the SPT (shortest path tree), similar to SSM, or stay on the shared tree (similar to BIDIR). For massive amounts of hosts sending (and receiving) multicast, the shared tree (particularly with PIM-BIDIR) provides the best potential scaling since no matter how many multicast sources exist within a VLAN, the tree number stays the same. IGMP snooping, IGMP proxy, and PIM-BIDIR have the potential to scale to the huge scaling numbers required in a data center.

5. Challenges of using multicast in the Data Center

Data Center environments may create unique challenges for IP Multicast. Data Center networks required a high amount of VM traffic and mobility within and between DC networks. DC networks have large numbers of servers. DC networks are often used with cloud orchestration software. DC networks often use IP Multicast in their unique environments. This section looks at the challenges of using multicast within the challenging data center environment.

When IGMP/MLD Snooping is not implemented, ethernet switches will flood multicast frames out of all switch-ports, which turns the traffic into something more like a broadcast.

VRRP uses multicast heartbeat to communicate between routers. The communication between the host and the default gateway is unicast. The multicast heartbeat can be very chatty when there are thousands of VRRP pairs with sub-second heartbeat calls back and forth.

Link-local multicast should scale well within one IP subnet particularly with a large layer3 domain extending down to the access or aggregation switches. But if multicast traverses beyond one IP subnet, which is necessary for an overlay like VXLAN, you could potentially have scaling concerns. If using a VXLAN overlay, it is necessary to map the L2 multicast in the overlay to L3 multicast in the underlay or do head end replication in the overlay and receive duplicate frames on the first link from the router to the core switch. The solution could be to run potentially thousands of PIM messages to generate/maintain the required multicast state in the IP underlay. The behavior of the upper layer, with respect to broadcast/multicast, affects the choice of head end (*,G) or (S,G) replication in the underlay, which affects the opex and capex of the

entire solution. A VXLAN, with thousands of logical groups, maps to head end replication in the hypervisor or to IGMP from the hypervisor and then PIM between the TOR and CORE 'switches' and the gateway router.

Requiring IP multicast (especially PIM BIDIR) from the network can prove challenging for data center operators especially at the kind of scale that the VXLAN/NVGRE proposals require. This is also true when the L2 topological domain is large and extended all the way to the L3 core. In data centers with highly virtualized servers, even small L2 domains may spread across many server racks (i.e. multiple switches and router ports).

It's not uncommon for there to be 10-20 VMs per server in a virtualized environment. One vendor reported a customer requesting a scale to 400VM's per server. For multicast to be a viable solution in this environment, the network needs to be able to scale to these numbers when these VMs are sending/receiving multicast.

A lot of switching/routing hardware has problems with IP Multicast, particularly with regards to hardware support of PIM-BIDIR.

Sending L2 multicast over a campus or data center backbone, in any sort of significant way, is a new challenge enabled for the first time by overlays. There are interesting challenges when pushing large amounts of multicast traffic through a network, and have thus far been dealt with using purpose-built networks. While the overlay proposals have been careful not to impose new protocol requirements, they have not addressed the issues of performance and scalability, nor the large-scale availability of these protocols.

There is an unnecessary multicast stream flooding problem in the link layer switches between the multicast source and the PIM First Hop Router (FHR). The IGMP-Snooping Switch will forward multicast streams to router ports, and the PIM FHR must receive all multicast streams even if there is no request from receiver. This often leads to waste of switch cache and link bandwidth when the multicast streams are not actually required. [I-D.pim-umf-problem-statement] details the problem and defines design goals for a generic mechanism to restrain the unnecessary multicast stream flooding.

6. Layer 3 / Layer 2 Topological Variations

As discussed in RFC6820, the ARMD problems statement, there are a variety of topological data center variations including L3 to Access Switches, L3 to Aggregation Switches, and L3 in the Core only. Further analysis is needed in order to understand how these variations affect IP Multicast scalability

7. Address Resolution

7.1. Solicited-node Multicast Addresses for IPv6 address resolution

Solicited-node Multicast Addresses are used with IPv6 Neighbor Discovery to provide the same function as the Address Resolution Protocol (ARP) in IPv4. ARP uses broadcasts, to send an ARP Requests, which are received by all end hosts on the local link. Only the host being queried responds. However, the other hosts still have to process and discard the request. With IPv6, a host is required to join a Solicited-Node multicast group for each of its configured unicast or anycast addresses. Because a Solicited-node Multicast Address is a function of the last 24-bits of an IPv6 unicast or anycast address, the number of hosts that are subscribed to each Solicited-node Multicast Address would typically be one (there could be more because the mapping function is not a 1:1 mapping). Compared to ARP in IPv4, a host should not need to be interrupted as often to service Neighbor Solicitation requests.

7.2. Direct Mapping for Multicast address resolution

With IPv4 unicast address resolution, the translation of an IP address to a MAC address is done dynamically by ARP. With multicast address resolution, the mapping from a multicast IP address to a multicast MAC address is derived from direct mapping. In IPv4, the mapping is done by assigning the low-order 23 bits of the multicast IP address to fill the low-order 23 bits of the multicast MAC address. When a host joins an IP multicast group, it instructs the data link layer to receive frames that match the MAC address that corresponds to the IP address of the multicast group. The data link layer filters the frames and passes frames with matching destination addresses to the IP module. Since the mapping from multicast IP address to a MAC address ignores 5 bits of the IP address, groups of 32 multicast IP addresses are mapped to the same MAC address. As a result a multicast MAC address cannot be uniquely mapped to a multicast IPv4 address. Planning is required within an organization to select IPv4 groups that are far enough away from each other as to not end up with the same L2 address used. Any multicast address in the [224-239].0.0.x and [224-239].128.0.x ranges should not be considered. When sending IPv6 multicast packets on an Ethernet link, the corresponding destination MAC address is a direct mapping of the last 32 bits of the 128 bit IPv6 multicast address into the 48 bit MAC address. It is possible for more than one IPv6 Multicast address to map to the same 48 bit MAC address.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

No new security considerations result from this document

10. Acknowledgements

The authors would like to thank the many individuals who contributed opinions on the ARMD wg mailing list about this topic: Linda Dunbar, Anoop Ghanwani, Peter Ashwoodsmith, David Allan, Aldrin Isaac, Igor Gashinsky, Michael Smith, Patrick Frejborg, Joel Jaeggli and Thomas Narten.

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>>.

11.2. Informative References

[RFC6820] Narten, T., Karir, M., and I. Foo, "Address Resolution Problems in Large Data Center Networks", RFC 6820, DOI 10.17487/RFC6820, January 2013, <<https://www.rfc-editor.org/info/rfc6820>>.

Author's Address

Mike McBride
Huawei

Email: michael.mcbride@huawei.com

MBONED Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 23, 2018

H. Asaeda
NICT
K. Meyer

W. Lee, Ed.
December 20, 2017

Mtrace Version 2: Traceroute Facility for IP Multicast
draft-ietf-mboned-mtrace-v2-22

Abstract

This document describes the IP multicast traceroute facility, named Mtrace version 2 (Mtrace2). Unlike unicast traceroute, Mtrace2 requires special implementations on the part of routers. This specification describes the required functionality in multicast routers, as well as how an Mtrace2 client invokes a query and receives a reply.

Status of This Memo

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

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

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

This Internet-Draft will expire on June 23, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

| | |
|---|----|
| 1. Introduction | 4 |
| 2. Terminology | 6 |
| 2.1. Definitions | 6 |
| 3. Packet Formats | 7 |
| 3.1. Mtrace2 TLV format | 8 |
| 3.2. Defined TLVs | 8 |
| 3.2.1. Mtrace2 Query | 9 |
| 3.2.2. Mtrace2 Request | 11 |
| 3.2.3. Mtrace2 Reply | 11 |
| 3.2.4. IPv4 Mtrace2 Standard Response Block | 12 |
| 3.2.5. IPv6 Mtrace2 Standard Response Block | 16 |
| 3.2.6. Mtrace2 Augmented Response Block | 18 |
| 3.2.7. Mtrace2 Extended Query Block | 19 |
| 4. Router Behavior | 20 |
| 4.1. Receiving Mtrace2 Query | 20 |
| 4.1.1. Query Packet Verification | 20 |
| 4.1.2. Query Normal Processing | 21 |
| 4.2. Receiving Mtrace2 Request | 21 |
| 4.2.1. Request Packet Verification | 21 |
| 4.2.2. Request Normal Processing | 22 |
| 4.3. Forwarding Mtrace2 Request | 24 |
| 4.3.1. Destination Address | 24 |
| 4.3.2. Source Address | 24 |
| 4.3.3. Appending Standard Response Block | 24 |
| 4.4. Sending Mtrace2 Reply | 25 |
| 4.4.1. Destination Address | 25 |
| 4.4.2. Source Address | 25 |
| 4.4.3. Appending Standard Response Block | 25 |
| 4.5. Proxying Mtrace2 Query | 25 |
| 4.6. Hiding Information | 26 |

| | | |
|--------|--|----|
| 5. | Client Behavior | 26 |
| 5.1. | Sending Mtrace2 Query | 26 |
| 5.1.1. | Destination Address | 27 |
| 5.1.2. | Source Address | 27 |
| 5.2. | Determining the Path | 27 |
| 5.3. | Collecting Statistics | 27 |
| 5.4. | Last Hop Router (LHR) | 27 |
| 5.5. | First Hop Router (FHR) | 28 |
| 5.6. | Broken Intermediate Router | 28 |
| 5.7. | Non-Supported Router | 28 |
| 5.8. | Mtrace2 Termination | 28 |
| 5.8.1. | Arriving at Source | 28 |
| 5.8.2. | Fatal Error | 29 |
| 5.8.3. | No Upstream Router | 29 |
| 5.8.4. | Reply Timeout | 29 |
| 5.9. | Continuing after an Error | 29 |
| 6. | Protocol-Specific Considerations | 29 |
| 6.1. | PIM-SM | 30 |
| 6.2. | Bi-Directional PIM | 30 |
| 6.3. | PIM-DM | 30 |
| 6.4. | IGMP/MLD Proxy | 30 |
| 7. | Problem Diagnosis | 31 |
| 7.1. | Forwarding Inconsistencies | 31 |
| 7.2. | TTL or Hop Limit Problems | 31 |
| 7.3. | Packet Loss | 31 |
| 7.4. | Link Utilization | 32 |
| 7.5. | Time Delay | 32 |
| 8. | IANA Considerations | 32 |
| 8.1. | "Mtrace2 Forwarding Codes" Registry | 32 |
| 8.2. | "Mtrace2 TLV Types" registry | 32 |
| 8.3. | UDP Destination Port | 33 |
| 9. | Security Considerations | 33 |
| 9.1. | Addresses in Mtrace2 Header | 33 |
| 9.2. | Filtering of Clients | 33 |
| 9.3. | Topology Discovery | 33 |
| 9.4. | Characteristics of Multicast Channel | 33 |
| 9.5. | Limiting Query/Request Rates | 33 |
| 9.6. | Limiting Reply Rates | 34 |
| 10. | Acknowledgements | 34 |
| 11. | References | 34 |
| 11.1. | Normative References | 34 |
| 11.2. | Informative References | 35 |
| | Authors' Addresses | 35 |

1. Introduction

Given a multicast distribution tree, tracing from a multicast source to a receiver is difficult, since we do not know on which branch of the multicast tree the receiver lies. This means that we have to flood the whole tree to find the path from a source to a receiver. On the other hand, walking up the tree from a receiver to a source is easy, as most existing multicast routing protocols know the upstream router for each source. Tracing from a receiver to a source can involve only the routers on the direct path.

This document specifies the multicast traceroute facility named Mtrace version 2 or Mtrace2 which allows the tracing of an IP multicast routing path. Mtrace2 is usually initiated from an Mtrace2 client by sending an Mtrace2 Query to a Last Hop Router (LHR) or to a Rendezvous Point (RP). The RP is a special router where sources and receivers meet in Protocol Independent Multicast - Sparse Mode (PIM-SM) [5]. From the LHR/RP receiving the query, the tracing is directed towards a specified source if a source address is specified and source specific state exists on the receiving router. If no source address is specified or if no source specific state exists on a receiving LHR, the tracing is directed toward the RP for the specified group address. Moreover, Mtrace2 provides additional information such as the packet rates and losses, as well as other diagnostic information. Mtrace2 is primarily intended for the following purposes:

- o To trace the path that a packet would take from a source to a receiver.
- o To isolate packet loss problems (e.g., congestion).
- o To isolate configuration problems (e.g., Time to live (TTL) threshold).

Figure 1 shows a typical case on how Mtrace2 is used. First-hop router (FHR) represents the first-hop router, LHR represents the last-hop router (LHR), and the arrow lines represent the Mtrace2 messages that are sent from one node to another. The numbers before the Mtrace2 messages represent the sequence of the messages that would happen. Source, Receiver and Mtrace2 client are typically hosts.

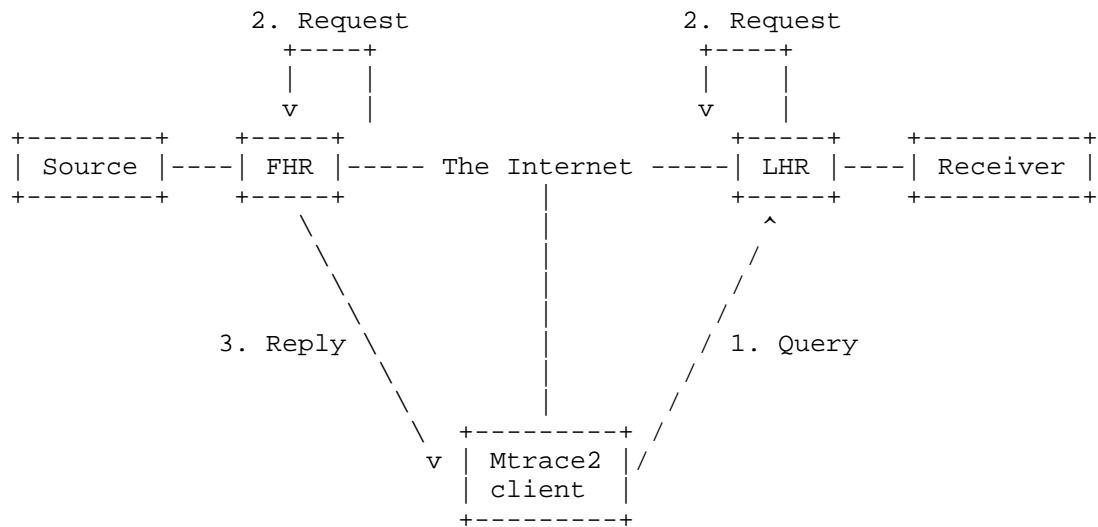


Figure 1

When an Mtrace2 client initiates a multicast trace, it sends an Mtrace2 Query packet to the LHR or RP for a multicast group and, optionally, a source address. The LHR/RP turns the Query packet into a Request. The Request message type enables each of the upstream routers processing the message to apply different packet and message validation rules than those required for handling of a Query message. The LHR/RP then appends a standard response block containing its interface addresses and packet statistics to the Request packet, then forwards the packet towards the source/RP. The Request packet is either unicasted to its upstream router towards the source/RP, or multicasted to the group if the upstream router's IP address is not known. In a similar fashion, each router along the path to the source/RP appends a standard response block to the end of the Request packet before forwarding it to its upstream router. When the FHR receives the Request packet, it appends its own standard response block, turns the Request packet into a Reply, and unicasts the Reply back to the Mtrace2 client.

The Mtrace2 Reply may be returned before reaching the FHR under some circumstances. This can happen if a Request packet is received at an RP or gateway, or when any of several types of error or exception conditions occur which prevent sending of a request to the next upstream router.

The Mtrace2 client waits for the Mtrace2 Reply message and displays the results. When not receiving an Mtrace2 Reply message due to network congestion, a broken router (see Section 5.6), or a non-

responding router (see Section 5.7), the Mtrace2 client may resend another Mtrace2 Query with a lower hop count (see Section 3.2.1), and repeat the process until it receives an Mtrace2 Reply message. The details are Mtrace2 client specific and outside the scope of this document.

Note that when a router's control plane and forwarding plane are out of sync, the Mtrace2 Requests might be forwarded based on the control states instead. In this case, the traced path might not represent the real path the data packets would follow.

Mtrace2 supports both IPv4 and IPv6. Unlike the previous version of Mtrace, which implements its query and response as Internet Group Management Protocol (IGMP) messages [8], all Mtrace2 messages are UDP-based. Although the packet formats of IPv4 and IPv6 Mtrace2 are different because of the address families, the syntax between them is similar.

This document describes the base specification of Mtrace2 that can serve as a basis for future proposals such as Mtrace2 for Automatic Multicast Tunneling (AMT) [9] and Mtrace2 for Multicast in MPLS/BGP IP VPNs (MVPN) [10]. They are therefore out of the scope of this document.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1], and indicate requirement levels for compliant Mtrace2 implementations.

2.1. Definitions

Since Mtrace2 Queries and Requests flow in the opposite direction to the data flow, we refer to "upstream" and "downstream" with respect to data, unless explicitly specified.

Incoming interface

The interface on which data is expected to arrive from the specified source and group.

Outgoing interface

This is one of the interfaces to which data from the source or RP is expected to be transmitted for the specified source and group. It is also the interface on which the Mtrace2 Request was received.

Upstream router

The router, connecting to the Incoming interface of the current router, which is responsible for forwarding data for the specified source and group to the current router.

First-hop router (FHR)

The router that is directly connected to the source the Mtrace2 Query specifies.

Last-hop router (LHR)

A router that is directly connected to a receiver. It is also the router that receives the Mtrace2 Query from an Mtrace2 client.

Group state

It is the state a shared-tree protocol, such as PIM-SM [5], uses to choose the upstream router towards the RP for the specified group. In this state, source-specific state is not available for the corresponding group address on the router.

Source-specific state

It is the state that is used to choose the path towards the source for the specified source and group.

ALL-[protocol]-ROUTERS group

It is a link-local multicast address for multicast routers to communicate with their adjacent routers that are running the same routing protocol. For instance, the IPv4 'ALL-PIM-ROUTERS' group is '224.0.0.13', and the IPv6 'ALL-PIM-ROUTERS' group is 'ff02::d' [5].

3. Packet Formats

This section describes the details of the packet formats for Mtrace2 messages.

All Mtrace2 messages are encoded in the Type/Length/Value (TLV) format (see Section 3.1). The first TLV of a message is a message header TLV specifying the type of message and additional context information required for processing of the message and for parsing of subsequent TLVs in the message. Subsequent TLVs in a message, referred to as Blocks, are appended after the header TLV to provide additional information associated with the message. If an implementation receives an unknown TLV type for the first TLV in a message (i.e., the header TLV), it SHOULD ignore and silently discard the entire packet. If an implementation receives an unknown TLV type for a subsequent TLV within a message, it SHOULD ignore and silently discard the entire packet. If the length of a TLV exceeds the available space in the containing packet, the implementation MUST

ignore and silently discard the TLV and any remaining portion of the containing packet.

All Mtrace2 messages are UDP packets. For IPv4, Mtrace2 Query and Request messages MUST NOT be fragmented. For IPv6, the packet size for the Mtrace2 messages MUST NOT exceed 1280 bytes, which is the smallest Maximum Transmission Unit (MTU) for an IPv6 interface [2]. The source port is uniquely selected by the local host operating system. The destination port is the IANA reserved Mtrace2 port number (see Section 8). All Mtrace2 messages MUST have a valid UDP checksum.

Additionally, Mtrace2 supports both IPv4 and IPv6, but not mixed. For example, if an Mtrace2 Query or Request message arrives in as an IPv4 packet, all addresses specified in the Mtrace2 messages MUST be IPv4 as well. Same rule applies to IPv6 Mtrace2 messages.

3.1. Mtrace2 TLV 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
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Type           |           Length           | Value ... |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Type: 8 bits

Describes the format of the Value field. For all the available types, please see Section 3.2

Length: 16 bits

Length of Type, Length, and Value fields in octets. Minimum length required is 3 octets. The maximum TLV length is not defined; however the entire Mtrace2 packet length SHOULD NOT exceed the available MTU.

Value: variable length

The format is based on the Type value. The length of the value field is Length field minus 3. All reserved fields in the Value field MUST be transmitted as zeros and ignored on receipt.

3.2. Defined TLVs

The following TLV Types are defined:

| Code | Type |
|------|----------------------------------|
| ==== | ===== |
| 0x01 | Mtrace2 Query |
| 0x02 | Mtrace2 Request |
| 0x03 | Mtrace2 Reply |
| 0x04 | Mtrace2 Standard Response Block |
| 0x05 | Mtrace2 Augmented Response Block |
| 0x06 | Mtrace2 Extended Query Block |

Each Mtrace2 message MUST begin with either a Query, Request or Reply TLV. The first TLV determines the type of each Mtrace2 message. Following a Query TLV, there can be a sequence of optional Extended Query Blocks. In the case of a Request or a Reply TLV, it is then followed by a sequence of Standard Response Blocks, each from a multicast router on the path towards the source or the RP. In the case more information is needed, a Standard Response Block can be followed by one or multiple Augmented Response Blocks.

We will describe each message type in detail in the next few sections.

3.2.1. Mtrace2 Query

An Mtrace2 Query is usually originated by an Mtrace2 client which sends an Mtrace2 Query message to the LHR. When tracing towards the source or the RP, the intermediate routers MUST NOT modify the Query message except the Type field. If the actual number of hops is not known, an Mtrace2 client could send an initial Query message with a large # Hops (e.g., 0xffffffff), in order to try to trace the full path.

An Mtrace2 Query message is shown as follows:

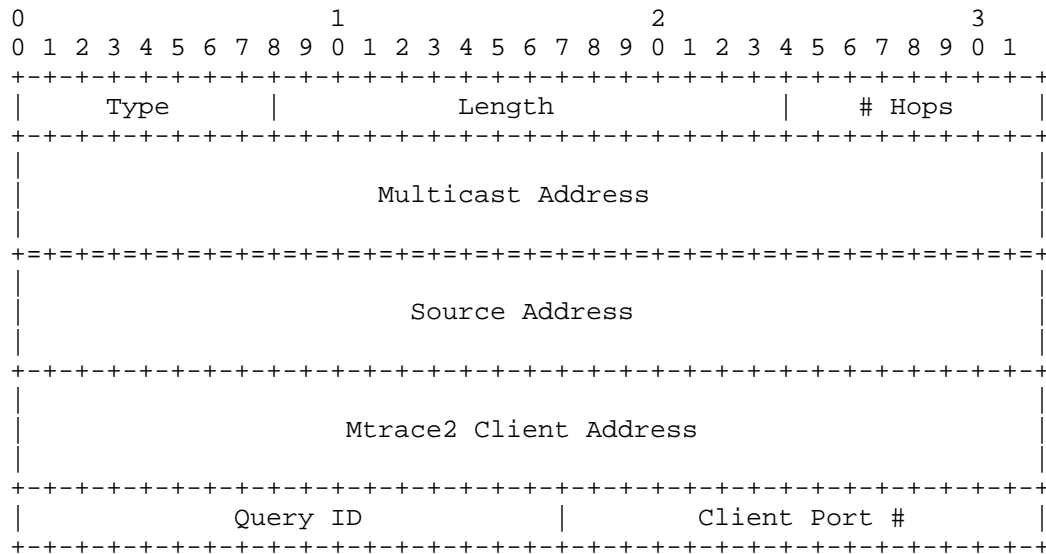


Figure 2

Hops: 8 bits

This field specifies the maximum number of hops that the Mtrace2 client wants to trace. If there are some error conditions in the middle of the path that prevent an Mtrace2 Reply from being received by the client, the client MAY issue another Mtrace2 Query with a lower number of hops until it receives a Reply.

Multicast Address: 32 bits or 128 bits

This field specifies an IPv4 or IPv6 address, which can be either:

m-1: a multicast group address to be traced; or,

m-2: all 1's in case of IPv4 or the unspecified address (::) in case of IPv6 if no group-specific information is desired.

Source Address: 32 bits or 128 bits

This field specifies an IPv4 or IPv6 address, which can be either:

s-1: a unicast address of the source to be traced; or,

s-2: all 1's in case of IPv4 or the unspecified address (::) in case of IPv6 if no source-specific information is desired. For example, the client is tracing a (*,g) group state.

Note that it is invalid to have a source-group combination of (s-2, m-2). If a router receives such combination in an Mtrace2 Query, it MUST silently discard the Query.

Mtrace2 Client Address: 32 bits or 128 bits

This field specifies the Mtrace2 client's IPv4 address or IPv6 global address. This address MUST be a valid unicast address, and therefore, MUST NOT be all 1's or an unspecified address. The Mtrace2 Reply will be sent to this address.

Query ID: 16 bits

This field is used as a unique identifier for this Mtrace2 Query so that duplicate or delayed Reply messages may be detected.

Client Port #: 16 bits

This field specifies the destination UDP port number for receiving the Mtrace2 Reply packet.

3.2.2. Mtrace2 Request

The format of an Mtrace2 Request message is similar to an Mtrace2 Query except the Type field is 0x02.

When a LHR receives an Mtrace2 Query message, it would turn the Query into a Request by changing the Type field of the Query from 0x01 to 0x02. The LHR would then append an Mtrace2 Standard Response Block (see Section 3.2.4) of its own to the Request message before sending it upstream. The upstream routers would do the same without changing the Type field until one of them is ready to send a Reply.

3.2.3. Mtrace2 Reply

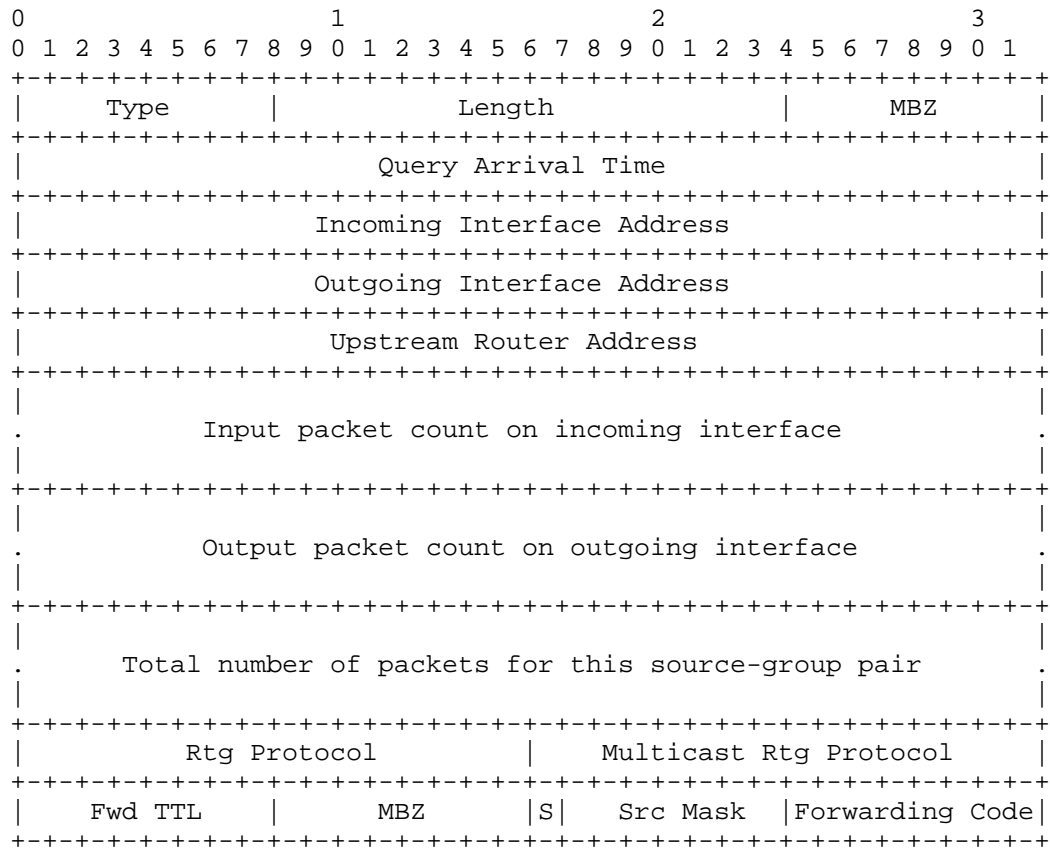
The format of an Mtrace2 Reply message is similar to an Mtrace2 Query except the Type field is 0x03.

When a FHR or a RP receives an Mtrace2 Request message which is destined to itself, it would append an Mtrace2 Standard Response Block (see Section 3.2.4) of its own to the Request message. Next, it would turn the Request message into a Reply by changing the Type field of the Request from 0x02 to 0x03. The Reply message would then be unicasted to the Mtrace2 client specified in the Mtrace2 Client Address field.

There are a number of cases in which an intermediate router might return a Reply before a Request reaches the FHR or the RP. See Section 4.1.1, Section 4.2.2, Section 4.3.3, and Section 4.5 for more details.

3.2.4. IPv4 Mtrace2 Standard Response Block

This section describes the message format of an IPv4 Mtrace2 Standard Response Block. The Type field is 0x04.



MBZ: 8 bits

This field MUST be zeroed on transmission and ignored on reception.

Query Arrival Time: 32 bits

The Query Arrival Time is a 32-bit Network Time Protocol (NTP) timestamp specifying the arrival time of the Mtrace2 Query or Request packet at this router. The 32-bit form of an NTP timestamp consists of the middle 32 bits of the full 64-bit form; that is, the low 16 bits of the integer part and the high 16 bits of the fractional part.

The following formula converts from a timespec (fractional part in nanoseconds) to a 32-bit NTP timestamp:

```
query_arrival_time
= ((tv.tv_sec + 32384) << 16) + ((tv.tv_nsec << 7) / 1953125)
```

The constant 32384 is the number of seconds from Jan 1, 1900 to Jan 1, 1970 truncated to 16 bits. $((tv.tv_nsec << 7) / 1953125)$ is a reduction of $((tv.tv_nsec / 1000000000) << 16)$.

Note that Mtrace2 does not require all the routers on the path to have synchronized clocks in order to measure one-way latency.

Additionally, Query Arrival Time is useful for measuring the packet rate. For example, suppose that a client issues two queries, and the corresponding requests R1 and R2 arrive at router X at time T1 and T2, then the client would be able to compute the packet rate on router X by using the packet count information stored in the R1 and R2, and the time T1 and T2.

Incoming Interface Address: 32 bits

This field specifies the address of the interface on which packets from the source or the RP are expected to arrive, or 0 if unknown or unnumbered.

Outgoing Interface Address: 32 bits

This field specifies the address of the interface on which packets from the source or the RP are expected to transmit towards the receiver, or 0 if unknown or unnumbered. This is also the address of the interface on which the Mtrace2 Query or Request arrives.

Upstream Router Address: 32 bits

This field specifies the address of the upstream router from which this router expects packets from this source. This may be a multicast group (e.g., ALL-[protocol]-ROUTERS group) if the upstream router is not known because of the workings of the multicast routing protocol. However, it should be 0 if the incoming interface address is unknown or unnumbered.

Input packet count on incoming interface: 64 bits

This field contains the number of multicast packets received for all groups and sources on the incoming interface, or all 1's if no count can be reported. This counter may have the same value as ifHCInMulticastPkts from the Interfaces Group MIB (IF-MIB) [12] for this interface.

Output packet count on outgoing interface: 64 bit

This field contains the number of multicast packets that have been transmitted or queued for transmission for all groups and sources on the outgoing interface, or all 1's if no count can be reported. This counter may have the same value as ifHCOutMulticastPkts from the IF-MIB [12] for this interface.

Total number of packets for this source-group pair: 64 bits

This field counts the number of packets from the specified source forwarded by the router to the specified group, or all 1's if no count can be reported. If the S bit is set (see below), the count is for the source network, as specified by the Src Mask field (see below). If the S bit is set and the Src Mask field is 127, indicating no source-specific state, the count is for all sources sending to this group. This counter should have the same value as ipMcastRoutePkts from the IP Multicast MIB [13] for this forwarding entry.

Rtg Protocol: 16 bits

This field describes the unicast routing protocol running between this router and the upstream router, and it is used to determine the RPF interface for the specified source or RP. This value should have the same value as ipMcastRouteRtProtocol from the IP Multicast MIB [13] for this entry. If the router is not able to obtain this value, all 0's must be specified.

Multicast Rtg Protocol: 16 bits

This field describes the multicast routing protocol in use between the router and the upstream router. This value should have the same value as ipMcastRouteProtocol from the IP Multicast MIB [13] for this entry. If the router cannot obtain this value, all 0's must be specified.

Fwd TTL: 8 bits

This field contains the configured multicast TTL threshold, if any, of the outgoing interface.

S: 1 bit

If this bit is set, it indicates that the packet count for the source-group pair is for the source network, as determined by masking the source address with the Src Mask field.

Src Mask: 7 bits

This field contains the number of 1's in the netmask the router has for the source (i.e. a value of 24 means the netmask is 0xfffff00). If the router is forwarding solely on group state, this field is set to 127 (0x7f).

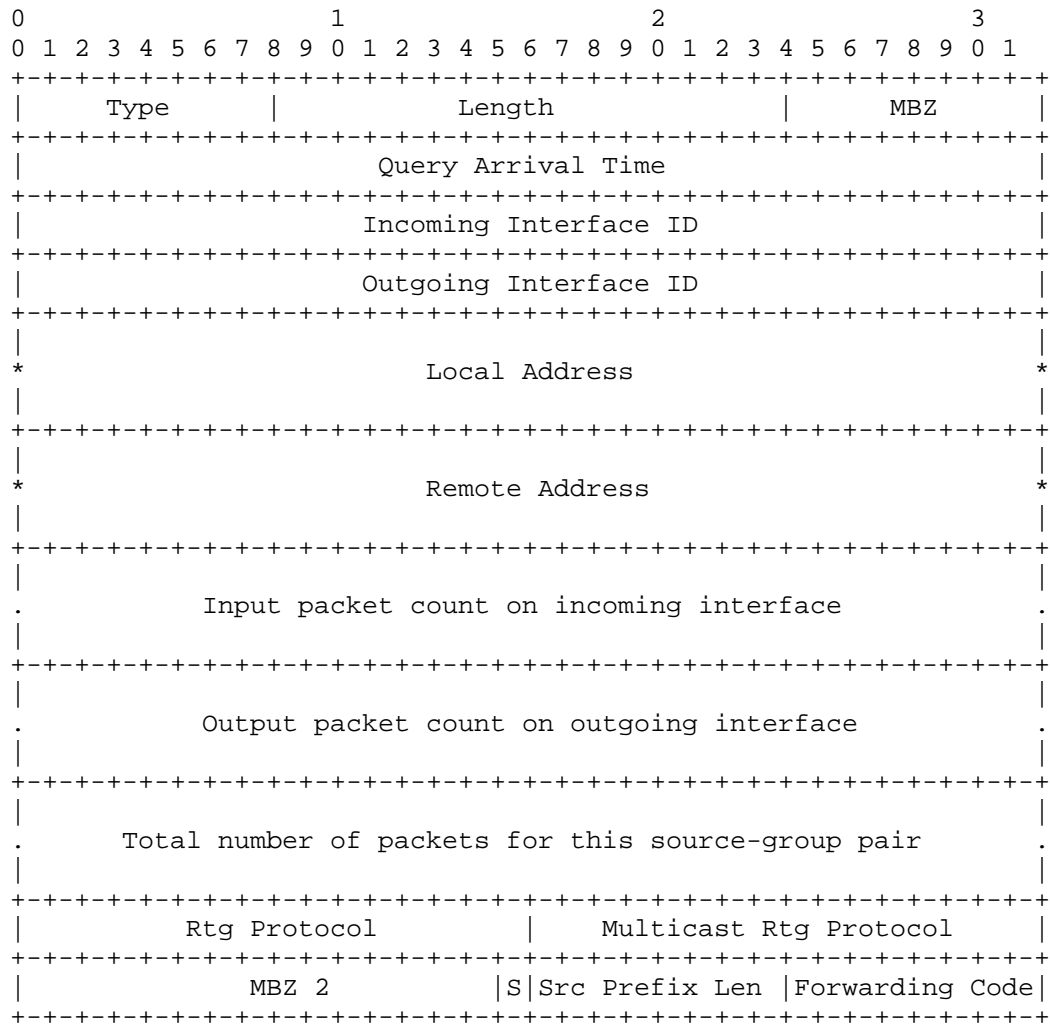
Forwarding Code: 8 bits

This field contains a forwarding information/error code. Values with the high order bit set (0x80-0xff) are intended for use as error or exception codes. Section 4.1 and Section 4.2 explain how and when the Forwarding Code is filled. Defined values are as follows:

| Value | Name | Description |
|-------|----------------|---|
| 0x00 | NO_ERROR | No error |
| 0x01 | WRONG_IF | Mtrace2 Request arrived on an interface to which this router would not forward for the specified group towards the source or RP. |
| 0x02 | PRUNE_SENT | This router has sent a prune upstream which applies to the source and group in the Mtrace2 Request. |
| 0x03 | PRUNE_RCVD | This router has stopped forwarding for this source and group in response to a request from the downstream router. |
| 0x04 | SCOPED | The group is subject to administrative scoping at this router. |
| 0x05 | NO_ROUTE | This router has no route for the source or group and no way to determine a potential route. |
| 0x06 | WRONG_LAST_HOP | This router is not the proper LHR. |
| 0x07 | NOT_FORWARDING | This router is not forwarding this source and group out the outgoing interface for an unspecified reason. |
| 0x08 | REACHED_RP | Reached the Rendezvous Point. |
| 0x09 | RPF_IF | Mtrace2 Request arrived on the expected RPF interface for this source and group. |
| 0x0A | NO_MULTICAST | Mtrace2 Request arrived on an interface which is not enabled for multicast. |
| 0x0B | INFO_HIDDEN | One or more hops have been hidden from this trace. |
| 0x0C | REACHED_GW | Mtrace2 Request arrived on a gateway (e.g., a NAT or firewall) that hides the information between this router and the Mtrace2 client. |
| 0x0D | UNKNOWN_QUERY | A non-transitive Extended Query Type was received by a router which does not support the type. |
| 0x80 | FATAL_ERROR | A fatal error is one where the router may know the upstream router but cannot forward the message to it. |
| 0x81 | NO_SPACE | There was not enough room to insert another Standard Response Block in the packet. |
| 0x83 | ADMIN_PROHIB | Mtrace2 is administratively prohibited. |

3.2.5. IPv6 Mtrace2 Standard Response Block

This section describes the message format of an IPv6 Mtrace2 Standard Response Block. The Type field is also 0x04.



MBZ: 8 bits

This field MUST be zeroed on transmission and ignored on reception.

Query Arrival Time: 32 bits

Same definition as in IPv4.

Incoming Interface ID: 32 bits

This field specifies the interface ID on which packets from the source or RP are expected to arrive, or 0 if unknown. This ID should be the value taken from InterfaceIndex of the IF-MIB [12] for this interface.

Outgoing Interface ID: 32 bits

This field specifies the interface ID to which packets from the source or RP are expected to transmit, or 0 if unknown. This ID should be the value taken from InterfaceIndex of the IF-MIB [12] for this interface

Local Address: 128 bits

This field specifies a global IPv6 address that uniquely identifies the router. A unique local unicast address [11] SHOULD NOT be used unless the router is only assigned link-local and unique local addresses. If the router is only assigned link-local addresses, its link-local address can be specified in this field.

Remote Address: 128 bits

This field specifies the address of the upstream router, which, in most cases, is a link-local unicast address for the upstream router.

Although a link-local address does not have enough information to identify a node, it is possible to detect the upstream router with the assistance of Incoming Interface ID and the current router address (i.e., Local Address).

Note that this may be a multicast group (e.g., ALL-[protocol]-ROUTERS group) if the upstream router is not known because of the workings of a multicast routing protocol. However, it should be the unspecified address (::) if the incoming interface address is unknown.

Input packet count on incoming interface: 64 bits

Same definition as in IPv4.

Output packet count on outgoing interface: 64 bits

Same definition as in IPv4.

Total number of packets for this source-group pair: 64 bits

Same definition as in IPv4, except if the S bit is set (see below), the count is for the source network, as specified by the Src Prefix Len field. If the S bit is set and the Src Prefix Len field is 255, indicating no source-specific state, the count is for all sources sending to this group. This counter should have

the same value as `ipMcastRoutePkts` from the IP Multicast MIB [13] for this forwarding entry.

Rtg Protocol: 16 bits
Same definition as in IPv4.

Multicast Rtg Protocol: 16 bits
Same definition as in IPv4.

MBZ 2: 15 bits
This field MUST be zeroed on transmission and ignored on reception.

S: 1 bit
Same definition as in IPv4, except the Src Prefix Len field is used to mask the source address.

Src Prefix Len: 8 bits
This field contains the prefix length this router has for the source. If the router is forwarding solely on group state, this field is set to 255 (0xff).

Forwarding Code: 8 bits
Same definition as in IPv4.

3.2.6. Mtrace2 Augmented Response Block

In addition to the Standard Response Block, a multicast router on the traced path can optionally add one or multiple Augmented Response Blocks before sending the Request to its upstream router.

The Augmented Response Block is flexible for various purposes such as providing diagnosis information (see Section 7) and protocol verification. Its Type field is 0x05, and its format is as follows:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------------------|---|---|---|---|---|---|---|---|---|------------|---|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|
| | | | | | | | | | | 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 | | | | | | | | |
| Type | | | | | | | | | | Length | | | | | | | | | | MBZ | | | | | | | | | | | | | | | | | | | |
| Augmented Response Type | | | | | | | | | | Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

MBZ: 8 bits
This field MUST be zeroed on transmission and ignored on reception.

Augmented Response Type: 16 bits

This field specifies the type of various responses from a multicast router that might need to communicate back to the Mtrace2 client as well as the multicast routers on the traced path.

The Augmented Response Type is defined as follows:

| Code | Type |
|------|--|
| ==== | ===== |
| 0x01 | # of the returned Standard Response Blocks |

When the NO_SPACE error occurs on a router, the router should send the original Mtrace2 Request received from the downstream router as a Reply back to the Mtrace2 client and continue with a new Mtrace2 Request. In the new Request, the router would add a Standard Response Block followed by an Augmented Response Block with 0x01 as the Augmented Response Type, and the number of the returned Mtrace2 Standard Response Blocks as the Value.

Each upstream router would recognize the total number of hops the Request has been traced so far by adding this number and the number of the Standard Response Block in the current Request message.

This document only defines one Augmented Response Type in the Augmented Response Block. The description on how to provide diagnosis information using the Augmented Response Block is out of the scope of this document, and will be addressed in separate documents.

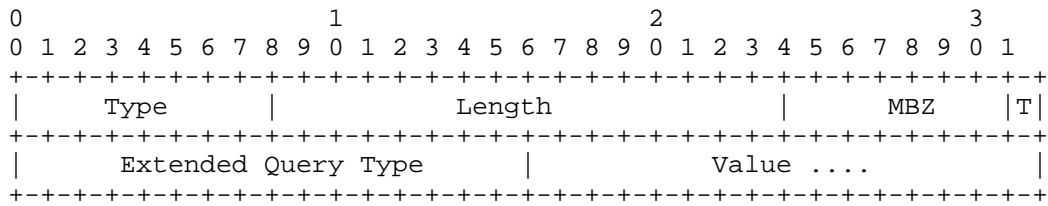
Value: variable length

The format is based on the Augmented Response Type value. The length of the value field is Length field minus 6.

3.2.7. Mtrace2 Extended Query Block

There may be a sequence of optional Extended Query Blocks that follow an Mtrace2 Query to further specify any information needed for the Query. For example, an Mtrace2 client might be interested in tracing the path the specified source and group would take based on a certain topology. In this case, the client can pass in the multi-topology ID as the Value for an Extended Query Type (see below). The Extended Query Type is extensible and the behavior of the new types will be addressed by separate documents.

The Mtrace2 Extended Query Block's Type field is 0x06, and is formatted as follows:



MBZ: 7 bits

This field MUST be zeroed on transmission and ignored on reception.

T-bit (Transitive Attribute): 1 bit

If the TLV type is unrecognized by the receiving router, then this TLV is either discarded or forwarded along with the Query, depending on the value of this bit. If this bit is set, then the router MUST forward this TLV. If this bit is clear, the router MUST send an Mtrace2 Reply with an UNKNOWN_QUERY error.

Extended Query Type: 16 bits

This field specifies the type of the Extended Query Block.

Value: 16 bits

This field specifies the value of this Extended Query.

4. Router Behavior

This section describes the router behavior in the context of Mtrace2 in detail.

4.1. Receiving Mtrace2 Query

An Mtrace2 Query message is an Mtrace2 message with no response blocks filled in, and uses TLV type of 0x01.

4.1.1. Query Packet Verification

Upon receiving an Mtrace2 Query message, a router MUST examine whether the Multicast Address and the Source Address are a valid combination as specified in Section 3.2.1, and whether the Mtrace2 Client Address is a valid IP unicast address. If either one is invalid, the Query MUST be silently ignored.

Mtrace2 supports a non-local client to the LHR/RP. A router SHOULD, however, support a mechanism to filter out queries from clients beyond a specified administrative boundary. The potential approaches are described in Section 9.2.

In the case where a local LHR client is required, the router must then examine the Query to see if it is the proper LHR/RP for the destination address in the packet. It is the proper local LHR if it has a multicast-capable interface on the same subnet as the Mtrace2 Client Address and is the router that would forward traffic from the given (S,G) or (*,G) onto that subnet. It is the proper RP if the multicast group address specified in the query is 0 and if the IP header destination address is a valid RP address on this router.

If the router determines that it is not the proper LHR/RP, or it cannot make that determination, it does one of two things depending on whether the Query was received via multicast or unicast. If the Query was received via multicast, then it MUST be silently discarded. If it was received via unicast, the router turns the Query into a Reply message by changing the TLV type to 0x03 and appending a Standard Response Block with a Forwarding Code of WRONG_LAST_HOP. The rest of the fields in the Standard Response Block MUST be zeroed. The router then sends the Reply message to the Mtrace2 Client Address on the Client Port # as specified in the Mtrace2 Query.

Duplicate Query messages as identified by the tuple (Mtrace2 Client Address, Query ID) SHOULD be ignored. This MAY be implemented using a cache of previously processed queries keyed by the Mtrace2 Client Address and Query ID pair. The duration of the cached entries is implementation specific. Duplicate Request messages MUST NOT be ignored in this manner.

4.1.2. Query Normal Processing

When a router receives an Mtrace2 Query and it determines that it is the proper LHR/RP, it turns the Query to a Request by changing the TLV type from 0x01 to 0x02, and performs the steps listed in Section 4.2.

4.2. Receiving Mtrace2 Request

An Mtrace2 Request is an Mtrace2 message that uses TLV type of 0x02. With the exception of the LHR, whose Request was just converted from a Query, each Request received by a router should have at least one Standard Response Block filled in.

4.2.1. Request Packet Verification

If the Mtrace2 Request does not come from an adjacent router, or if the Request is not addressed to this router, or if the Request is addressed to a multicast group which is not a link-scoped group (i.e., 224.0.0.0/24 for IPv4, FFx2::/16 [3] for IPv6), it MUST be silently ignored. The Generalized TTL Security Mechanism (GTSM) [14]

SHOULD be used by the router to determine whether the router is adjacent or not.

If the sum of the number of the Standard Response Blocks in the received Mtrace2 Request and the value of the Augmented Response Type of 0x01, if any, is equal or more than the # Hops in the Mtrace2 Request, it MUST be silently ignored.

4.2.2. Request Normal Processing

When a router receives an Mtrace2 Request message, it performs the following steps. Note that it is possible to have multiple situations covered by the Forwarding Codes. The first one encountered is the one that is reported, i.e. all "note Forwarding Code N" should be interpreted as "if Forwarding Code is not already set, set Forwarding Code to N". Note that in the steps described below the "Outgoing Interface" is the one on which the Mtrace2 Request message arrives.

1. Prepare a Standard Response Block to be appended to the packet, setting all fields to an initial default value of zero.
2. If Mtrace2 is administratively prohibited, note the Forwarding Code of ADMIN_PROHIB and skip to step 4.
3. In the Standard Response Block, fill in the Query Arrival Time, Outgoing Interface Address (for IPv4) or Outgoing Interface ID (for IPv6), Output Packet Count, and Fwd TTL (for IPv4).
4. Attempt to determine the forwarding information for the specified source and group, using the same mechanisms as would be used when a packet is received from the source destined for the group. A state need not be instantiated, it can be a "phantom" state created only for the purpose of the trace, such as "dry-run."

If using a shared-tree protocol and there is no source-specific state, or if no source-specific information is desired (i.e., all 1's for IPv4 or unspecified address (::) for IPv6), group state should be used. If there is no group state or no group-specific information is desired, potential source state (i.e., the path that would be followed for a source-specific Join) should be used.

5. If no forwarding information can be determined, the router notes a Forwarding Code of NO_ROUTE, sets the remaining fields that have not yet been filled in to zero, and then sends an Mtrace2 Reply back to the Mtrace2 client.

6. If a Forwarding Code of ADMIN_PROHIB has been set, skip to step 7. Otherwise, fill in the Incoming Interface Address (or Incoming Interface ID and Local Address for IPv6), Upstream Router Address (or Remote Address for IPv6), Input Packet Count, Total Number of Packets, Routing Protocol, S, and Src Mask (or Src Prefix Len for IPv6) using the forwarding information determined in step 4.
7. If the Outgoing interface is not enabled for multicast, note Forwarding Code of NO_MULTICAST. If the Outgoing interface is the interface from which the router would expect data to arrive from the source, note forwarding code RPF_IF. If the Outgoing interface is not one to which the router would forward data from the source or RP to the group, a Forwarding code of WRONG_IF is noted. In the above three cases, the router will return an Mtrace2 Reply and terminate the trace.
8. If the group is subject to administrative scoping on either the Outgoing or Incoming interfaces, a Forwarding Code of SCOPED is noted.
9. If this router is the RP for the group for a non-source-specific query, note a Forwarding Code of REACHED_RP. The router will send an Mtrace2 Reply and terminate the trace.
10. If this router is directly connected to the specified source or source network on the Incoming interface, it sets the Upstream Router Address (for IPv4) or the Remote Address (for IPv6) of the response block to zero. The router will send an Mtrace2 Reply and terminate the trace.
11. If this router has sent a prune upstream which applies to the source and group in the Mtrace2 Request, it notes a Forwarding Code of PRUNE_SENT. If the router has stopped forwarding downstream in response to a prune sent by the downstream router, it notes a Forwarding Code of PRUNE_RCVD. If the router should normally forward traffic downstream for this source and group but is not, it notes a Forwarding Code of NOT_FORWARDING.
12. If this router is a gateway (e.g., a NAT or firewall) that hides the information between this router and the Mtrace2 client, it notes a Forwarding Code of REACHED_GW. The router continues the processing as described in Section 4.5.
13. If the total number of the Standard Response Blocks, including the newly prepared one, and the value of the Augmented Response Type of 0x01, if any, is less than the # Hops in the Request, the packet is then forwarded to the upstream router as described

in Section 4.3; otherwise, the packet is sent as an Mtrace2 Reply to the Mtrace2 client as described in Section 4.4.

4.3. Forwarding Mtrace2 Request

This section describes how an Mtrace2 Request should be forwarded.

4.3.1. Destination Address

If the upstream router for the Mtrace2 Request is known for this request, the Mtrace2 Request is sent to that router. If the Incoming interface is known but the upstream router is not, the Mtrace2 Request is sent to an appropriate multicast address on the Incoming interface. The multicast address SHOULD depend on the multicast routing protocol in use, such as ALL-[protocol]-ROUTERS group. It MUST be a link-scoped group (i.e., 224.0.0.0/24 for IPv4, FF02::/16 for IPv6), and MUST NOT be the all-systems multicast group (224.0.0.1) for IPv4 and All Nodes Address (FF02::1) for IPv6. It MAY also be the all-routers multicast group (224.0.0.2) for IPv4 or All Routers Address (FF02::2) for IPv6 if the routing protocol in use does not define a more appropriate multicast address.

4.3.2. Source Address

An Mtrace2 Request should be sent with the address of the Incoming interface. However, if the Incoming interface is unnumbered, the router can use one of its numbered interface addresses as the source address.

4.3.3. Appending Standard Response Block

An Mtrace2 Request MUST be sent upstream towards the source or the RP after appending a Standard Response Block to the end of the received Mtrace2 Request. The Standard Response Block includes the multicast states and statistics information of the router described in Section 3.2.4.

If appending the Standard Response Block would make the Mtrace2 Request packet longer than the MTU of the Incoming Interface, or, in the case of IPv6, longer than 1280 bytes, the router MUST change the Forwarding Code in the last Standard Response Block of the received Mtrace2 Request into NO_SPACE. The router then turns the Request into a Reply and sends the Reply as described in Section 4.4.

The router will continue with a new Request by copying from the old Request excluding all the response blocks, followed by the previously prepared Standard Response Block, and an Augmented Response Block with Augmented Response Type of 0x01 and the number of the returned

Standard Response Blocks as the value. The new Request is then forwarded upstream.

4.4. Sending Mtrace2 Reply

An Mtrace2 Reply MUST be returned to the client by a router if any of the following conditions occur:

1. The total number of the traced routers are equal to the # of hops in the request (including the one just added) plus the number of the returned blocks, if any.
2. Appending the Standard Response Block would make the Mtrace2 Request packet longer than the MTU of the Incoming interface. (In case of IPv6 not more than 1280 bytes; see Section 4.3.3 for additional details on handling of this case.)
3. The request has reached the RP for a non source specific query or has reached the first hop router for a source specific query (see Section 4.2.2, items 9 and 10 for additional details).

4.4.1. Destination Address

An Mtrace2 Reply MUST be sent to the address specified in the Mtrace2 Client Address field in the Mtrace2 Request.

4.4.2. Source Address

An Mtrace2 Reply SHOULD be sent with the address of the router's Outgoing interface. However, if the Outgoing interface address is unnumbered, the router can use one of its numbered interface addresses as the source address.

4.4.3. Appending Standard Response Block

An Mtrace2 Reply MUST be sent with the prepared Standard Response Block appended at the end of the received Mtrace2 Request except in the case of NO_SPACE forwarding code.

4.5. Proxying Mtrace2 Query

When a gateway (e.g., a NAT or firewall), which needs to block unicast packets to the Mtrace2 client, or hide information between the gateway and the Mtrace2 client, receives an Mtrace2 Query from an adjacent host or Mtrace2 Request from an adjacent router, it appends a Standard Response Block with REACHED_GW as the Forwarding Code. It turns the Query or Request into a Reply, and sends the Reply back to the client.

At the same time, the gateway originates a new Mtrace2 Query message by copying the original Mtrace2 header (the Query or Request without any of the response blocks), and makes the changes as follows:

- o sets the RPF interface's address as the Mtrace2 Client Address;
- o uses its own port number as the Client Port #; and,
- o decreases # Hops by ((number of the Standard Response Blocks that were just returned in a Reply) - 1). The "-1" in this expression accounts for the additional Standard Response Block appended by the gateway router.

The new Mtrace2 Query message is then sent to the upstream router or to an appropriate multicast address on the RPF interface.

When the gateway receives an Mtrace2 Reply whose Query ID matches the one in the original Mtrace2 header, it MUST relay the Mtrace2 Reply back to the Mtrace2 client by replacing the Reply's header with the original Mtrace2 header. If the gateway does not receive the corresponding Mtrace2 Reply within the [Mtrace Reply Timeout] period (see Section 5.8.4), then it silently discards the original Mtrace2 Query or Request message, and terminates the trace.

4.6. Hiding Information

Information about a domain's topology and connectivity may be hidden from the Mtrace2 Requests. The Forwarding Code of INFO_HIDDEN may be used to note that. For example, the incoming interface address and packet count on the ingress router of a domain, and the outgoing interface address and packet count on the egress router of the domain can be specified as all 1's. Additionally, the source-group packet count (see Section 3.2.4 and Section 3.2.5) within the domain may be all 1's if it is hidden.

5. Client Behavior

This section describes the behavior of an Mtrace2 client in detail.

5.1. Sending Mtrace2 Query

An Mtrace2 client initiates an Mtrace2 Query by sending the Query to the LHR of interest.

5.1.1. Destination Address

If an Mtrace2 client knows the proper LHR, it unicasts an Mtrace2 Query packet to that router; otherwise, it MAY send the Mtrace2 Query packet to the all-routers multicast group (224.0.0.2) for IPv4 or All Routers Address (FF02::2) for IPv6. This will ensure that the packet is received by the LHR on the subnet.

See also Section 5.4 on determining the LHR.

5.1.2. Source Address

An Mtrace2 Query MUST be sent with the client's interface address, which would be the Mtrace2 Client Address.

5.2. Determining the Path

An Mtrace2 client could send an initial Query messages with a large # Hops, in order to try to trace the full path. If this attempt fails, one strategy is to perform a linear search (as the traditional unicast traceroute program does); set the # Hops field to 1 and try to get a Reply, then 2, and so on. If no Reply is received at a certain hop, the hop count can continue past the non-responding hop, in the hopes that further hops may respond. These attempts should continue until the [Mtrace Reply Timeout] timeout has occurred.

See also Section 5.6 on receiving the results of a trace.

5.3. Collecting Statistics

After a client has determined that it has traced the whole path or as much as it can expect to (see Section 5.8), it might collect statistics by waiting a short time and performing a second trace. If the path is the same in the two traces, statistics can be displayed as described in Section 7.3 and Section 7.4.

5.4. Last Hop Router (LHR)

The Mtrace2 client may not know which is the last-hop router, or that router may be behind a firewall that blocks unicast packets but passes multicast packets. In these cases, the Mtrace2 Request should be multicasted to the all-routers multicast group (224.0.0.2) for IPv4 or All Routers Address (FF02::2) for IPv6. All routers except the correct last-hop router SHOULD ignore any Mtrace2 Request received via multicast.

5.5. First Hop Router (FHR)

The IANA assigned 224.0.1.32 as the default multicast group for old IPv4 mtrace (v1) responses, in order to support mtrace clients that are not unicast reachable from the first-hop router. Mtrace2, however, does not require any IPv4/IPv6 multicast addresses for the Mtrace2 Replies. Every Mtrace2 Reply is sent to the unicast address specified in the Mtrace2 Client Address field of the Mtrace2 Reply.

5.6. Broken Intermediate Router

A broken intermediate router might simply not understand Mtrace2 packets, and drop them. The Mtrace2 client will get no Reply at all as a result. It should then perform a hop-by-hop search by setting the # Hops field until it gets an Mtrace2 Reply. The client may use linear or binary search; however, the latter is likely to be slower because a failure requires waiting for the [Mtrace Reply Timeout] period.

5.7. Non-Supported Router

When a non-supported router receives an Mtrace2 Query or Request message whose destination address is a multicast address, the router will silently discard the message.

When the router receives an Mtrace2 Query which is destined to itself, the router would return an Internet Control Message Protocol (ICMP) port unreachable to the Mtrace2 client. On the other hand, when the router receives an Mtrace2 Request which is destined to itself, the router would return an ICMP port unreachable to its adjacent router from which the Request receives. Therefore, the Mtrace2 client needs to terminate the trace when the [Mtrace Reply Timeout] timeout has occurred, and may then issue another Query with a lower number of # Hops.

5.8. Mtrace2 Termination

When performing an expanding hop-by-hop trace, it is necessary to determine when to stop expanding.

5.8.1. Arriving at Source

A trace can be determined to have arrived at the source if the Incoming Interface of the last router in the trace is non-zero, but the Upstream Router is zero.

5.8.2. Fatal Error

A trace has encountered a fatal error if the last Forwarding Error in the trace has the 0x80 bit set.

5.8.3. No Upstream Router

A trace cannot continue if the last Upstream Router in the trace is set to 0.

5.8.4. Reply Timeout

This document defines the [Mtrace Reply Timeout] value, which is used to time out an Mtrace2 Reply as seen in Section 4.5, Section 5.2, and Section 5.7. The default [Mtrace Reply Timeout] value is 10 (seconds), and can be manually changed on the Mtrace2 client and routers.

5.9. Continuing after an Error

When the NO_SPACE error occurs, as described in Section 4.2, a router will send back an Mtrace2 Reply to the Mtrace2 client, and continue with a new Request (see Section 4.3.3). In this case, the Mtrace2 client may receive multiple Mtrace2 Replies from different routers along the path. When this happens, the client MUST treat them as a single Mtrace2 Reply message.

If a trace times out, it is very likely that a router in the middle of the path does not support Mtrace2. That router's address will be in the Upstream Router field of the last Standard Response Block in the last received Reply. A client may be able to determine (via minfo or the Simple Network Management Protocol (SNMP) [11][13]) a list of neighbors of the non-responding router. The neighbors obtained in this way could then be probed (via the multicast MIB [13]) to determine which one is the upstream neighbor (i.e., Reverse Path Forwarding (RPF) neighbor) of the non-responding router. This algorithm can identify the upstream neighbor because, even though there may be multiple neighbors, the non-responding router should only have sent a "join" to the one neighbor corresponding to its selected RPF path. Because of this, only the RPF neighbor should contain the non-responding router as a multicast next hop in its MIB output list for the affected multicast route.

6. Protocol-Specific Considerations

This section describes the Mtrace2 behavior with the presence of different multicast protocols.

6.1. PIM-SM

When an Mtrace2 reaches a PIM-SM RP, and the RP does not forward the trace on, it means that the RP has not performed a source-specific join so there is no more state to trace. However, the path that traffic would use if the RP did perform a source-specific join can be traced by setting the trace destination to the RP, the trace source to the traffic source, and the trace group to 0. This Mtrace2 Query may be unicasted to the RP, and the RP takes the same actions as an LHR.

6.2. Bi-Directional PIM

Bi-directional PIM [6] is a variant of PIM-SM that builds bi-directional shared trees connecting multicast sources and receivers. Along the bi-directional shared trees, multicast data is natively forwarded from the sources to the Rendezvous Point Link (RPL), and from which, to receivers without requiring source-specific state. In contrast to PIM-SM, Bi-directional PIM always has the state to trace.

A Designated Forwarder (DF) for a given Rendezvous Point Address (RPA) is in charge of forwarding downstream traffic onto its link, and forwarding upstream traffic from its link towards the RPL that the RPA belongs to. Hence Mtrace2 Reply reports DF addresses or RPA along the path.

6.3. PIM-DM

Routers running PIM Dense Mode [15] do not know the path packets would take unless traffic is flowing. Without some extra protocol mechanism, this means that in an environment with multiple possible paths with branch points on shared media, Mtrace2 can only trace existing paths, not potential paths. When there are multiple possible paths but the branch points are not on shared media, the upstream router is known, but the LHR may not know that it is the appropriate last hop.

When traffic is flowing, PIM Dense Mode routers know whether or not they are the LHR for the link (because they won or lost an Assert battle) and know who the upstream router is (because it won an Assert battle). Therefore, Mtrace2 is always able to follow the proper path when traffic is flowing.

6.4. IGMP/MLD Proxy

When an IGMP or Multicast Listener Discovery (MLD) Proxy [7] receives an Mtrace2 Query packet on an incoming interface, it notes a WRONG_IF in the Forwarding Code of the last Standard Response Block (see

Section 3.2.4), and sends the Mtrace2 Reply back to the Mtrace2 client. On the other hand, when an Mtrace2 Query packet reaches an outgoing interface of the IGMP/MLD proxy, it is forwarded onto its incoming interface towards the upstream router.

7. Problem Diagnosis

This section describes different scenarios Mtrace2 can be used to diagnose the multicast problems.

7.1. Forwarding Inconsistencies

The Forwarding Error code can tell if a group is unexpectedly pruned or administratively scoped.

7.2. TTL or Hop Limit Problems

By taking the maximum of hops from the source and forwarding TTL threshold over all hops, it is possible to discover the TTL or hop limit required for the source to reach the destination.

7.3. Packet Loss

By taking multiple traces, it is possible to find packet loss information by tracking the difference between the output packet count for the specified source-group address pair at a given upstream router and the input packet count on the next hop downstream router. On a point-to-point link, any steadily increasing difference in these counts implies packet loss. Although the packet counts will differ due to Mtrace2 Request propagation delay, the difference should remain essentially constant (except for jitter caused by differences in propagation time among the trace iterations). However, this difference will display a steady increase if packet loss is occurring. On a shared link, the count of input packets can be larger than the number of output packets at the previous hop, due to other routers or hosts on the link injecting packets. This appears as "negative loss" which may mask real packet loss.

In addition to the counts of input and output packets for all multicast traffic on the interfaces, the Standard Response Block includes a count of the packets forwarded by a node for the specified source-group pair. Taking the difference in this count between two traces and then comparing those differences between two hops gives a measure of packet loss just for traffic from the specified source to the specified receiver via the specified group. This measure is not affected by shared links.

On a point-to-point link that is a multicast tunnel, packet loss is usually due to congestion in unicast routers along the path of that tunnel. On native multicast links, loss is more likely in the output queue of one hop, perhaps due to priority dropping, or in the input queue at the next hop. The counters in the Standard Response Block do not allow these cases to be distinguished. Differences in packet counts between the incoming and outgoing interfaces on one node cannot generally be used to measure queue overflow in the node.

7.4. Link Utilization

Again, with two traces, you can divide the difference in the input or output packet counts at some hop by the difference in time stamps from the same hop to obtain the packet rate over the link. If the average packet size is known, then the link utilization can also be estimated to see whether packet loss may be due to the rate limit or the physical capacity on a particular link being exceeded.

7.5. Time Delay

If the routers have synchronized clocks, it is possible to estimate propagation and queuing delay from the differences between the timestamps at successive hops. However, this delay includes control processing overhead, so is not necessarily indicative of the delay that data traffic would experience.

8. IANA Considerations

The following new registries are to be created and maintained under the "RFC Required" registry policy as specified in [4].

8.1. "Mtrace2 Forwarding Codes" Registry

This is an integer in the range 0-255. Assignment of a Forwarding Code requires specification of a value and a name for the Forwarding Code. Initial values for the forwarding codes are given in the table at the end of Section 3.2.4. Additional values (specific to IPv6) may also be specified at the end of Section 3.2.5. Any additions to this registry are required to fully describe the conditions under which the new Forwarding Code is used.

8.2. "Mtrace2 TLV Types" registry

Assignment of a TLV Type requires specification of an integer value "Code" in the range 0-255 and a name ("Type"). Initial values for the TLV Types are given in the table at the beginning of Section 3.2.

8.3. UDP Destination Port

IANA has assigned UDP user port 33435 (mtrace) for use by this protocol as the Mtrace2 UDP destination port.

9. Security Considerations

This section addresses some of the security considerations related to Mtrace2.

9.1. Addresses in Mtrace2 Header

An Mtrace2 header includes three addresses, source address, multicast address, and Mtrace2 client address. These addresses **MUST** be congruent with the definition defined in Section 3.2.1 and forwarding Mtrace2 messages having invalid addresses **MUST** be prohibited. For instance, if Mtrace2 Client Address specified in an Mtrace2 header is a multicast address, then a router that receives the Mtrace2 message **MUST** silently discard it.

9.2. Filtering of Clients

A router **SHOULD** support a mechanism to filter out queries from clients beyond a specified administrative boundary. Such a boundary could, for example, be specified via a list of allowed/disallowed client addresses or subnets. If a query is received from beyond the specified administrative boundary, the Query **MUST NOT** be processed. The router **MAY**, however, perform rate limited logging of such events.

9.3. Topology Discovery

Mtrace2 can be used to discover any actively-used topology. If your network topology is a secret, Mtrace2 may be restricted at the border of your domain, using the ADMIN_PROHIB forwarding code.

9.4. Characteristics of Multicast Channel

Mtrace2 can be used to discover what sources are sending to what groups and at what rates. If this information is a secret, Mtrace2 may be restricted at the border of your domain, using the ADMIN_PROHIB forwarding code.

9.5. Limiting Query/Request Rates

A router may limit Mtrace2 Queries and Requests by ignoring some of the consecutive messages. The router **MAY** randomly ignore the received messages to minimize the processing overhead, i.e., to keep

fairness in processing queries, or prevent traffic amplification. The rate limit is left to the router's implementation.

9.6. Limiting Reply Rates

The proxying and NO_SPACE behaviors may result in one Query returning multiple Reply messages. In order to prevent abuse, the routers in the traced path MAY need to rate-limit the Replies. The rate limit function is left to the router's implementation.

10. Acknowledgements

This specification started largely as a transcription of Van Jacobson's slides from the 30th IETF, and the implementation in mtrouted 3.3 by Ajit Thyagarajan. Van's original slides credit Steve Casner, Steve Deering, Dino Farinacci and Deb Agrawal. The original multicast traceroute client, mtrace (version 1), has been implemented by Ajit Thyagarajan, Steve Casner and Bill Fenner. The idea of the "S" bit to allow statistics for a source subnet is due to Tom Pusateri.

For the Mtrace version 2 specification, the authors would like to give special thanks to Tatsuya Jinmei, Bill Fenner, and Steve Casner. Also, extensive comments were received from David L. Black, Ronald Bonica, Yiqun Cai, Liu Hui, Bharat Joshi, Robert Kebler, John Kristoff, Mankamana Mishra, Heidi Ou, Pekka Savola, Shinsuke Suzuki, Dave Thaler, Achmad Husni Thamrin, Stig Venaas, and Cao Wei.

11. References

11.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.
- [2] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 8200, July 2017.
- [3] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [4] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 8126, June 2017.

- [5] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 7761, March 2016.
- [6] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, October 2007.
- [7] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.

11.2. Informative References

- [8] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [9] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, February 2015.
- [10] Rosen, E. and R. Aggarwal, "Multicast in MPLS/BGP IP VPNs", RFC 6513, February 2012.
- [11] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [12] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [13] McWalter, D., Thaler, D., and A. Kessler, "IP Multicast MIB", RFC 5132, December 2007.
- [14] Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, October 2007.
- [15] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, January 2005.

Authors' Addresses

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

Kerry Meyer

Email: kerry.meyer@me.com

WeeSan Lee (editor)

Email: weesan@weesan.com

Mboned
Internet-Draft
Intended status: Experimental
Expires: December 31, 2018

J. Holland
K. Rose
Akamai Technologies, Inc.
June 29, 2018

Asymmetric Manifest Based Integrity
draft-jholland-mboned-ambi-00

Abstract

This document introduces Asymmetric Manifest-Based Integrity (AMBI). AMBI allows each receiver of a stream of multicast packets to check the integrity of the contents of each packet in the data stream. AMBI operates by passing cryptographically verifiable manifests for the data packets, over out-of-band communication channels.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 31, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 1.1. Comparison with TESLA | 4 |
| 1.2. Terminology | 4 |
| 2. Protocol Specification | 4 |
| 2.1. Packet Identifiers | 4 |
| 2.1.1. Overview | 4 |
| 2.1.2. RTP Sequence Number | 5 |
| 2.1.3. SRTP Sequence Number | 5 |
| 2.1.4. UDP Option | 5 |
| 2.2. Anchor Message | 6 |
| 2.2.1. Overview | 6 |
| 2.2.2. DNS-based Anchor URI Bootstrap | 6 |
| 2.2.3. Anchor Message YANG model | 6 |
| 2.2.4. Example Anchor Message | 13 |
| 2.3. Manifests | 15 |
| 2.3.1. Overview | 15 |
| 2.3.2. Manifest Layout | 15 |
| 3. IANA Considerations | 16 |
| 4. Security Considerations | 17 |
| 4.1. Packet Identifiers | 17 |
| 5. References | 17 |
| 5.1. Normative References | 17 |
| 5.2. Informative References | 17 |
| Authors' Addresses | 18 |

1. Introduction

Multicast transport poses security problems that are not easily addressed by the same security mechanisms used for unicast transport.

The "Introduction" sections of the documents describing TESLA [RFC4082], and TESLA in SRTP [RFC4383], and TESLA with ALC and NORM [RFC5776] present excellent overviews of the challenges unique to multicast authentication, briefly summarized here:

- o A MAC based on a symmetric shared secret cannot be used because each packet has multiple receivers that do not trust each other.
- o Asymmetric per-packet signatures can handle only very low bit-rates because of the computational overhead.
- o An asymmetric signature of a larger message comprising multiple packets requires reliable receipt of all such packets, something that cannot be guaranteed in a timely manner even for protocols that do provide reliable delivery, and the retransmission of which

may anyway exceed the useful lifetime for data formats that can otherwise tolerate some degree of loss.

Asymmetric Manifest-Based Integrity (AMBI) specifies a method for receivers or middle boxes to cryptographically authenticate and verify the integrity of a stream of packets, by communicating packet "manifests" (described in Section 2.3) via an out-of-band communication channel that provides authentication and verifiable integrity.

Each manifest contains cryptographic hashes of packet payloads corresponding to specific packets in the authenticated data stream.

Three ways to authenticate a manifest are defined:

- o Asymmetric signature of a message containing the manifest.
- o Authenticated unicast stream providing a sequence of manifests.
- o Using one of the prior two constructions to bootstrap a root manifest containing authentication information for further manifests. This we term "recursive authentication".

When using asymmetric signatures, recursive authentication allows the sender to amortize the computational overhead for a single asymmetric signature across enough data packets to sustain high data rates. When using a secure unicast stream, the recursive verification allows for scaling the authenticated data stream to more receivers than can otherwise be sustained with a limited set of trusted servers.

Upon successful verification of the contents of a manifest and receipt of any subset of the corresponding data packets, the receiver has proof of the integrity of the contents of the data packets listed in the manifest.

An "anchor message", described in Section 2.2, provides the link between an authenticated data stream and the out-of-band channel of manifests that authenticates it.

Authenticating the integrity of the data packets depends on:

- o authentication of the anchor message that provides the linkage between the manifest channel and the data stream; and
- o the secrecy and cryptographic strength of private keys used for signing manifests, or the authentication of the secure unicast streams used for transmitting manifests; and

- o the difficulty of generating a collision for the packet hashes in the manifest.

1.1. Comparison with TESLA

AMBI and TESLA [RFC4082] and [RFC5776] attempt to achieve a similar goal of authenticating the integrity of streams of multicast packets. AMBI imposes a higher overhead, as measured in the amount of extra data required, than TESLA imposes. In exchange, AMBI provides non-repudiation (which TESLA does not), and relaxes the requirement for establishing an upper bound on clock synchronization between sender and receiver.

This tradeoff enables new capabilities for AMBI, relative to TESLA. In particular, when receiving multicast traffic from an untrusted transit network, AMBI can be used by a middle box to authenticate packets from a trusted source before forwarding traffic through the network, and the receiver also can separately authenticate the packets. (This use case is not possible with TESLA because the data packets can't be authenticated until a key is disclosed, so either the middlebox has to forward data packets without first authenticating so that the receiver has them prior to key disclosure, or the middlebox has to hold packets until the key is disclosed, at which point the receiver can no longer establish their authenticity.)

1.2. Terminology

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].

2. Protocol Specification

2.1. Packet Identifiers

2.1.1. Overview

Packet identifiers are a sequence number contained within the authenticated payload of the packet. This sequence number is used inside a manifest to associate each packet hash with a specific packet. Each authenticated packet MUST contain a packet identifier. See Section 4.1 for a discussion of the security implications.

This document defines a new UDP option in Section 2.1.4 for use as a packet identifier.

Some multicast-capable transport protocols have a sequence number embedded in data packets in the protocol. The sequence numbers in

these protocols MAY be used instead of the new UDP option, to avoid introducing extra overhead in the authenticated data packets.

In Section 2.1.2, Section 2.1.3, and Section 2.1.4, this document defines some sample ways to specify packet identifiers based on such sequence numbers embedded in existing protocols.

Other appropriate sequence number systems may exist, such as the anti-replay Sequence Number field in Section 3.1 of [RFC6584], when NORM or FLUTE operates with an authentication profile that uses it (however, since that example already provides authentication, it is not added as an option in this document). The AMBI anchor message format can be extended in future documents to support those or other suitable schemes by adding values to the registry defined in Section 3.

In some deployments, in contrast to using the new UDP option, the approach of using an existing sequence number may carry a benefit because it requires no change to the stream of packets being authenticated, possibly enabling interoperability with legacy receivers.

2.1.2. RTP Sequence Number

Sequence number from Section 5.1 of [RFC3550].

TBD: discussion of security consequences of using 16 bits- recommend a bigger hash in manifests for this case?

2.1.3. SRTP Sequence Number

Packet Index from Section 3.3.1 of [RFC3711].

2.1.4. UDP Option

Define a new UDP option [I-D.ietf-tsvwg-udp-options] (TBD2).

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
| UDP Kind=TBD2 | Length=6 | 32-bit sequence number |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

2.2. Anchor Message

2.2.1. Overview

An anchor message provides the information that makes it possible to associate the manifests with the data packets they authenticate. ID values that appear as text integers in the anchor message also appear in the manifest binary data, with the anchor message providing context on how to interpret the values.

An anchor message MAY be discovered and transmitted by any means which provides adequate source authentication and data integrity to meet the security needs of the receiver.

In order to support middle-box authentication, it is RECOMMENDED that senders arrange to distribute anchor messages according to the method outlined in Section 2.2.2.

2.2.2. DNS-based Anchor URI Bootstrap

When a middle box tries to process a join for a specific source, if it is configured to perform authentication on SSM multicast channels it can forward, it SHOULD make a DNS request for `ambi.(reverse-source-ip).ip6.arpa` or `ambi.(reverse-source-ip).in-addr.arpa`, for IPv6 or IPv4 source addresses.

When AMBI is provided to authenticate traffic from this source IP, this domain name SHOULD be configured with a TXT field which contains a URI that can be used to securely fetch an anchor message that describes all the AMBI- authenticatable traffic from this source IP.

Other methods MAY be used to discover and transfer an anchor message. The description of alternate methods is out of scope for this document.

TBD: consider breaking up anchor message to avoid large, frequently changing anchors for sources with many groups.

TBD: consider graceful rollover for anchors, instead of synchronized update of anchor hash.

2.2.3. Anchor Message YANG model

```
<CODE BEGINS> file "ietf-ambi-anchor.yang"
module ietf-ambi-anchor {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-ambi-anchor";
```



```
prefix "ambi";

import ietf-yang-types {
    prefix "yang";
    reference "RFC6991 Section 3";
}

import ietf-inet-types {
    prefix "inet";
    reference "RFC6991 Section 4";
}

import ietf-routing-types {
    prefix "rt-types";
    reference "RFC8294";
}

organization "IETF";

contact
    "Author:    Jake Holland
               <mailto:jholland@akamai.com>
    Author:    Kyle Rose
               <mailto:krose@akamai.com>
    ";

description
    "This module contains the definition for the AMBI anchor
    message data type.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of
    draft-jholland-mboned-ambi,
    see the internet draft itself for full legal notices.";

revision 2018-06-27 {
    description "Initial revision.";
    reference
        "";
```

```
}

/* TBD: copied some from https://tools.ietf.org/html/rfc8177,
   but the model doesn't seem to match what I want. is there another
   I can import instead of making these here? Or a registry
   to reference? */
identity crypto-hash {
    description
        "Base identity of cryptographic hash options. ";
}

identity sha-256 {
    base crypto-hash;
    description
        "The SHA-256 algorithm.";
}

identity blake2b {
    base crypto-hash;
    description
        "The BLAKE2b algorithm.";
}

identity crypto-signature {
    description
        "Base identity of cryptographic asymmetric signature
        options.";
}

identity ed25519 {
    base crypto-signature;
    description
        "The Ed25519 algorithm.";
}

identity rsa {
    base crypto-signature;
    description
        "The RSA algorithm.";
}

identity sequence-type {
    description
        "Base identity for sequence number type options.";
}

identity rtp {
    base sequence-type;
```

```
        description
            "The sequence number from RTP.";
    }

    identity srtp {
        base sequence-type;
        description
            "The sequence number from SRTP.";
    }

    identity udp {
        base sequence-type;
        description
            "The sequence number from UDP.";
    }

    typedef key-identifier {
        type uint16 {
            range 1..65535;
        }
        description "Key identifier within a manifest";
    }

    typedef bitrate {
        type string {
            pattern '[1-9][0-9]*[GMK]?bps';
        }
        description "Bit-rate of a data stream";
    }

    typedef packetrate {
        type string {
            pattern '[1-9][0-9]*[GMK]?pps';
        }
        description "Packet rate of a data stream";
    }

    typedef manifest-transport {
        type union {
            type leafref {
                path "/anchor/data_stream/id";
            }
            type inet:uri;
        }
        description "Transport method for a manifest stream";
    }

    container anchor {
        container self {
```

```
presence "An anchor message exists";
description
  "Self-referential properties about the anchor message";
leaf uri {
  type inet:uri;
  mandatory true;
  description
    "The canonical URI for this anchor message.";
}
leaf version {
  type uint16;
  mandatory true;
  description
    "The version number for this anchor message.";
}
leaf hash_algorithm {
  type identityref {
    base crypto-hash;
  }
  mandatory true;
  description
    "The algorithm for the anchor message hash provided
    in a manifest.";
}
leaf hash_bits {
  type uint16;
  mandatory true;
  description
    "The number of bits for the anchor's hash provided
    in a manifest.";
}
leaf expires {
  type yang:date-and-time;
  mandatory true;
  description
    "The expiration time for this anchor message.";
}
}
description "Anchor message for AMBI";

list public_key {
  key id;
  description "Public key for non-recursive manifest";
  leaf id {
    type key-identifier;
    mandatory true;
    description
      "The key identifier referenced in a manifest.";
  }
}
```

```
    }
    leaf algorithm {
      type identityref {
        base crypto-signature;
      }
      mandatory true;
      description
        "The signature algorithm for use with this key.";
    }
    leaf signature_bits {
      type uint16;
      mandatory true;
      description
        "The length of the signature provided in manifests
        signed with this key.";
    }
    leaf value {
      type string;
      mandatory true;
      description
        "The base64-encoded value of the public key.";
    }
  }
}

list data_stream {
  key id;
  unique "source destination port";
  description "Stream of data packets to be authenticated";
  leaf id {
    type uint16;
    mandatory true;
    description
      "The datastream_id referenced by a
      manifest_stream.";
  }
  leaf source {
    type inet:ip-address;
    mandatory true;
    description
      "The source IP address of the authenticated data
      stream.";
  }
  leaf destination {
    type rt-types:ip-multicast-group-address;
    mandatory true;
    description
      "The destination group IP address of the
      authenticated data stream.";
  }
}
```

```
    }
    leaf port {
        type uint16;
        mandatory true;
        description
            "The destination UDP port of the authenticated data
            stream.";
    }
    leaf max_bitrate {
        type bitrate;
        mandatory true;
        description
            "The maximum bitrate expected for this data
            stream.";
    }
    leaf max_packetrate {
        type packetrate;
        mandatory true;
        description
            "The maximum packetrate expected for this data
            stream.";
    }
    list authenticator {
        key manifest_id;
        description
            "A manifest stream that authenticates this data";
        leaf manifest_id {
            type leafref {
                path "/anchor/manifest_stream/id";
            }
            mandatory true;
            description
                "The ID of a manifest stream that provides
                authentication for this data stream.";
        }
    }
}

list manifest_stream {
    key id;
    description "Stream of manifests";
    leaf id {
        type uint16;
        mandatory true;
        description "The Manifest ID referenced in a manifest.";
    }
    leaf transport {
        type manifest-transport;
    }
}
```

```

        mandatory true;
        description
            "The ID of the data stream that carries this
             manifest stream or a uri that provides a websocket
             with the stream of manifests.";
    }
    leaf hash_algorithm {
        type identityref {
            base crypto-hash;
        }
        mandatory true;
        description
            "The hash algorithm for the packet hashes within
             manifests in this stream.";
    }
    leaf hash_bits {
        type uint16;
        mandatory true;
        description
            "The number of bits of hash provided for packet
             hashes.";
    }
    leaf sequence_type {
        type identityref {
            base sequence-type;
        }
        mandatory true;
        description
            "The linkage to the data packet sequence numbers in
             the manifest.";
    }
}

}

}
<CODE ENDS>
```

Figure 1: Anchor Message YANG model

2.2.4. Example Anchor Message

```
{
  "ietf-ambi-anchor:anchor": {
    "self": {
      "uri": "https://example.com/ambi/anchor/example_1.json",
      "version": 1,
      "hash_algorithm": "blake2b",
      "hash_bits": 256,
      "expires": "2018-03-05T23:59:59Z"
    }
  }
}
```

```
    },
    "public_key": [
      {
        "id": 1,
        "algorithm": "ed25519",
        "signature_bits": 256,
        "value": "VGhpncyBpcyBub3QgYSBnb29kIGtleSB0byB1c2UuLi4NCg=="
      }
    ],
    "data_stream": [
      {
        "id": 10,
        "source": "192.0.2.10",
        "destination": "232.10.10.1",
        "port": 18001,
        "max_bitrate": "10Mbps",
        "max_packetrate": "1Kpps",
        "authenticator": [
          {
            "manifest_id": 1
          }
        ]
      },
      {
        "id": 20,
        "source": "192.0.2.10",
        "destination": "232.10.10.1",
        "port": 18002,
        "max_bitrate": "400Kbps",
        "max_packetrate": "40pps",
        "authenticator": [
          {
            "manifest_id": 2
          }
        ]
      }
    ],
    "manifest_stream": [
      {
        "id": 1,
        "transport": 20,
        "hash_algorithm": "blake2b",
        "hash_bits": 80,
        "sequence_type": "udp"
      },
      {
        "id": 2,
        "transport": "https://example.com/ambi/manifest_stream/3",
      }
    ]
  }
}
```



```
        "hash_algorithm": "blake2b",  
        "hash_bits": 80,  
        "sequence_type": "udp"  
      }  
    ]  
  }  
}
```

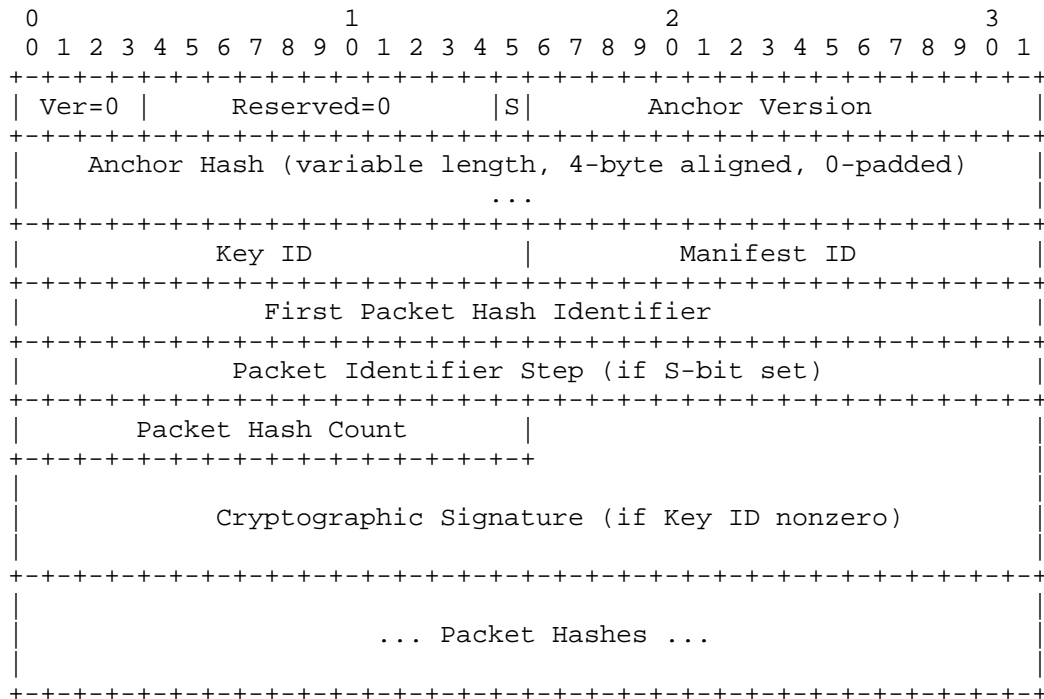
Figure 2: Example Anchor Message

2.3. Manifests

2.3.1. Overview

A manifest cannot be interpreted except in context of a known anchor message. In order for a manifest to be considered as potentially authenticating a set of packets, the anchor version **MUST** match the value in a known unexpired anchor message, and the hash **MUST** match the hash of the contents of that anchor message, according to the `/anchor/self/hash_algorithm` and `/anchor/self/hash_bits` fields, in order for a manifest to be accepted for use as evidence of authenticity and integrity.

2.3.2. Manifest Layout



3. IANA Considerations

TBD1: Request a "Specification Required" registry for Packet identifier methods, from <https://tools.ietf.org/html/rfc5226#section-4.1> . Reserve anything beginning with "experimental:"?

TBD2: Add a new entry to the "UDP Option Kind" numbers registry: <https://tools.ietf.org/html/draft-ietf-tsvwg-udp-options-02#section-14>

TBD: check guidelines in <https://tools.ietf.org/html/rfc5226> and remove this paragraph

Example from: <https://tools.ietf.org/html/rfc5226#section-5.1>

[TO BE REMOVED: Please add the yang model in Section 2.2.3 to: <https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml>

Name:ietf-ambi-anchor Maintained by IANA: N Namespace:
urn:ietf:params:xml:ns:yang:ietf-ambi-anchor Prefix: anchor
Reference: I-D.draft-jholland-mboned-ambi Notes:]

4. Security Considerations

4.1. Packet Identifiers

TBD: explain attack from generating malicious packets and then looking for collisions, as opposed to having to generate a collision including a sequence number and then hitting a match

TBD: DNSSEC vis-a-vis anchor url discovery. (we need a diagram about for middle-box handling of a revers-path propagated join?) Explain why malicious DNS could deny service, but cannot cause accepting attack packets.

TBD: follow the rest of the guidelines: <https://tools.ietf.org/html/rfc3552>

5. References

5.1. Normative References

- [I-D.ietf-tsvwg-udp-options] Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-02 (work in progress), January 2018.
- [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>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.

5.2. Informative References

- [DIGSIGN] Rohatgi, P., "How to Sign Digital Streams", 1997, <<https://link.springer.com/content/pdf/10.1007/BFb0052235.pdf>>.
- Published in CRYPTO '97 "Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology", Pages 180-197

- [RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, DOI 10.17487/RFC4082, June 2005, <<https://www.rfc-editor.org/info/rfc4082>>.
- [RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <<https://www.rfc-editor.org/info/rfc4383>>.
- [RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, DOI 10.17487/RFC5776, April 2010, <<https://www.rfc-editor.org/info/rfc5776>>.
- [RFC6584] Roca, V., "Simple Authentication Schemes for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 6584, DOI 10.17487/RFC6584, April 2012, <<https://www.rfc-editor.org/info/rfc6584>>.

Authors' Addresses

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com

Kyle Rose
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: krose@krose.org

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

J. Xie
Huawei Technologies
X. Xu
Alibaba Inc.
G. Yan
M. McBride
Huawei Technologies
July 2, 2018

Use of BIER Entropy for Data Center CLOS Networks
draft-xie-mboned-bier-entropy-staged-dc-clos-00

Abstract

Bit Index Explicit Replication (BIER) introduces a new multicast-specific BIER Header. BIER can be applied to the Multi Protocol Label Switching (MPLS) data plane or Non-MPLS data plane. Entropy is a technique used in BIER to support load-balancing. This document examines and describes how BIER Entropy is to be applied to Data Center CLOS networks for path selection.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

| | |
|--|---|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. Problem Statement and Considerations | 3 |
| 3.1. Problem Statement | 3 |
| 3.2. Considerations | 4 |
| 4. Use of BIER Entropy for DC CLOS Network | 5 |
| 4.1. Use of BIER Entropy for DC CLOS Network | 5 |
| 4.2. Steering for elephant flows | 6 |
| 4.3. Path Division for Tenant flows to different SIs | 6 |
| 4.4. Link Failure and Convergence | 6 |
| 5. Data-Plane Processing | 7 |
| 6. Security Considerations | 7 |
| 7. IANA Considerations | 7 |
| 8. Acknowledgements | 7 |
| 9. References | 7 |
| 9.1. Normative References | 7 |
| 9.2. Informative References | 8 |
| Authors' Addresses | 8 |

1. Introduction

Bit Index Explicit Replication (BIER) [RFC8279] is an architecture that provides optimal multicast forwarding without requiring intermediate routers to maintain any per-flow state by using a multicast-specific BIER header. [RFC8296] defines two types of BIER encapsulation formats: one is MPLS encapsulation, the other is non-MPLS encapsulation. Entropy is a technique used in BIER to support load-balancing. This document examines and describes how BIER Entropy is to be applied to Data Center CLOS networks for path selection.

2. Terminology

Readers of this document are assumed to be familiar with the terminology and concepts of the documents listed as Normative References.

3. Problem Statement and Considerations

3.1. Problem Statement

A common choice for a horizontally scalable topology used in Data Center is a Clos topology. This topology features an odd number of stages, for example, a 5-Stage Clos Topology as an example in [RFC7938].

ECMP is the fundamental load-sharing mechanism used by a Clos topology. Effectively, every lower-tier device will use all of its directly attached upper-tier devices to load-share traffic destined to the same IP prefix. The number of ECMP paths between any two Tier 3 devices in Clos topology is equal to the number of the devices in the middle stage (Tier 1). For example, Figure 1 illustrates a topology where Tier 3 device L1 has four paths to reach servers X and Y, via Tier 2 devices S1 and S2 and then Tier 1 devices S11, S12, S21, and S22, respectively.

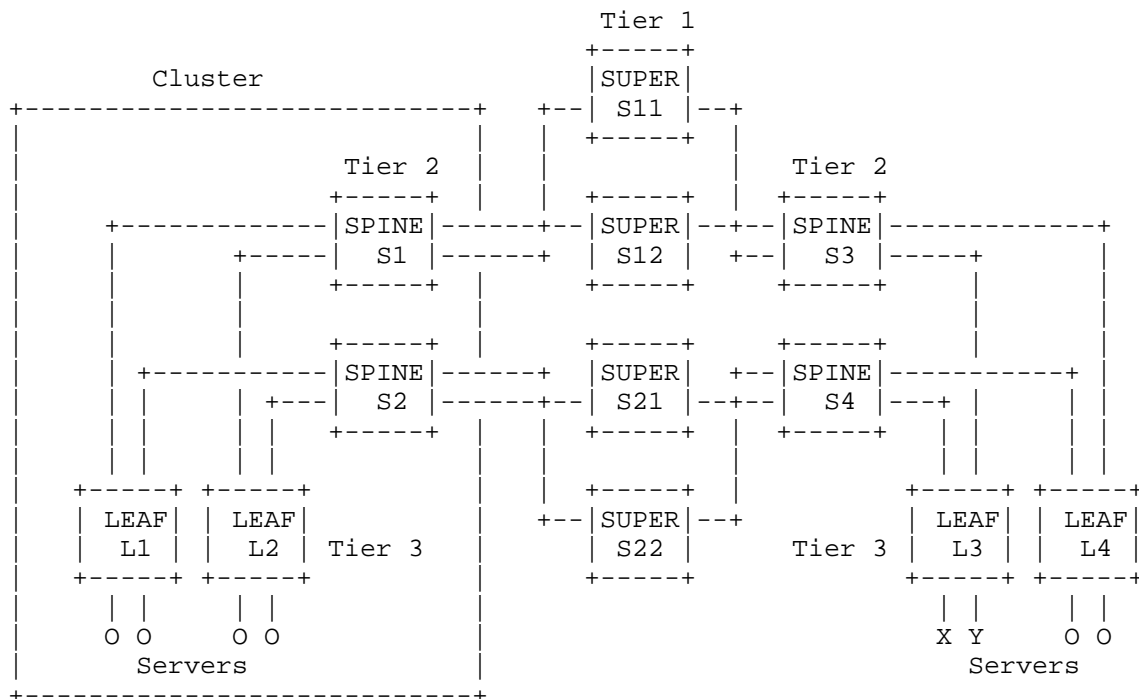


Figure 1: 5-Stage Clos Topology

When BIER is deployed in a multi-tenant data center network environment for efficient delivery of Broadcast, Unknown-unicast and Multicast (BUM) traffic, a network operator may want a deterministic path for every packet. For example, when L1 needs to send a BUM packet to L3 and L4, which are in different SIs, L1 has to send the packet twice, and expects the packet along two deterministic paths of L1->S1->S11->L3 and L1->S2->S21->L4 separately. Another example of using a deterministic path in a DC is for per-flow steering of "elephant" flows defined in [I-D.ietf-spring-segment-routing-msdc].

A deterministic path for a multicast path, with multiple staged equal cost paths, is comparable to a traffic-engineering path defined in [I-D.ietf-mpis-spring-entropy-label] for a unicast path with multiple hop equal cost paths.

3.2. Considerations

The idea behind entropy is that the ingress router computes a hash based on several fields from a given packet and places the result in an additional label, named "entropy label". Then this entropy label can be used as part of the hash keys used by a transit router. When

entropy label is used, the keys used in the hashing functions are still a local configuration matter. A router may solely use the entropy label or use a combination of multiple fields from the incoming packet. The hashing function is to randomly load balance the mass of flows between the small number of equal cost paths.

If one wants, however, to get a deterministic path from the equal cost paths, one can use part of the 20-bit entropy field. For example, bit 0 to bit 2 of entropy label can represent a value of 0 to 7, and thus can be used to select a deterministic path from 8 equal cost paths. And thus, a 20-bit entropy label can be used by routers in different tiers to select a deterministic path independently by using different parts of the 20-bit entropy label, and form an end-to-end deterministic path.

This is simple and applicable especially for DC CLOS networks, because data delivery in DC CLOS networks for tenants is always multi-staged, with the upstream direction stages having equal cost paths.

4. Use of BIER Entropy for DC CLOS Network

4.1. Use of BIER Entropy for DC CLOS Network

Take the 5-stage CLOS network in figure 1 as an example.

Tier 2 in every cluster has N nodes, and the Tier 1 has M nodes. M is equal to N multiplied by P.

Tier 3 switches, in upstream direction, act as stage 1 of data delivery and have N equal cost paths to every BFERs in other clusters. Tier 2 switches, in upstream direction, act as stage 2 of data delivery and have P equal cost paths to every BFERs in other clusters.

Example 1: One can configure, on each Tier 3 switch, the use of bit 0 for path selection when N is equal to 2, and configure, on each Tier 2 switch, to use bit 1 for path selection when P is equal to 2.

Example 2: One can configure, on each Tier 3 switch, the use of bit 0 to bit 1 for path selection when N is equal to 4, and configure on each Tier 2 switches the use of bit 2 to bit 7 for path selection when P is equal to 48.

Assume that, each Tier 3 and Tier 2 switch the the example have two parameters, X and Y, for using part of entropy label to do path selection, then in example 2:

- o Each of Tier 3 (Stage 1) switches has a pair of parameters ($X1=1$, $Y1=4$)
- o Each of Tier 2 (Stage 2) switches has a pair of parameters ($X2=X1*Y1=4$, $Y2=64$)
- o Each of Tier 3 (Stage 1) switches populates its BIFTs for ECMP, for example, BIFT-0 to BIFT-3.
- o Each of Tier 2 (Stage 2) switches populates its BIFTs for ECMP, for example, BIFT-0 to BIFT-47.

For each of Tier 3 (Stage 1) switches, each of the BIFT will have a preferred neighboring BFR. For example, LEAF L1 will have a preferred neighbor S1/S2 for BIFT-0/1 separately, and when forming the BIFT-0 table through the underlay routing to every BFER, the preferred neighboring BFR will have a highest priority among all the locally available ECMP path.

Then an end-to-end deterministic path for a BIER packet can be had by calculating an entropy label value like this:

$$\text{Entropy} = (P1-1)*X1 + (P2-1)*X2$$

Where P1 represents one of the Stage 1 equal cost paths with a value between 1 and N, and P2 represents one of the Stage 2 equal cost paths with a value between 1 and P.

4.2. Steering for elephant flows

One can steer an "elephant" flow to an end-to-end deterministic path, or some divided end-to-end deterministic paths across different SIs.

4.3. Path Division for Tenant flows to different SIs

When the VNEs for a tenant span multiple SIs, then it is useful to divide the BUM packets paths across different SIs.

One can configure a policy to use different paths for BIER SIs when using BIER as the BUM tunnel, on each VNE for each VNI.

4.4. Link Failure and Convergence

As stated above, each of the BIFT on a BFR will have a preferred neighboring BFR. But when the link to the preferred neighbor of some BIFT (say BIFT-X) fail, BIFT-X will converge normally, and will then probably not being the 'best' path. For example, the link between S1 and L2 fail, then the preferred neighbor of BIFT-0 of LEAF L1, S1, is

no longer the neighboring BFR for LEAF L2, and the flow using a Entropy using LEAF L1's BIFT-0 will have to replicate on L1, one packet to S1 for BFER L3 and L4, and one packet to S2 for BFER L2. If the flow changes to use a Entropy using LEAF L1's BIFT-1, it will then be the 'best' path, because the flow doesn't have to replicate on L1, only one to S1 for BFER L2 and L3 and L4. Such a change to a flow's entropy is the Ingress switch's responsibility, possibly with the assistance of a controller.

5. Data-Plane Processing

The use of BIER entropy label to select a path between some equal cost paths is a local configuration matter. This draft defines a method to use part of the 20-bit entropy label in each router, and this needs a data-plane to do some bit operation function. It is expected to be easier than hashing function.

6. Security Considerations

This document introduces no new security considerations beyond those already specified in [RFC8279] and [RFC8296].

7. IANA Considerations

This document contains no actions for IANA.

8. Acknowledgements

TBD.

9. References

9.1. Normative References

- [I-D.ietf-mpls-spring-entropy-label]
Kini, S., Kompella, K., Sivabalan, S., Litkowski, S., Shakir, R., and J. Tantsura, "Entropy label for SPRING tunnels", draft-ietf-mpls-spring-entropy-label-11 (work in progress), May 2018.
- [I-D.ietf-spring-segment-routing-msdc]
Filsfils, C., Previdi, S., Dawra, G., Aries, E., and P. Lapukhov, "BGP-Prefix Segment in large-scale data centers", draft-ietf-spring-segment-routing-msdc-09 (work in progress), May 2018.

- [RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <<https://www.rfc-editor.org/info/rfc7938>>.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.

9.2. Informative 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>>.

Authors' Addresses

Jingrong Xie
Huawei Technologies

Email: xiejingrong@huawei.com

Xiaohu Xu
Alibaba Inc.

Email: xiaohu.xxh@alibaba-inc.com

Gang Yan
Huawei Technologies

Email: yangang@huawei.com

Mike McBride
Huawei Technologies

Email: mmcbride7@gmail.com