ROLL
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2013

J. Hui
Cisco
R. Kelsey
Silicon Labs

October 19, 2012

Multicast Protocol for Low power and Lossy Networks (MPL) draft-ietf-roll-trickle-mcast-02

Abstract

This document specifies the Multicast Protocol for Low power and Lossy Networks (MPL) that provides IPv6 multicast forwarding in constrained networks. MPL avoids the need to construct or maintain any multicast forwarding topology, disseminating messages to all MPL forwarders in an MPL domain. MPL uses the Trickle algorithm to drive packet transmissions for both control and data-plane packets. Specific Trickle parameter configurations allow MPL to trade between dissemination latency and transmission efficiency.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 22, 2013.

Table of Contents

1. Introduction

Low power and Lossy Networks typically operate with strict resource constraints in communication, computation, memory, and energy. Such resource constraints may preclude the use of existing IPv6 multicast topology and forwarding mechanisms. Traditional IP multicast forwarding typically relies on topology maintenance mechanisms to forward multicast messages to all subscribers of a multicast group. However, maintaining such topologies in LLNs is costly and may not be feasible given the available resources.

Memory constraints may limit devices to maintaining links/routes to one or a few neighbors. For this reason, the Routing Protocol for LLNs (RPL) specifies both storing and non-storing modes [RFC6550]. The latter allows RPL routers to maintain only one or a few default routes towards a LLN Border Router (LBR) and use source routing to forward packets away from the LBR. For the same reasons, a LLN device may not be able to maintain a multicast forwarding topology when operating with limited memory.

Furthermore, the dynamic properties of wireless networks can make the cost of maintaining a multicast forwarding topology prohibitively expensive. In wireless environments, topology maintenance may involve selecting a connected dominating set used to forward multicast messages to all nodes in an administrative domain. However, existing mechanisms often require two-hop topology information and the cost of maintaining such information grows polynomially with network density.

This document specifies the Multicast Protocol for Low power and Lossy Networks (MPL), which provides IPv6 multicast forwarding in constrained networks. MPL avoids the need to construct or maintain any multicast forwarding topology, disseminating multicast messages to all MPL forwarders in an MPL domain. By using the Trickle algorithm [RFC6206], MPL requires only small, constant state for each MPL device that initiates disseminations. The Trickle algorithm also allows MPL to be density-aware, allowing the communication rate to scale logarithmically with density.

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 RFC 2119 [RFC2119].

The following terms are used throughout this document:

MPL forwarder
: An IPv6 router that subscribes to the MPL multicast group and participates in disseminating MPL multicast packets.

MPL multicast scope
: The multicast scope that MPL uses when forwarding MPL multicast packets. In other words, the multicast scope of the IPv6 Destination Address of an MPL multicast packet.

MPL domain
: A connected set of MPL forwarders that define the extent of the MPL dissemination process. As a form of flood, all MPL forwarders in an MPL domain will receive MPL multicast packets. The MPL domain MUST be composed of at least one MPL multicast scope and MAY be composed of multiple MPL multicast scopes.

MPL seed
: A MPL forwarder that begins the dissemination process for an MPL multicast packet. The MPL seed may be different than the source of the original multicast packet.

MPL seed identifier
: An identifier that uniquely identifies an MPL forwarder within its MPL domain.

original multicast packet
: An IPv6 multicast packet that is disseminated using MPL.

MPL multicast packet
: An IPv6 multicast packet that contains an MPL Hop-by-Hop Option. When either source or destinations are beyond the MPL multicast scope, the MPL multicast packet is an IPv6-in-IPv6 packet that contains an MPL Hop-by-Hop Option in the outer IPv6 header and encapsulates an original multicast packet. When both source and destinations are within the MPL multicast scope, the MPL Hop-by-Hop Option may be included directly within the original multicast packet.

3. Overview

MPL delivers IPv6 multicast packets by disseminating them to all MPL forwarders within an MPL domain. MPL dissemination is a form of flood. An MPL forwarder may broadcast/multicast an MPL multicast packet out of the same physical interface on which it was received. Using link-layer broadcast/multicast allows MPL to forward multicast packets without explicitly identifying next-hop destinations. An MPL forwarder may also broadcast/multicast MPL multicast packets out other interfaces to disseminate the message across different links. MPL does not build or maintain a multicast forwarding topology to forward multicast packets.

Any MPL forwarder may initiate the dissemination process by serving as an MPL seed for an original multicast packet. The MPL seed may or may not be the same device as the source of the original multicast packet. When the original multicast packet's source is outside the LLN, the MPL seed may be the ingress router. Even if an original multicast packet source is within the LLN, the source may first forward the multicast packet to the MPL seed using IPv6-in-IPv6 tunneling. Because MPL state requirements grows with the number of active MPL seeds, limiting the number of MPL seeds reduces the amount of state that MPL forwarders must maintain.

Because MPL typically broadcasts/multicasts MPL packets out of the same interface on which they were received, MPL forwarders are likely to receive an MPL multicast packet more than once. The MPL seed tags each original multicast packet with an MPL seed identifier and a sequence number. The sequence number provides a total ordering of MPL multicast packets disseminated by the MPL seed.

MPL defines a new IPv6 Hop-by-Hop Option, the MPL Option, to include MPL-specific information along with the original multicast packet. Each IPv6 multicast packet that MPL disseminates includes the MPL Option. Because the original multicast packet's source and the MPL seed may not be the same device, the MPL Option may be added to the original multicast packet en-route. To allow Path MTU discovery to work properly, MPL encapsulates the original multicast packet in another IPv6 header that includes the MPL Option.

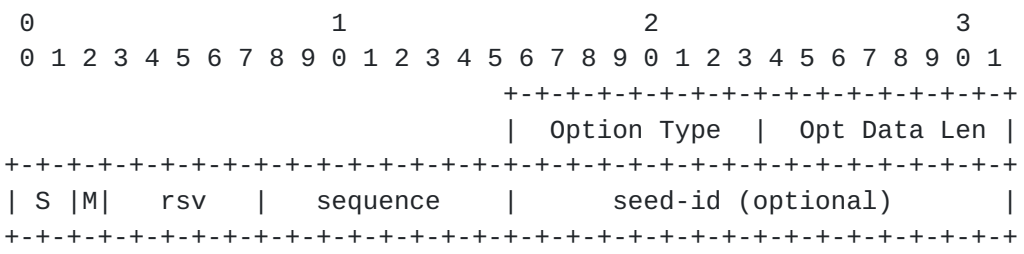
Upon receiving a new MPL multicast packet for forwarding, the MPL forwarder may proactively transmit the MPL multicast packet packet a limited number of times and then falls back into an optional reactive mode. In maintenance mode, an MPL forwarder buffers recently received MPL multicast packets and advertises a summary of recently received MPL multicast packets from time to time, allowing neighboring MPL forwarders to determine if they have any new multicast packets to offer or receive.

MPL forwarders schedule their packet (control and data) transmissions using the Trickle algorithm [RFC6206]. Trickle's adaptive transmission interval allows MPL to quickly disseminate messages when there are new MPL multicast packets, but reduces transmission overhead as the dissemination process completes. Trickle's suppression mechanism and transmission time selection allow MPL's communication rate to scale logarithmically with density.

4. Message Formats

4.1. MPL Option

The MPL Option is carried in an IPv6 Hop-by-Hop Options header, immediately following the IPv6 header. The MPL Option has the following format:



Option Type: XX (to be confirmed by IANA).

Opt Data Len: Length of the Option Data field in octets. MUST be set to either 2 or 4.

S: 2-bit unsigned integer. Identifies the length of seed-id. 0 indicates that the seed-id is 0 and not included in the MPL Option. 1 indicates that the seed-id is a 16-bit unsigned integer. 2 indicates that the seed-id is a 64-bit unsigned integer. 3 indicates that the seed-id is a 128-bit unsigned integer.

M: 1-bit flag. 0 indicates that the value in sequence is not the greatest sequence number that was received from the MPL seed.

rsv: 5-bit reserved field. MUST be set to zero and incoming MPL multicast packets in which they are not zero MUST be dropped.

sequence: 8-bit unsigned integer. Identifies relative ordering of MPL multicast packets from the source identified by seed-id.

seed-id: Uniquely identifies the MPL seed that initiated dissemination of the MPL multicast packet. The size of seed-id is indicated by the S field.

The Option Data of the Trickle Multicast option MUST NOT change as the MPL multicast packet is forwarded. Nodes that do not understand

the Trickle Multicast option MUST discard the packet. Thus, according to [RFC2460] the three high order bits of the Option Type must be set to '010'. The Option Data length is variable.

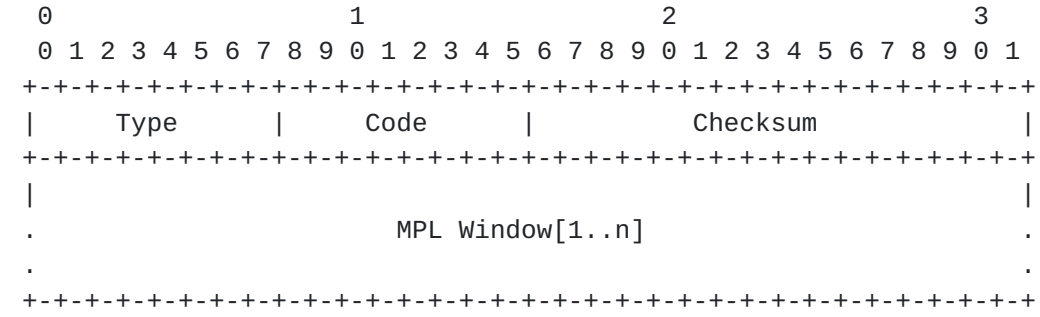
The seed-id uniquely identifies an MPL seed within the MPL domain. When seed-id is 128 bits (S=3), the MPL seed MAY use an IPv6 address assigned to one of its interfaces that is unique within the MPL domain. Managing MPL seed identifiers is not within scope of this document.

The sequence field establishes a total ordering of MPL multicast packets from the same MPL seed. The MPL seed MUST increment the sequence field's value on each new MPL multicast packet that it disseminates. Implementations MUST follow the Serial Number Arithmetic as defined in [RFC1982] when incrementing a sequence value or comparing two sequence values.

Future updates to this specification may define additional fields following the seed-id field.

4.2. ICMPv6 MPL Message

The MPL forwarder uses ICMPv6 MPL messages to advertise information about recently received MPL multicast packets. The ICMPv6 MPL message has the following format:



IP Fields:

Source Address A link-local address assigned to the sending interface.

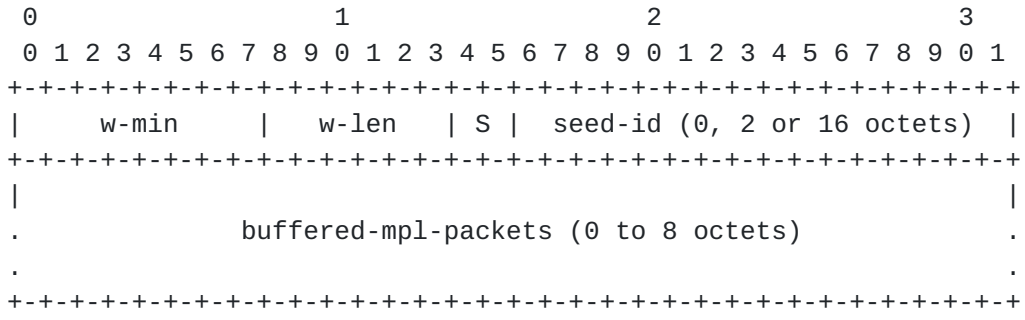
Destination Address The link-local all-nodes MPL forwarders multicast address (FF02::TBD).

Hop Limit	255
ICMPv6 Fields:	
Type	XX (to be confirmed by IANA).
Code	0
Checksum	The ICMP checksum. See [RFC4443].
MPL Window[1..n]	List of one or more MPL Windows (defined in Section 4.2.1).

An MPL forwarder transmits an ICMPv6 MPL message to advertise information about buffered MPL multicast packets. More explicitly, the ICMPv6 MPL message encodes the sliding window state (described in Section 5.2) that the MPL forwarder maintains for each MPL seed. The advertisement serves to indicate to neighboring MPL forwarders regarding newer messages that it may send or the neighboring MPL forwarders have yet to receive.

4.2.1. MPL Window

An MPL Window encodes the sliding window state (described in Section 5.2 that the MPL forwarder maintains for an MPL seed. Each MPL Window has the following format:



w-min	8-bit unsigned integer. Indicates the first sequence number associated with the first bit in buffered-mpl-packets.
w-len	6-bit unsigned integer. Indicates the size of the sliding window and the number of valid bits in buffered-mpl-packets. The sliding window's upper bound is the sum of w-min and w-len.

S — 2-bit unsigned integer. Identifies the length of seed-id. 0 indicates that the seed-id value is 0 and not included in the MPL Option. 1 indicates that the seed-id value is a 16-bit unsigned integer. 2 indicates that the seed-id value is a 128-bit unsigned integer. 3 is reserved.

seed-id — Indicates the MPL seed associated with this sliding window.

buffered-mpl-packets — Variable-length bit vector. Identifies the sequence numbers of MPL multicast packets that the MPL forwarder has buffered. The sequence number is determined by w-min + i, where i is the offset within buffered-mpl-packets.

The MPL Window does not have any octet alignment requirement.

5. MPL Forwarder Behavior

An MPL forwarder implementation needs to manage sliding windows for each active MPL seed. The sliding window allows the MPL forwarder to determine what multicast packets to accept and what multicast packets are buffered. An MPL forwarder must also manage MPL packet transmissions.

5.1. Multicast Packet Dissemination

MPL uses the Trickle algorithm to control packet transmissions when disseminating MPL multicast packets [RFC6206]. MPL provides two propagation mechanisms for disseminating MPL multicast packets.

1. With proactive propagation, an MPL forwarder transmits buffered MPL multicast packets using the Trickle algorithm. This method is called proactive propagation since an MPL forwarder actively transmits MPL multicast packets without discovering that a neighboring MPL forwarder has yet to receive the message.

2. With reactive propagation, an MPL forwarder transmits ICMPv6 MPL messages using the Trickle algorithm. An MPL forwarder only transmits buffered MPL multicast packets upon discovering that neighboring devices have not yet to receive the corresponding MPL multicast packets.

When receiving a new multicast packet, an MPL forwarder first utilizes proactive propagation to forward the MPL multicast packet. Proactive propagation reduces dissemination latency since it does not require discovering that neighboring devices have not yet received the MPL multicast packet. MPL forwarders utilize proactive propagation for newly received MPL multicast packets since they can assume that some neighboring MPL forwarders have yet to receive the MPL multicast packet. After a limited number of MPL multicast packet transmissions, the MPL forwarder may terminate proactive propagation for the MPL multicast packet.

An MPL forwarder may optionally use reactive propagation to continue the dissemination process with lower communication overhead. With reactive propagation, neighboring MPL forwarders use ICMPv6 MPL messages to discover new MPL multicast messages that have not yet been received. When discovering that a neighboring MPL forwarder has not yet received a new MPL multicast packet, the MPL forwarder enables proactive propagation again.

5.1.1. Trickle Parameters and Variables

As specified in RFC 6206 [RFC6206], a Trickle timer runs for a defined interval and has three configuration parameters: the minimum interval size Imin, the maximum interval size Imax, and a redundancy constant k.

MPL defines a fourth configuration parameter, TimerExpirations, which indicates the number of Trickle timer expiration events that occur before terminating the Trickle algorithm.

Each MPL forwarder maintains a separate Trickle parameter set for the proactive and reactive propagation methods. TimerExpirations MUST be greater than 0 for proactive propagation. TimerExpirations MAY be set to 0 for reactive propagation, which effectively disables reactive propagation.

As specified in RFC 6206 [RFC6206], a Trickle timer has three variables: the current interval size I, a time within the current interval t, and a counter c.

MPL defines a fourth variable, e, which counts the number of Trickle timer expiration events since the Trickle timer was last reset.

5.1.2. Proactive Propagation

With proactive propagation, the MPL forwarder transmits buffered MPL multicast packets using the Trickle algorithm. Each buffered MPL multicast packet that is proactively being disseminated with proactive propagation has an associated Trickle timer. Adhering to Section 5 of RFC 6206 [RFC6206], this document defines the following:

- This document defines a "consistent" transmission for proactive propagation as receiving an MPL multicast packet that has the same MPL seed identifier and sequence number as a buffered MPL packet.

- This document defines an "inconsistent" transmission for proactive propagation as receiving an MPL multicast packet that has the same MPL seed identifier, the M flag set, and has a sequence number less than the buffered MPL multicast packet's sequence number.

- This document does not define any external "events".

- This document defines both MPL multicast packets and ICMPv6 MPL multicast packets as Trickle messages. These messages are defined in the sections below.

- The actions outside the Trickle algorithm that the protocol takes involve managing sliding window state, and is specified in Section 5.2.

5.1.3. Reactive Propagation

With reactive propagation, the MPL forwarder transmits ICMPv6 MPL messages using the Trickle algorithm. A MPL forwarder maintains a single Trickle timer for reactive propagation with each MPL domain. When REACTIVE_TIMER_EXPIRATIONS is 0, the MPL forwarder does not execute the Trickle algorithm for reactive propagation and reactive propagation is disabled. Adhering to Section 5 of RFC 6206 [RFC6206], this document defines the following:

- This document defines a "consistent" transmission for reactive propagation as receiving an ICMPv6 MPL message that indicates neither the receiving nor transmitting node has new MPL multicast packets to offer.

- This document defines an "inconsistent" transmission for reactive propagation as receiving an ICMPv6 MPL message that indicates either the receiving or transmitting node has at least one new MPL multicast packet to offer.

- This document defines an "event" for reactive propagation as updating any sliding window (i.e. changing the value of WindowMin, WindowMax, or the set of buffered MPL multicast packets) in response to receiving an MPL multicast packet.

- This document defines both MPL multicast packets and ICMPv6 MPL multicast packets as Trickle messages. These messages are defined in the sections below.

- The actions outside the Trickle algorithm that the protocol takes involve managing sliding window state, and is specified in Section 5.2.

5.2. Sliding Windows

Every MPL forwarder MUST maintain a sliding window of sequence numbers for each MPL seed of recently received MPL packets. The sliding window performs two functions:

1. Indicate what MPL multicast packets the MPL forwarder should accept.

2. Indicate what MPL multicast packets are buffered and may be transmitted to neighboring MPL forwarders.

Each sliding window logically consists of:

1. A lower-bound sequence number, WindowMin, that represents the sequence number of the oldest MPL multicast packet the MPL forwarder is willing to receive or has buffered. An MPL forwarder MUST ignore any MPL multicast packet that has sequence value less than than WindowMin.

2. An upper-bound sequence value, WindowMax, that represents the sequence number of the next MPL multicast packet that the MPL forwarder expects to receive. An MPL forwarder MUST accept any MPL multicast packet that has sequence number greater than or equal to WindowMax.

3. A list of MPL multicast packets, BufferedPackets, buffered by the MPL forwarder. Each entry in BufferedPackets MUST have a sequence number in the range [WindowMin, WindowMax).

4. A timer, HoldTimer, that indicates the minimum lifetime of the sliding window. The MPL forwarder MUST NOT free a sliding window before HoldTimer expires.

When receiving an MPL multicast packet, if no existing sliding window exists for the MPL seed, the MPL forwarder MUST create a new sliding window before accepting the MPL multicast packet. The MPL forwarder may reclaim memory resources by freeing a sliding window for another MPL seed if its HoldTimer has expired. If, for any reason, the MPL forwarder cannot create a new sliding window, it MUST discard the packet.

If a sliding window exists for the MPL seed, the MPL forwarder MUST ignore the MPL multicast packet if the packet's sequence number is less than WindowMin or appears in BufferedPackets. Otherwise, the MPL forwarder MUST accept the packet and determine whether or not to forward the packet and/or pass the packet to the next higher layer.

When accepting an MPL multicast packet, the MPL forwarder MUST update the sliding window based on the packet's sequence number. If the sequence number is not less than WindowMax, the MPL forwarder MUST set WindowMax to 1 greater than the packet's sequence number. If WindowMax - WindowMin > MPL_MAX_WINDOW_SIZE, the MPL forwarder MUST increment WindowMin such that WindowMax - WindowMin <= MPL_MAX_WINDOW_SIZE. At the same time, the MPL forwarder MUST free any entries in BufferedPackets that have a sequence number less than WindowMin.

If the MPL forwarder has available memory resources, it MUST buffer the MPL multicast packet for proactive propagation. If not enough

memory resources are available to buffer the packet, the MPL forwarder MUST increment WindowMin and free entries in BufferedPackets that have a sequence number less than WindowMin until enough memory resources are available. Incrementing WindowMin will ensure that the MPL forwarder does not accept previously received packets.

An MPL forwarder MAY reclaim memory resources from sliding windows for other MPL seeds. If a sliding window for another MPL seed is actively disseminating messages and has more than one entry in its BufferedPackets, the MPL forwarder may free entries for that MPL seed by incrementing WindowMin as described above.

If the MPL forwarder cannot free enough memory resources to buffer the MPL multicast packet, the MPL forwarder MUST set WindowMin to 1 greater than the packet's sequence number.

When memory resources are available, an MPL forwarder SHOULD buffer a MPL multicast packet until the proactive propagation completes (i.e. the Trickle algorithm stops execution) and MAY buffer for longer. After proactive propagation completes, the MPL forwarder may advance WindowMin to the packet's sequence number to reclaim memory resources. When the MPL forwarder no longer buffers any packets, it MAY set WindowMin equal to WindowMax. When setting WindowMin equal to WindowMax, the MPL forwarder MUST initialize HoldTimer to WINDOW_HOLD_TIME and start HoldTimer. After HoldTimer expires, the MPL forwarder MAY free the sliding window to reclaim memory resources.

5.3. Transmission of MPL Multicast Packets

The MPL forwarder manages buffered MPL multicast packet transmissions using the Trickle algorithm. When adding a packet to BufferedPackets, the MPL forwarder MUST create a Trickle timer for the packet and start execution of the Trickle algorithm.

After PROACTIVE_TIMER_EXPIRATIONS Trickle timer events, the MPL forwarder MUST stop executing the Trickle algorithm. When a buffered MPL multicast packet does not have an active Trickle timer, the MPL forwarder MAY free the buffered packet by advancing WindowMin to 1 greater than the packet's sequence number.

Each interface that supports MPL is configured with exactly one MPL multicast scope. The MPL multicast scope MUST be site-local or smaller and defaults to link-local. A scope larger than link-local MAY be used only when that scope corresponds exactly to the MPL domain.

An MPL domain may therefore be composed of one or more MPL multicast scopes. For example, the MPL domain may be composed of a single MPL multicast scope when using a site-local scope. Alternatively, the MPL domain may be composed of multiple MPL multicast scopes when using a link-local scope.

IPv6-in-IPv6 encapsulation MUST be used when using MPL to forward an original multicast packet whose source or destination address is outside the MPL multicast scope. IPv6-in-IPv6 encapsulation is necessary to support Path MTU discovery when the MPL forwarder is not the source of the original multicast packet. IPv6-in-IPv6 encapsulation also allows an MPL forwarder to remove the MPL Option when forwarding the original multicast packet over a link that does not support MPL. The destination address scope for the outer IPv6 header MUST be the MPL multicast scope.

When an MPL domain is composed of multiple MPL multicast scopes (e.g. when the MPL multicast scope is link-local), an MPL forwarder MUST decapsulate and encapsulate the original multicast packet when crossing between different MPL multicast scopes. In doing so, the MPL forwarder MUST duplicate the MPL Option, unmodified, in the new outer IPv6 header.

The IPv6 destination address of the MPL multicast packet is the all-MPL-forwarders multicast address (TBD). The scope of the IPv6 destination address is set to the MPL multicast scope.

5.4. Reception of MPL Multicast Packets

Upon receiving an MPL multicast packet, the MPL forwarder first determines whether or not to accept and buffer the MPL multicast packet based on its MPL seed and sequence value, as specified in Section 5.2.

If the MPL forwarder accepts the MPL multicast packet, the MPL forwarder determines whether or not to deliver the original multicast packet to the next higher layer. For example, if the MPL multicast packet uses IPv6-in-IPv6 encapsulation, the MPL forwarder removes the outer IPv6 header, which also removes MPL Option.

5.5. Transmission of ICMPv6 MPL Messages

The MPL forwarder generates and transmits a new ICMPv6 MPL message whenever Trickle requests a transmission. The MPL forwarder includes an encoding of each sliding window in the ICMPv6 MPL message.

Each sliding window is encoded using an MPL Window entry, defined in Section 5.2. The MPL forwarder sets the MPL Window fields as

follows:

S If the MPL seed identifier is 0, set S to 0. If the MPL seed identifier is within the range [1, 65535], set S to 2. Otherwise, set S to 3.

w-min Set to the lower bound of the sliding window (i.e. WindowMin).

w-len Set to the length of the window (i.e. WindowMax - WindowMin).

seed-id If S is non-zero, set to the MPL seed identifier.

buffered-mpl-packets Set each bit that represents a sequence number of a packet in BufferedPackets to 1. Set all other bits to 0. The i'th bit in buffered-mpl-packets represents a sequence number of w-min + i.

5.6. Reception of ICMPv6 MPL Messages

An MPL forwarder processes each ICMPv6 MPL message that it receives to determine if it has any new MPL multicast packets to receive or offer.

An MPL forwarder determines if a new MPL multicast packet has not been received from a neighboring node if any of the following conditions hold true:

1. The ICMPv6 MPL message includes an MPL Window for an MPL seed that does not have a corresponding sliding window entry on the MPL forwarder.

2. The neighbor has a packet in its BufferedPackets that has sequence value greater than or equal to WindowMax (i.e. w-min + w-len >= WindowMax).

3. The neighbor has a packet in its BufferedPackets that has sequence number within range of the sliding window but is not included in BufferedPackets (i.e. the i'th bit in buffered-mpl-packets is set to 1, where the sequence number is w-min + i).

When an MPL forwarder determines that it has not yet received a new MPL multicast packet buffered by a neighboring device, the MPL forwarder resets the Trickle timer associated with reactive propagation.

An MPL forwarder determines if an entry in BufferedPackets has not been received by a neighboring MPL forwarder if any of the following

conditions hold true:

1. The ICMPv6 MPL message does not include an MPL Window for the packet's MPL seed.

2. The packet's sequence number is greater than or equal to the neighbor's WindowMax value (i.e. the packet's sequence number is greater than or equal to w-min + w-len).

3. The packet's sequence number is within the range of the neighbor's sliding window [WindowMin, WindowMax), but not included in the neighbor's BufferedPacket (i.e. the packet's sequence number is greater than or equal to w-min, strictly less than w-min + w-len, and the corresponding bit in buffered-mpl-packets is set to 0.

When an MPL forwarder determines that it has at least one buffered MPL multicast packet that has not yet been received by a neighbor, the MPL forwarder resets the Trickle timer associated with reactive propagation. Additionally, for each buffered MPL multicast packet that should be transferred, the MPL forwarder MUST reset the Trickle timer and reset e to 0 for proactive propagation. If the Trickle timer for proactive propagation has already stopped execution, the MPL forwarder MUST initialize a new Trickle timer and start execution of the Trickle algorithm.

6. MPL Parameters

An MPL forwarder maintains two sets of Trickle parameters for the proactive and reactive methods. The Trickle parameters are listed below:

PROACTIVE_IMIN The minimum Trickle timer interval, as defined in [RFC6206] for proactive propagation.

PROACTIVE_IMAX The maximum Trickle timer interval, as defined in [RFC6206] for proactive propagation.

PROACTIVE_K The redundancy constant, as defined in [RFC6206] for proactive propagation.

PROACTIVE_TIMER_EXPIRATIONS The number of Trickle timer expirations that occur before terminating the Trickle algorithm. MUST be set to a value greater than 0.

REACTIVE_IMIN The minimum Trickle timer interval, as defined in [RFC6206] for reactive propagation.

REACTIVE_IMAX The maximum Trickle timer interval, as defined in [RFC6206] for reactive propagation.

REACTIVE_K The redundancy constant, as defined in [RFC6206] for reactive propagation.

REACTIVE_TIMER_EXPIRATIONS The number of Trickle timer expirations that occur before terminating the Trickle algorithm. MAY be set to 0, which disables reactive propagation.

WINDOW_HOLD_TIME The minimum lifetime for sliding window state.

7. Acknowledgements

The authors would like to acknowledge the helpful comments of Robert Cragie, Esko Dijk, Ralph Droms, Paul Duffy, Owen Kirby, Joseph Reddy, Dario Tedeschi, and Peter van der Stok, which greatly improved the document.

8. IANA Considerations

The Trickle Multicast option requires an IPv6 Option Number.

HEX	act	chg	rest
C	01	0	TBD

The first two bits indicate that the IPv6 node MUST discard the packet if it doesn't recognize the option type, and the third bit indicates that the Option Data MUST NOT change en-route.

9. Security Considerations

TODO.

10. References

10.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

10.2. Informative References

[I-D.ietf-roll-terminology]
Vasseur, J., "Terminology in Low power And Lossy Networks", draft-ietf-roll-terminology-06 (work in progress), September 2011.

Authors' Addresses

Jonathan W. Hui
Cisco
170 West Tasman Drive
San Jose, California 95134
USA

Phone: +408 424 1547
Email: jonhui@cisco.com

Richard Kelsey
Silicon Labs
25 Thomson Place
Boston, Massachusetts 02210
USA

Phone: +617 951 1225
Email: richard.kelsey@silabs.com