Workgroup: PIM
Internet-Draft: draft-eckert-msr6-rbs-00
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
Authors: T. Eckert
Futurewei Technologies USA
X. Geng
Huawei 2012 NT Lab
X. Zheng
Huawei 2012 NT Lab
R. Meng
Huawei 2012 NT Lab
F. Li
Huawei 2012 NT Lab

Recursive Bitstring Structure (RBS) for Multicast Source Routing over IPv6 (MSR6)

Abstract

This document defines an encoding and corresponding packet processing procedures for native IPv6 multicast source routing (MSR6) using a so-called "Recursive Bitstring" (RBS) address structure.

The RBS address structure encodes the source-routed multicast tree as a tree of bitstrings. Each router on the tree only needs to examine and perform replication for the one bitstring destined for it.

The MSR6/RBS IPv6 extension header encoding and processing is modeled after that of unicast source routing headers, RFC6554 and RFC8754, and shares all elements that can be shared. To support the RBS structure, it is replacing the "Segments Left" pointer to the next segment with two fields to point to the next sub-tree to parse.

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 12 January 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Overview

1.1. Introduction

Eliminating hop-by-hop per-multicast-tree state in the forwarding plane as well as the per-hop, per-tree control plane complexity that goes along with it has long since been a concern against the deployment of multicast services. Short of MSR6, there are no IETF standardized mechanisms to enable this with native hop-by-hop IPv6 forwarding according to [RFC8200] and per-hop stateless replication.

"Multicast Source Routing over IPv6" (MSR6), is such a stateless, native IPv6 forwarding based multicast source routing (MSR6) solution, defined in [I-D.cheng-spring-ipv6-msr-design-consideration].

MSR6 intends to be compatible with and reuse all the IPv6 mechanisms introduced by prior stateless hop-by-hop native IPv6 unicast forwarding, including [RFC6554] (IPv6 Source Routing Header for networks using RPL routing), and [RFC8754] (IPv6 Segment Routing Header for SRv6). The MSR6 extension header and semantic shares as much as possible with these unicast approaches. It especially attempts to allow introducing MSR6 as the multicast extension to for the IPv6 Segment Routing architecture called SRv6 ([RFC8402] and [RFC8986]).

MSR6 considers two basic modes of forwarding: one is based on Shortest Path First(SPF). In this mode, the tree only encodes tree leaves in the extension header, but no traffic steering. This is called MSR6 BE mode. The other mode is based on path steering with replication, which is called MSR6 TE mode. [I-D.geng-msr6-traffic-engineering], [I-D.chen-pim-srv6-p2mp-path] and [I-D.geng-msr6-rlb-segment] have introduced structured segment lists in support of MSR6 TE mode.

This document proposes a variant of an MSR6 extension header that uses the "Recursive Bitstrings" (RBS) address structure to encode the source-routed multicast tree as a tree of bitstrings, in support of MSR6 TE mode. Each router on the tree only needs to examine and perform replication for the one bitstring encoded in the RBS-Address for that MSR.

The logic of MSR6/RBS replication and tree representation is derived (and simplified) from the BIER-TE [I-D.ietf-bier-te-arch] architecture. The RBS address structure replaces a single, end-to-end "flat" bitstring used in BIER-TE. This eliminates the scalability and controller-plane complexity of BIER-TE.

Likewise, MSR6/RBS forwarding is based on the architecture specified in [I-D.eckert-bier-cgm2-rbs]. Because this document intends to only specify the forwarding specification, it does not cover the system architecture details. Please refer to [I-D.ietf-bier-te-arch] and [I-D.eckert-bier-cgm2-rbs] for system level details, such as scalability and complexity comparisons.

A comparison between this document and [I-D.xu-msr6-rbs] and [I-D.eckert-bier-cgm2-rbs] is given below in Appendix A.

1.2. Forwarding overview

In MSR6/RBS, routers are IPv6 MSR6 Segment Routers (MSR). An ingress MSR (MSIR) forms an IPv6 packet and includes a Multicast Source Routing Header (MRH) that uses the RBS format. The MRH controls the steering and replication of the packet across one or more MSR6 Segment Routers (MSR), terminating the packet in one or more egress MSR (MSER).

Note that the terms MSR, MSIR and MSER are chosen to be replicating the terms BFR, BFIR and BFER used for equivalent router roles in BIER [RFC8279] and BIER-TE [I-D.ietf-bier-te-arch]. BIER and BIER-TE are based on a separate L2.5 forwarding mechanism and encapsulation, optimized for MPLS networks (see [RFC8296]).

Figure 1 shows an example network topology and an example multicast tree. R1 has connections to connections to R2, R3, R4, R5 (not shown) and R6. For the purpose of explaining RBS, it is irrelevant whether those connections are separate L2 point-to-point links, links or adjacencies on a shared LAN. Likewise, R3 has connections to R1, R7, R8, R9 and R10, R4 has connections to R1, R7, R8, R8 and R10, and and R9 has connections to R3, R4, and some additional unnamed MSR.

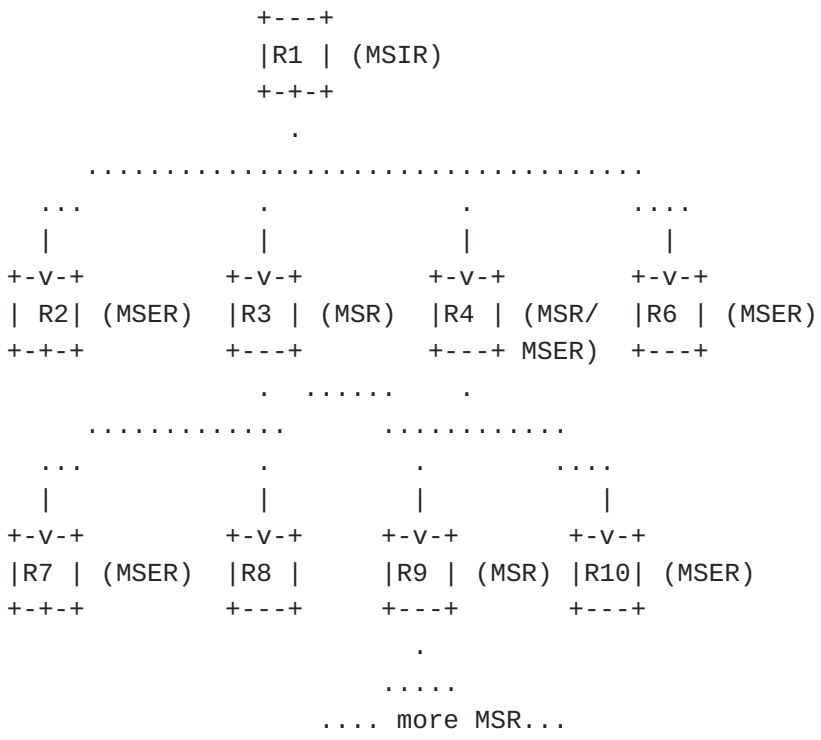

Figure 1: Example Topology/RBS tree

R1 wants to send a packet that is to be received by R2, R4, R6, R7, R10 and some MSER behind R9. Given how R7, R8, R8, R10 and the MSR behind R9 can be reached via both either R3 and R4, there is a packet steering and replication (traffic engineering) choice to be made: R3 should forward and replicate to R8 and R8, and R4 should replicate to to R9 (to reach the msr behind it, and R10.

Every MSR has an RBS "Bit Index Forwarding Table" (RBS-BIFT) that defines which BitPosition (BP) (1..N) indicates which adjacency. Figure 2, shows the example RBS-BIFT for R1.

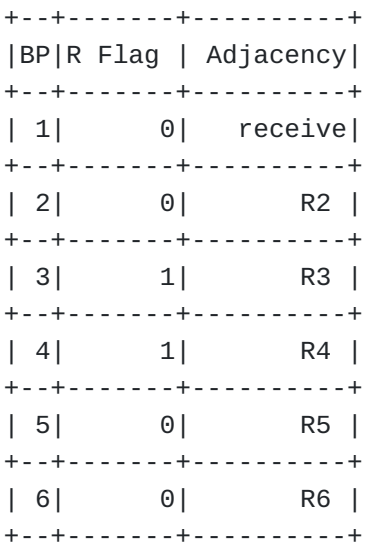

Figure 2: BIFT on R1

The receive adjacency is the bit position indicating that the packet is destined for the router itself. The R)ecursive flag indicates whether the adjacency is an intermediate MSR that acts as a replication point to further MSR. If an MSR is never a transit but can always only be a leaf in a multicast distribution tree, then R=0. This allows for more compact encoding of the RBS address structure. In the example, R2, R5 and R6 are connected to R1 and also leaf router in the topology, hence they have R=0 in the R1 RBS-BIFT.

When a router receives and processes an IPv6 packet with an MRH that uses the RBS address structure, the router needs to only act upon the "RecursiveUnit" (RU) within that address structure destined to it.

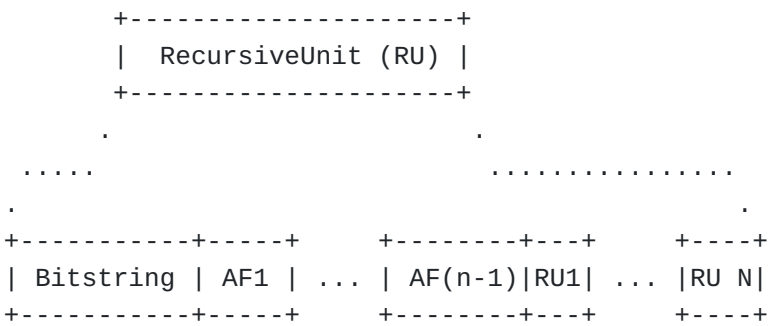

Figure 3: Structure of Recursive Unit

As shown in Figure 3, a Recursive Unit (RU) starts with the Bitstring for the MSR to which this RU is intended. In the example,

the first MSR is R1, so the Bitstring in the RU is as shown in Figure 4

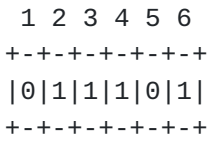

Figure 4: Bitstring for R1 in the example

When an MSR processes its RU, the length of the BS is derived from the length of the BIFT. In the case of R1 it is therefore known to be 6 bits long.

To support replication via intermediate MSR, the RBS address structure needs to contain for each of those MSR a separate RU. In the example, the packet is to be further replicated by R3 and R4 and then further on by R9.

The RU for R1 therefore needs to contain two further RU, one for R3 and one for R4. The one for R4 will also need to contain RUs for the MSR below it.

When creating packet copies to R3 and R4, R1 needs to rewrite the MRH such that R3 and R4 will find their respective RU. Therefore, R1 needs to be able to parse its own RU such that it can locate those further RU for R3 and R4. This is supported by the AddressFields (AF) following the BS. Each AF indicates the length of one RU that follows.

When N (in the example N=2) RU follow, only N-1 (in the example 1) AF are needed, because the length of the N'th RU can be calculated from the length of the RU minus the sum of the length of the other RU as indicated in the N-1 AF.

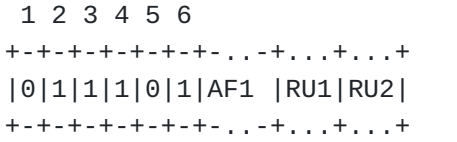

Figure 5: RU for R1

In result, the RU for R1 looks as shown in Figure 5. It has the aforementioned 6-bit long Bitstring because the BIFT of R1 is 6 BP long, it has one AF1 indicating the length of RU1, which is the RU for the first set BP in the Bitstring with R=1, so it is for R3, and the RU finishes with RU2 for the second BP in the BS with R=1, so it is for R4.

2. Specification

2.1. RBS-Address

As shown in Figure 3 and explained in Section 1.2, an RBS address consists of the RU for the first MSR of a tree and is composed of a Bitstring for this MSR, the AddressFields for all but the last bits (N-1) set in that Bitstring with R=1 flag in the BIFT, followed by N RU for each of those bits, which are recursivly composed in the same way.

The RU for any MSR only needs to be decoded (in high-speed hardware) by the MSR itself, but not any other MSR (along the path/tree). Creation of an MSR is assumed to be part of application/network stack on hosts or router control plane software and is therefore assumed to be able to support arbitrary formats of the AF fields, as long as there is a standard data model (e.g.: YANG) and/or control plane protocol specification (e.g.: OSPF or ISIS extensions) for it. specifically, different router (MSR) implementations may choose to support different AF formats.

Any MSR MUST support to decode RU where the AF entries are 8 bit in size. Any MSR SHOULD support to decode a variable length AF encoding, where 0XXXXXXX (8-bit length AF field) is used to encode a 7-bit XXXXXXX (0..127) values, and where 1XXXXXXXXXXXX is used to encode an 12-bit value XXXXXXXXXXX.

Note that in the MSR/RBS IPv6 extension header, the RBS-Address can be as long as 256 bytes. Therefore, non-support of any AF field not supporting to indicate RU lengths as long as 2048 bit may not allow to build maximum size MSR/RBS extension headers.

2.2. RBS-BIFT

RBS-BIFT are composed as explained in Section 1.2. Their size can be any number of entries from 2 to 1024 bits (2^10), resulting in equal length Bitstrings for the MSR in an RBS-Address.

The leftmost bit in an RBS RU Bitstrings is BIFT entry 1.

The adjacency is an IPv6 link-local, ULA or global IPv6 unicast address of the next-hop assigned to the BitPosition. Further requirements are explained in Section 2.6.

2.3. Multicast Source Routing (MSR6) Header with RBS Sub-type

2.3.1. MRH extension header (refresher)

The "Multicast Routing Header" (MRH) is a new [RFC8200] IPv6 routing header defined according to [I-D.geng-msr6-traffic-engineering] as follows.

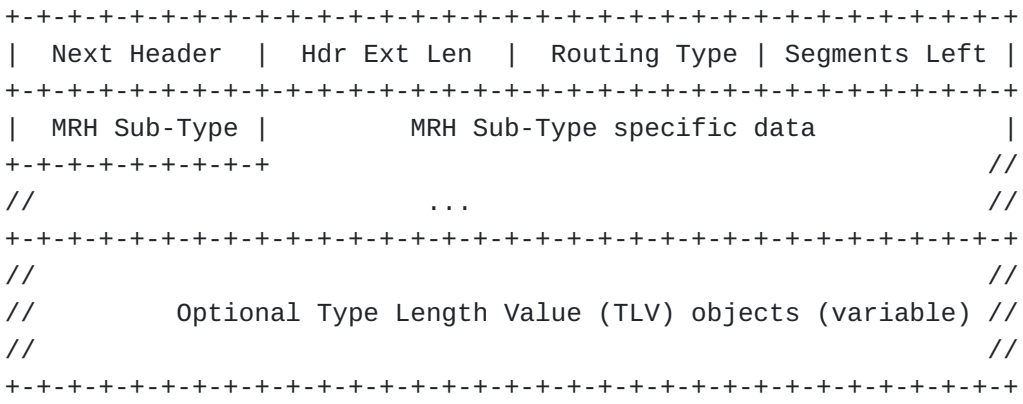

Figure 6: MRH format

Next Header: Defined in [RFC8200], section 4.4 (Type of the next header following so that it can be correctly parsed).

Hdr Ext Len: Defined in [RFC8200], section 4.4 (Length of the extension header in octets, not counting the first 8 octets).

Routing Type: Code point to be allocated (TBD1) for the RBS Sub-type for MRH (as part of a registry to be established for the MRH).

Segments Left: Filled with segments left according to [RFC8200], section 4.4, see Section 2.5.2.

The "Optional TLV objects" are intended to encode applicable TLV from SRH [RFC8754] or multicast/MRH specific TLVs. Examination of these TLV is based on their semantic. Current TLV defined in conjunction with [RFC8754] are examined upon reception of a packet, but not when forwarding the packet from one segment to another. In case of RBS, reception is triggered either by Segments Left being 0, or when parsing the Bitstring and acting upon the BP that is indicating the receive adjacency.

The RBS Sub-Type specific data contains the RBS address structure as follows.

2.4. MRH Sub-Type specific data for RBS

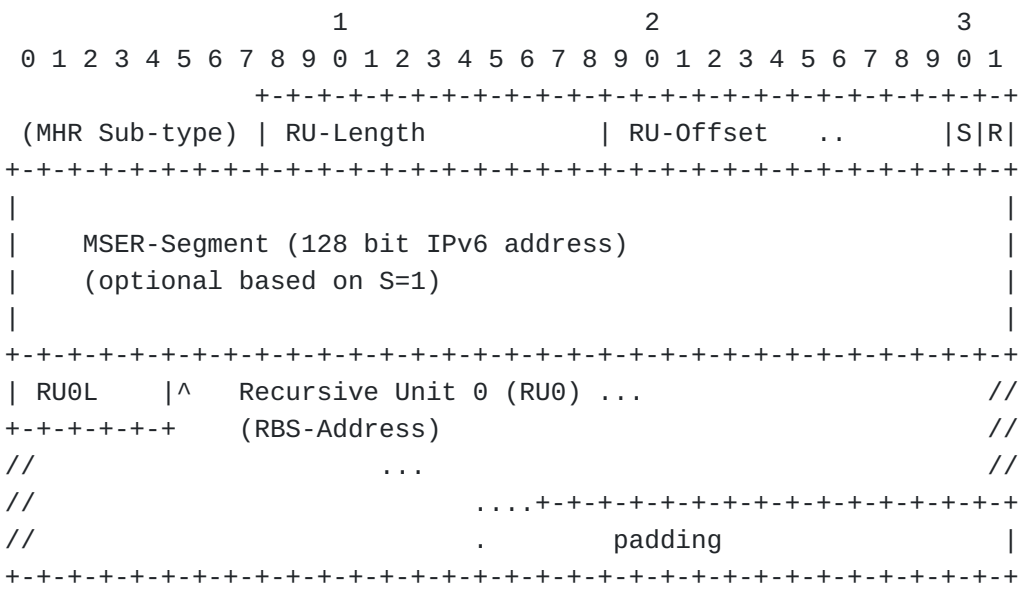

Figure 7: MRH Sub-type specific data for RBS (RBS-Address)

RBS-Address is the Recursive Unit as it is to be processed by the MSIR. It contains (as explained above recursively all the RU for MSR along the tree for this packet. Processing the complete RBS tree encoded across multiple MSR is defined here to be as processing a single End.RBS segment.

padding extends the RBS-Address field to 32-bit alignment. RU0L is the number of 32-bit units occupied by (RU0L, RBS-Address, padding).

MSER-Segment is a segment to be processed by MSER after the End.RBS segment. S is a flag that MUST be set to 1 when the MSER-Segment is present, else it MUST be set to 0. R is a reserved bit (MUST be ignored upon reception).

RU-Length (RecursiveUnit Length) is the length of the Recursive Unit to be examined by the processing MSR. It is counted in bits. Given how [RFC8200] Hdr Ext Len only allows for up to 255 bytes, RU-Length can at most be only 11 bits long.

RU-offset (Recursive Unit Offset) is the offset in bits of the Recursive Unit to be examined by the processing MSR, where 0 is the first bit of RBS-Address.

RU-Length and RU-offset are mutable, all other fields are immutable.

2.5. MRS6/RBS processing

[TBD: This section may need to be re-written with more formalistic language if the pseudocode (see below) is not a preferred formal description.]

2.5.1. MSIR header creation

Upon creation of the RBS header with an RBS-Address, RU-Length is set to the length of the RBS-Address, and RU-offset is set to 0.

MSER-Segment is included when the packet is an IPv6 multicast packet. In this case, the MSER-Segment carries the IPv6 Destination (multicast group) Address. The MSER-Segment MAY also contain any non IPv6 multicast group address when it has been defined/signaled accordingly as a SID for processing by all MSER that could potentially receive it. S is set according to the presence of MSER-Segment.

Segments Left is set to 2 if MSER-Segment is included, otherwise it is set to 1.

2.5.2. Common processing

When the MSR6/RBS header is received, including by the creating MSIR, the need to process the RBS-Address (End.RBS segment) is examined. This is the case when (Segments Left - 1) = 1. See below for further details. When the End.RBS is not to be processed, then the MSR it needs to act upon the header as an MSER.

2.5.3. MSER header processing

An MSER examines the presence of MSER-Segment (according to S). If present, and if MSER-Segment carries an IPv6 multicast address then the MSER copies the IPv6 multicast address into IPv6 Destination Address field, discards the MSR6/RBS extension header and proceeds with processing of the packet as an IPv6 multicast packet.

Note that non withstanding the previous paragraphs behavior, host stacks SHOULD maintain a copy of the MSR6/RBS extension header data so that socket / application code can retrieve for advanced functionality, such as identifying the path taken, as desirable for resilient transmission.

If the MSER-Segment is not an IPv6 multicast address, the packet is NOT an IPv6 Multicast packet and MUST NOT be further processed as an IPv6 Multicast Packet. Instead, the address MUST be accordingly registered as a SID by the control plane and further processing of the MSR6/RBS header is subject to the definition of the SID. If the address does not match a registered SID, the packet MUST be discarded and an error be raised.

If the MSER-Segment is not present, the router MUST remove the MSR6/RBS extension header and proceed processing with "receiving" the packet with the next header.

2.6. MSR processing of RBS-Address

When an MSR received a packet with MSR6/RBS extension header in which it needs to process the RBS-Address (End.RBS segment), it MUST first validate that the IPv6 Destination Address is a SID with End.RBS function.

It MUST be a link-local, ULA or global address on the router not used for any other functions (IPv6 unicast, Segment Routing). The ability to send packets to such addresses with End.RBS functions MUST be tightly controlled in the network to prohibit the ability of unauthorized senders to cause packet replication attacks by sending of packets with MSR6/RBS headers.

These requirements logically apply equally to the generating router (MSIR), but can of course appropriately be optimized in implementation.

After this ingress check, the MSR parses the RBS-Address field starting at RU-Offset, taking the RU-Length as a known parameter into account. This subset of the RU-Address is the RU for this MSR. Parsing is shown in more detail in Section 3.

Upon parsing, the MSR creates a packet copy for every BP set in Bitstring and rewrites it according to the following rules. It finally discards the received packet.

2.6.1. MSR processing of receive adjacency

Packet copies for a receive adjacency have their Segments Left reduced by 1 and then passed to MSER processing Section 2.5.3.

2.6.2. MSR per-hop processing

Per-hop processing of packets with MSR6/RBS extension header that include an MSER-Segment with an IPv6 multicast address are IPv6 multicast packets. In result, they inherit all per-hop IPv6 forwarding rules of [RFC8200], processing of any additional industry common per-hop rules for IPv6 multicast packets (as desirable by implementations), and additional per-hop applicable IPv6 extension headers.

For example, the IPv6 header Hopcount field is reduced on every hop, and the packet discarded if Hopcount reaches 0.

For example, routers/operators along the path may choose to support filtering of MSR6/RBS packets based on their IPv6 multicast destination address in the MSER-Segment field. Or perform IPFIX accounting against those addresses.

For example (TBD): For ECMP situations, the IPv6 Flow Label is used to choose a next-hop adjacency. This can include BIFT adjacencies that include multiple next-hop addresses/interfaces.

If the MSER-Segment is not present, or not carrying an IPv6 Multicast address, more liberty can be taken wrt. processing rules, especially through definition of additional SID Functions for MSER-Segment.

2.6.2.1. MSR processing for R=0 adjacency

Packet copies for for an adjacency to an MSR neighbor with R=0 have their Segments Left reduced by 1. RU-Length and RU-Offset SHOULD be set to 0.

The MSR neighbor IPv6 address/SID from the BIFT entry is copied into the IPv6 Destination Address field and the packet is forwarded (via IPv6 unicast forwarding procedures).

The MSR MUST only permit IP6 addresses in the RBS-BIFT for R=0/R=1 entries that have the End.RBS function.

2.6.2.2. MSR processing for R=1 adjacency

The MSR calculates new values for RU-Offset and RU-Length for a copy to an MSR neighbor with R=1. It updates the RU-Offset, RU-Length field in the MSR type-specific field for RBS.

The MSR neighbor IPv6 address/SID from the BIFT entry is copied into the IPv6 Destination Address field and the packet is forwarded (via IPv6 unicast forwarding procedures).

The MSR MUST only permit IP6 addresses in the RBS-BIFT for R=1 entries that have the End.RBS function.

3. MSR/RBS forwarding Pseudocode

The following example RBS forwarding Pseudocode assumes the reference encoding of bit-accurate length of Bitstrings and RecursiveUnits as well as 8-bit long TotalLen and AddressingField Lengths. All packet field addressing and address/offset calculations is therefore bit-accurate instead of byte accurate (which is what most CPU memory access today is).

```
void ProcessMSR6header(Packet)
{
  MSR6 = GetPacketMSR6Header(Packet);
  case (MSR6.MRHSubType)
    RBS) ProcessRBSSubtype(Packet); break
    // ... other MSR6 subtypes
  esac
}

void ProcessRBSSubtype(Packet)
{
  MSR6 = GetPacketMSR6Header(Packet);
  RBS = MSR6.MRHSubType

  if(MSR6.RULength == 0) return ReceiveRBSsubtype(Packet)

  RU0 = RBS + 29 + (MSR6.S ? 128 : 0)
  RU = RU0 + MSR6.RUOffset
  RUL = MSR6.RULength

  BitstringA = MSR6.RUOffset
  AddressingField =  BitstringA + BIFT.entries;

  // [1] calculate number of recursive bits set in Bitstring
  CopyBitstring(*BitstringA, *RecursiveBits, BIFT.entries);
  And(*RecursiveBits,*BIFTRecursiveBits, BIFT.entries);
  N = CountBits(*RecursiveBits, BIFT.entries);

  // Start of first RecursiveUnit in RBS address
  // After AddressingField array with 8-bit length fields
  RecursiveUnit = AddressingField + (N - 1) * 8;

  RemainLength = *(RBS.RULength);
  Index = GetFirstBitPosition(*BitstringA);
  while (Index) {
    PacketCopy = Copy(Packet);
    if (BIFT.BP[Index].adjacency == receive)
      ReceiveRBSsubtype(PacketCopy)
      next;
    }

    RBSc = RBS - Packet + PacketCopy
    MSR6c = MSR6 - Packet + PacketCopy
    If (BIFT.BP[Index].recursive) {
      if(N == 1) {
        RecursiveUnitLength = RemainLength;
      } else {
        RecursiveUnitLength = *AddressingField;
        N--;
        AddressingField += 8;
```

```
        RemainLength -= RecursiveUnitLength;
        RemainLength -= 8; // 8 bit of AddressingField
      }
      *(RBSc.RUOffset) = RecursiveUnit - RU0
      *(RBSc.RULength) = RecursiveUnitLength
      RecursiveUnit += RecursiveUnitLength;
    } else {
      *(RBSc.RUOffset) = 0
      *(RBSc.RULength) = 0
      *(MSR6c.SegmentsLeft) -= 1
    }
    *(PacketCopy.IPv6hdr.DA) = *(BIFT.BP[Index].adjacency)
    // ProcessMSR6TLV(Packet) - needed ?
    IPv6Forward(PacketCopy)
    Index = GetNextBitPosition(*BitstringA, Index);
  }
}

void ReceiveRBSsubtype(Packet)
{
  MSR6 = GetPacketMSR6Header(Packet);
  RBS = MSR6.MRHSubType
  if(MSR6.S) {
    *(Packet.IPv6hdr.DA) = *(RBS.MSETSegment)
    *(MSR6c.SegmentsLeft) = 0
  }
  ProcessMSR6TLV(Packet)
  // header not needed any further except for diagnostics
  // DisposeMSR6Header(Packet)
  if(IsIPv6MulticastAddr(Packet.IPv6hdr.DA))
    ReceiveIpv6Multicast(Packet)
  else
    ProcessSRv6DASID(Packet)
}
```

Figure 8: RBS forwarding Pseudocode

Explanations for Figure 8.

ProcessMSR6header(Packet) is called upon receipt of an IPv6 packet with an MSR6header. It is preceded by (not shown) standard IPv6 processing of a packet destined to an address of the node (such as HopCount processing), and other common processing of a Routing Header. This function only demultiplexes into the MSR6 option specific code.

ProcessRBSSubtype(Packet) processes the RBS option header. All address pointers shown use bit accurate addressing, because the elements of the RU are at bit-accurate offsets.

MSR6 is the address of the MSR6 extension header in the packet, RBS is the address of the RBS address in the packet.

BitstringA is the address of the RBS address Bitstring in memory. Other variables use names matching those from the packet header figures (without " -_").

The BFR local BIFT has a total number of BIFT.entries addressable BP 1...BIFTentries. The Bitstring therefore has BIFT.entries bits.

BIFT.RecursiveBits is a Bitstring pre-filled by the control plane with all the BP with the recursive flag set. This is constructed from the Recursive flag setting of the BP of the BIFT. The code starting at [1] therefore counts the number of recursive BP in the packets Bitstring.

Because the AddressingField does not have an entry for the last (or only) RecursiveUnit, its length has to be calculated By subtracting the length of the prior N-1 RecursiveUnits from RULength. This is done via variable RemainLength.

For every PacketCopy that is to be forwarded, the RU-Length, RU-Offset and IPv6 header DestinationAddress (DA) field are updated. For non-recursive adjacencies, the SegmentsLeft field is also updated.

For packet copies that are to be received by this node, The DA is updated from the RBS MSER-Segment field when present, and depending on what type of address it is, the packet continues to be processed as a received IPv6 Multicast packet or SRv6 SID.

4. IANA requests

This specification requests a TBD1 code point within a TBD registry of MRH extension header options.

5. Security considerations

The specification painstakingly attempts to ensure that IPv6 addresses used to deliver MSR6/RBS extension header packets are ONLY used for such packets such that common IPv6 "clamshell" filtering of address ranges can ensure that no unauthenticated sender (such as from outside the domain) can send packets to these addresses.

5.1. Changelog

[RFC-Editor: please remove this section].

This document is written in https://github.com/cabo/kramdown-rfc2629 markup language. This documents source is maintained at https://github.com/toerless/multicast-rbs, please provide feedback to the msr6@ietf.org mailing list and submit an Issue to the GitHub.

6. Acknowledgments

Many thanks for Bing Xu (bing.xu@huawei.com) for editorial work on the prior variation of this work [I-D.xu-msr6-rbs].

7. References

7.1. Normative References

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6

(SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

7.2. Informative References

[I-D.chen-pim-srv6-p2mp-path]
Chen, H., McBride, M., Fan, Y., Li, Z., Geng, X., Toy, M., Mishra, G. S., Wang, A., Liu, L., and X. Liu, "Stateless SRv6 Point-to-Multipoint Path", Work in Progress, Internet-Draft, draft-chen-pim-srv6-p2mp-path-06, 30 April 2022, <https://www.ietf.org/archive/id/draft-chen-pim-srv6-p2mp-path-06.txt>.

[I-D.cheng-spring-ipv6-msr-design-consideration]
Cheng, W., Mishra, G., Li, Z., Wang, A., Qin, Z., and C. Fan, "Design Consideration of IPv6 Multicast Source Routing (MSR6)", Work in Progress, Internet-Draft, draft-cheng-spring-ipv6-msr-design-consideration-01, 25 October 2021, <https://www.ietf.org/archive/id/draft-cheng-spring-ipv6-msr-design-consideration-01.txt>.

[I-D.eckert-bier-cgm2-rbs] Eckert, T. and B. (. Xu, "Carrier Grade Minimalist Multicast (CGM2) using Bit Index Explicit Replication (BIER) with Recursive BitString Structure (RBS) Addresses", Work in Progress, Internet-Draft, draft-eckert-bier-cgm2-rbs-01, 9 February 2022, <https://www.ietf.org/archive/id/draft-eckert-bier-cgm2-rbs-01.txt>.

[I-D.geng-msr6-rlb-segment] Geng, X., Li, Z., and J. Xie, "RLB (Replication through Local Bitstring) Segment for Multicast Source Routing over IPv6", Work in Progress, Internet-Draft, draft-geng-msr6-rlb-segment-00, 10 February 2022, <https://www.ietf.org/archive/id/draft-geng-msr6-rlb-segment-00.txt>.

[I-D.geng-msr6-traffic-engineering] Geng, X., Li, Z., and J. Xie, "IPv6 Multicast Source Routing Traffic Engineering", Work in Progress, Internet-Draft, draft-geng-msr6-traffic-engineering-01, 7 March 2022, <https://www.ietf.org/archive/id/draft-geng-msr6-traffic-engineering-01.txt>.

[I-D.ietf-bier-te-arch] Eckert, T., Menth, M., and G. Cauchie, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", Work in Progress, Internet-Draft, draft-ietf-bier-

te-arch-13, 25 April 2022, <https://www.ietf.org/archive/id/draft-ietf-bier-te-arch-13.txt>.

[I-D.xu-msr6-rbs] Xu, B., Geng, X., and T. Eckert, "RBS(Recursive BitString Structure) for Multicast Source Routing over IPv6", Work in Progress, Internet-Draft, draft-xu-msr6-rbs-01, 30 March 2022, <https://www.ietf.org/archive/id/draft-xu-msr6-rbs-01.txt>.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <https://www.rfc-editor.org/info/rfc6554>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

Appendix A. Background / Explanations

[TBD: This section to be removed, but maybe some explanations will make sense to move into different sections.]

A.1. Evolution from draft-xu-msr6-rbs

This document is an option for MSR6/RBS that is derived from [I-D.xu-msr6-rbs]. The key changes over that draft are as follows.

A.1.1. RBS-Offset/RBS-Length

In [I-D.xu-msr6-rbs], the RBS-Address was rewritten on every copy to a different adjacency by replacing the RU in the RBS-address with the RU for the adjacency. This required a potentially significant amount of write cycles to packet memory for each copy and changes the size of the packet header on each hop.

This draft proposes to add RBS-Offset and RBS-Length fields and changes the processing of the RBS-address, so that only these two indices need to be re-calculated and re-written on every packet copy, keeping the extension header size the same and minimizing the amount of writes required.

A.1.2. Type-specific data encoding

This draft further reduces the size of the MSR/RBS extension header by encoding the RBS-address not as a TLV, but as the MRH Type-specific data field, thereby saving the TL parts of the TLV option (32 bits). It also replaces the TotalLen field (which did change on

every hop) for the RBS address with an (immutable) 32-bit unit counter called RU0L which saves 2 bits.

A.1.3. IP Multicast compatibility

This draft adds the (optional) MSER-Segment field (IPv6 address), with the primary option being that IPv6 packets with MSR/RBS extension header can support IPv6 multicast without additional IPv6 in IPv6 extension headers or IPv6 in IPv6 encapsulation. Without this MSER-Segment, there is no field to carry the IPv6 Multicast Destination Address required to support IPv6 Multicast.

Support for IPv6 Multicast with MSR/RBS not only enables efficient end-to-end IPv6 multicast with stateless source-routing, but it also allows to use MSR/RBS even when it only encapsulates another IP or IPv6 multicast packet. This is the common case when using MVPN, where the CE multicast packets (IP or IPv6) are IP Multicast encapsulated on the PE (IPv4 or IPv6). Because of the MSER Segment field, all MVPN signaling protocols defined for this so-called SP IP Multicast instance can be reused with MSR6/RBS.

IP Multicast compatibility also should make it easier to support MSR6/RBS in Host stacks via socket APIs. These already support extension headers, but it is a lot more complex to introduce new socket types, which would ve required when MSR6/RBS can not be made to look like either IP Multicast (or IP Unicast) to the Socket API.

A.1.4. Terminology

This document proposes the terms MSR, MSIR and MSER for routers using MSR6 stateless multicast.

A.1.5. Text changes

Large part of the text where rewritten, and pseudocode from [I-D.eckert-bier-cgm2-rbs] was inherited.

A.2. Comparison with RBS for BIER

[I-D.eckert-bier-cgm2-rbs] introduced RBS-Address encoding for BIER without being specific to what encapsulation to use for it. It also describes the overall architectural use of RBS addresses and their scalability benefits.

[I-D.eckert-bier-cgm2-rbs] as an architecture document (wrt. to use of a controller for example) is also applicable to MSR6/RBS, as are the scalability benefits of RBS. For current brevity of this draft, none of that text has been copied here (yet).

[I-D.eckert-bier-cgm2-rbs] stays valid as a valuable protocol option for BIER, especially as an improvement over BIER-TE due to the simplification in architectural complexity (variety of adjacencies to further save bits in the static Bitstring in BIER-TE), and the better scaling of RBS addresses compared to BIER-TE and even BIER Bitstrings in large networks. Scale specifically means the need for fewer packet copies to the same set of BFER (MSER) in large SP networks.

[I-D.eckert-bier-cgm2-rbs] does not currently include the optimization of RBS-Length/RBS-Offset to avoid rewriting/shortening the whole RBS-Address on every copy, but that would be equally an option there.

Authors' Addresses

Toerless Eckert
Futurewei Technologies USA
2220 Central Expressway
Santa Clara, CA 95050
United States of America

Email: tte@cs.fau.de

Xuesong Geng
Huawei 2012 NT Lab
China

Email: gengxuesong@huawei.com

Xiuli Zheng
Huawei 2012 NT Lab
China

Email: zhengxiuli@huawei.com

Rui Meng
Huawei 2012 NT Lab
China

Email: mengrui@huawei.com

Fengkai Li
Huawei 2012 NT Lab

Email: lifengkai@huawei.com