SPRING
Internet-Draft
Intended status: Standards Track
Expires: January 29, 2022

W. Cheng, Ed.
China Mobile
C. Filsfils
Cisco Systems, Inc.
Z. Li
Huawei Technologies
B. Decraene
Orange
D. Cai
Alibaba
D. Voyer
Bell Canada
F. Clad, Ed.
Cisco Systems, Inc.
S. Zadok
Broadcom
J. Guichard
Futurewei Technologies Ltd.
L. Aihua
ZTE Corporation
R. Raszuk
NTT Network Innovations
C. Li
Huawei Technologies
July 28, 2021

Compressed SRv6 Segment List Encoding in SRH

draft-filsfilscheng-spring-srv6-srh-compression-02

Abstract

This document defines a compressed SRv6 Segment List Encoding in the Segment Routing Header (SRH). This solution does not require any SRH data plane change nor any SRv6 control plane change. This solution leverages the SRv6 Network Programming model.

Status of This Memo

Table of Contents

1. Introduction

The Segment Routing architecture is defined in [RFC8402].

SRv6 Network Programming [RFC8986] defines a framework to build a network program with topological and service segments carried in a Segment Routing header (SRH) [RFC8754].

This document adds new flavors to the SR endpoint behaviors defined in Section 4 of [RFC8986]. These flavors enable a compressed encoding of the SRv6 Segment-List in the SRH and therefore address the requirements described in [I-D.srcompdt-spring-compression-requirement].

The flavors defined in this document leverage the SRH data plane without any change and do not require any SRv6 control plane change.

2. Terminology

This document leverages the terms defined in [RFC8402], [RFC8754] and [RFC8986]. The reader is assumed to be familiar with this terminology.

This document introduces the following new terms:

o Compressed-SID (C-SID): A C-SID is a short encoding of a SID in SRv6 packet that does not include the SID block bits (locator block).

o Compressed-SID container (C-SID container): An entry of the SRH Segment-List field (128 bits) that contains a sequence of C-SIDs.

o Compressed-SID sequence (C-SID sequence): A group of one or more C-SID containers in a segment list that share the same SRv6 SID block.

o Uncompressed SID sequence: A group of one or more uncompressed SIDs in a segment list.

o Compressed Segment List encoding: A segment list encoding that reduces the packet header length thanks to one or more C-SID sequences. A compressed Segment List encoding may also contain any number of uncompressed SID sequences.

2.1. Requirements Language

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

3. Basic Concepts

In an SRv6 domain, the SIDs are allocated from a particular IPv6 prefix: the SRv6 SID block. Therefore, all SRv6 SIDs instantiated from the same SRv6 SID block share the same most significant bits. These common bits are named Locator-Block in [RFC8986]. Furthermore, when the combined length of the SRv6 SID Locator, Function and Argument is smaller than 128 bits, the trailing bits are set to zero.

When a sequence of consecutive SIDs in a Segment List shares a common Locator-Block, a compressed SRv6 Segment-List encoding can optimize the packet header length by avoiding the repetition of the Locator-Block and trailing bits with each individual SID.

The compressed Segment List encoding is fully compliant with the specifications in [RFC8402], [RFC8754] and [RFC8986]. Efficient encoding is achieved by combining a compressed Segment List encoding logic on the SR policy headend with new flavors of the base SRv6 endpoint behaviors that decode this compressed encoding. No SRv6 SRH data plane change nor control plane extension is required.

A Segment List can be encoded in the packet header using any combination of compressed and uncompressed sequences. The C-SID sequences leverage the flavors defined in this document, while the uncompressed sequences use behaviors and flavors defined in other documents, such as [RFC8986]. An SR Policy headend constructs and compresses the SID-list depending on the capabilities of each SR endpoint node that the packet should traverse, as well as its own compression capabilities.

It is expected that compressed encoding flavors be available on devices with limited packet manipulation capabilities, such as legacy ASICs.

The compressed Segment List encoding supports any SRv6 SID Block allocation. While other options are supported and may provide higher efficiency, each routing domain can be allocated a /48 prefix from a global IPv6 block (see Section 6.2).

4. SR Endpoint Flavors

This section defines several options to achieve compressed Segment List encoding, in the form of two new flavors for the END, END.X and END.T behaviors of [RFC8986]. These flavors could also be combined with behaviors defined in other documents.

The compressed encoding can be achieved by leveraging any of these SR endpoint flavors. The NEXT-C-SID flavor and the REPLACE-C-SID flavor expose the same high-level behavior in their use of the SID argument to determine the next segment to be processed, but they have different low-level characteristics that can make one more or less efficient than the other for a particular SRv6 deployment. The NEXT-and-REPLACE-C-SID flavor is the combination of the NEXT-C-SID flavor and the REPLACE-C-SID flavor. It provides the best efficiency in terms of encapsulation size at the cost of increased complexity.

It is recommended for ease of operation that a single compressed encoding flavor be used in a given SRv6 domain. However, in a multi-domain deployment, different flavors can be used in different domains.

All three flavors leverage the following variables:

o Variable B is the Locator Block length of the SID.

o Variable NF is the sum of the Locator Node and the Function lengths of the SID. It is also referred to as C-SID length.

o Variable A is the Argument length of the SID.

4.1. NEXT-C-SID Flavor

A SID instantiated with the NEXT-C-SID flavor takes an argument that carries the remaining C-SIDs in the current C-SID container.

The length A of the argument is equal to 128-B-NF and should be a multiple of NF.

```
+------------------------------------------------------------------+
|     Locator-Block      |Loc-Node|           Argument             |
|                        |Function|                                |
+------------------------------------------------------------------+
 <--------- B ----------> <- NF -> <------------- A -------------->
```

Example of a NEXT-C-SID flavored SID structure using a 48-bit block, 16-bit combined locator and function, and 64-bit argument

The NEXT-C-SID flavor has been previously documented in [I-D.filsfils-spring-net-pgm-extension-srv6-usid] under the name "SHIFT" flavor. In that context, a C-SID and a C-SID-sequence are respectively named a Micro-Segment (uSID) and a Micro-Program.

4.1.1. End with NEXT-C-SID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End SID with the NEXT-C-SID flavor, the procedure described in Section 4.1 of [RFC8986] is executed with the following modifications.

The below pseudocode is inserted between lines S01 and S02 of the SRH processing in Section 4.1 of [RFC8986], and a second time before line S01 of the upper-layer header processing in Section 4.1.1 of [RFC8986].

```
S01. If (DA.Argument != 0) {
S02.   If (IPv6 Hop Limit <= 1) {
S03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
S04.   }
S05.   Copy the value of DA.Argument into the bits [B..(B+A-1)]
         of the Destination Address.
S06.   Set the bits [(B+A)..127] of the Destination Address to
         zero.
S07.   Decrement Hop Limit by 1.
S08.   Submit the packet to the egress IPv6 FIB lookup for
         transmission to the next destination.
S09. }
```

Notes:

- o "DA.Argument" identifies the bits "[(B+NF)..127]" in the Destination Address of the IPv6 header.

- o The value in the Segments Left field of the SRH is not modified when "DA.Argument" in the received packet has a non-zero value.

4.1.2. End.X with NEXT-C-SID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.X SID with the NEXT-C-SID flavor, the procedure described in Section 4.2 of [RFC8986] is executed with the same modifications as in Section 4.1.1 of this document, except for line S08 that is replaced as follows.

```
S08.   Submit the packet to the IPv6 module for transmission to the
         new destination via a member of J.
```

4.1.3. Combination with PSP, USP and USD flavors

PSP: The PSP flavor defined in Section 4.16.1 of [RFC8986] is unchanged when combined with the NEXT-C-SID flavor.

USP: The USP flavor defined in Section 4.16.2 of [RFC8986] is unchanged when combined with the NEXT-C-SID flavor.

USD: The USD flavor is unchanged when combined with the NEXT-C-SID flavor. The pseudocodes defined in Section 4.1.1 and Section 4.1.2 of this document are inserted at the beginning of the modified upper-layer header processing defined in Section 4.16.3 of [RFC8986] for End and End.X, respectively.

4.2. REPLACE-C-SID Flavor

A SID instantiated with the REPLACE-C-SID flavor takes an argument that indicates the index of the next C-SID in the appropriate container.

The length A of the argument should be at least ceil(log_2(128/NF)).

All SIDs that are part of a C-SID sequence using the REPLACE-C-SID flavor have the same C-SID length NF.

```
+------------------------------------------------------------------+
|     Locator-Block      |  Locator-Node  |Argument|       0        |
|                        |   + Function   |        |                |
+------------------------------------------------------------------+
 <--------- B ----------> <----- NF -----> <- A -->
```

Example of a REPLACE-C-SID flavored SID structure using a 48-bit block, 32-bit combined locator and function, and 16-bit argument

The REPLACE-C-SID flavor has been previously documented in [I-D.cl-spring-generalized-srv6-for-cmpr] under the name "COC(Continue of Compression)" flavor. In that context, a C-SID and

a C-SID-sequence are respectively named a G-SID and G-SRv6 compression sub-path.

4.2.1. End with REPLACE-C-SID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End SID with the REPLACE-C-SID flavor, the SRH processing described in Section 4.1 of [RFC8986] is replaced as follows.

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0 and DA.Argument == 0) {
S03.      Stop processing the SRH, and proceed to process the next
            header in the packet, whose type is identified by
            the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.      Send an ICMP Time Exceeded message to the Source Address,
            Code 0 (Hop limit exceeded in transit),
            interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If (DA.Argument != 0) {
S10.     If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
S11.        Send an ICMP Parameter Problem to the Source Address,
              Code 0 (Erroneous header field encountered),
              Pointer set to the Segments Left field,
              interrupt packet processing and discard the packet.
S11.     }
S12.     Decrement DA.Argument by 1.
S13.   } Else {
S14.     If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
S15.        Send an ICMP Parameter Problem to the Source Address,
              Code 0 (Erroneous header field encountered),
              Pointer set to the Segments Left field,
              interrupt packet processing and discard the packet.
S11.     }
S12.     Decrement Segments Left by 1.
S13.     Set DA.Argument to (128/NF - 1).
S14.   }
S15.   Decrement IPv6 Hop Limit by 1
S16.   Write Segment List[Segments Left][DA.Argument] into the bits
         [B..B+NF-1] of the Destination Address of the IPv6 header.
S17.   Write DA.Argument into the bits [B+NF..B+NF+A-1] of the
         Destination Address of the IPv6 header.
S18.   Submit the packet to the egress IPv6 FIB lookup for
          transmission to the new destination.
S19. }
```

Notes:

- o "DA.Argument" identifies the bits "[(B+NF)..(B+NF+A-1)]" in the Destination Address of the IPv6 header.

- o "Segment List[Segments Left][DA.Argument]" identifies the bits "[DA.Argument*NF..(DA.Argument+1)*NF-1]" in the SRH Segment List entry at index Segments Left.

The upper-layer header processing described in Section 4.1.1 of [RFC8986] is unchanged.

4.2.2. End.X with REPLACE-C-SID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.X SID with the REPLACE-C-SID flavor, the procedure described in Section 4.2 of [RFC8986] is executed with the same modifications as in Section 4.2.1 of this document, except for line S18 that is replaced as follows.

```
S18.   Submit the packet to the IPv6 module for transmission to the
         new destination via a member of J.
```

4.2.3. Combination with PSP, USP and USD flavors

PSP: When combined with the REPLACE-C-SID flavor, the additional PSP flavor instructions defined in Section 4.16.1.2 of [RFC8986] are inserted after line S17 of the pseudocode in Section 4.2.1, and the first line of the inserted instructions is modified as follows.

```
S17.1.   If (Segments Left == 0 and DA.Argument == 0) {
```

USP: When combined with the REPLACE-C-SID flavor, the lines S02-S04 of the pseudocode in Section 4.2.1 are substituted by the USP flavor instructions defined in Section 4.16.2 of [RFC8986], with the following modification.

```
S02.   If (Segments Left == 0 and DA.Argument == 0) {
```

USD: The USD flavor defined in Section 4.16.3 of [RFC8986] is unchanged when combined with the REPLACE-C-SID flavor.

4.3. Combined NEXT-and-REPLACE-C-SID Flavor

A SID instantiated with the NEXT-and-REPLACE-C-SID flavor takes a two-parts argument comprising, Arg.Next and Arg.Index, and encoded in the SID in this order.

The length A_I of Arg.Index should be at least ceil(log_2(128/NF)).

The length A_N of Arg.Next is equal to 128-B-NF-A_I and must be a multiple of NF.

The total SID argument length A is the sum of A_I and A_N.

The NEXT-and-REPLACE-C-SID flavor also leverages an additional variable, C_DA, that is equal to (1 + (A_N/NF)) and represents the number of C-SID's that can be encoded in the IPv6 Destination Address.

All SIDs that are part of a C-SID sequence using the NEXT-and-REPLACE-C-SID flavor must have the same C-SID length NF. Furthermore, this NF must be a divisor of 128.

```
+--------------------------------------------------------------------+
|     Locator-Block      |Loc-Node|        Arg.Next        |  Arg.  |
|                        |Function|                        | Index  |
+--------------------------------------------------------------------+
 <--------- B ----------> <- NF -> <-------- A_N ---------> <- A_I ->
```

Example of a NEXT-and-REPLACE-C-SID flavored SID structure using a 48-bit block, 16-bit combined locator and function, 48-bit Arg.Next and 16-bit Arg.Index

Pseudo-code:

```
 1.   If (DA.Arg.Next != 0) {
 2.     Copy DA.Arg.Next into the bits [B..(B+A_N-1)] of the
          Destination Address of the IPv6 header.
 3.     Set the bits [(B+A_N)..(B+NF+A_N-1)] of the Destination Address
          of the IPv6 header to zero.
 4.   } Else If (DA.Arg.Index >= C_DA) {
 5.     Decrement DA.Arg.Index by C_DA.
 6.     Copy C_DA*NF bits from Segment List[Segments Left][DA.Arg.Index]
          into the bits [B..B+C_DA*NF-1] of the Destination Address of
          the IPv6 header.
 7.   } Else If (Segments Left != 0) {
 8.     Decrement Segments Left by 1.
 9.     Set DA.Arg.Index to ((DA.Arg.Index - C_DA) % (128/NF)).
10.     Copy C_DA*NF bits from Segment List[Segments Left][DA.Arg.Index]
          into the bits [B..B+C_DA*NF-1] of the Destination Address of
          the IPv6 header.
11.   } Else {
12.     Copy DA.Arg.Index*NF bits from Segment List[0][0] into the bits
          [B..B+DA.Arg.Index*NF-1] of the Destination Address of the
          IPv6 header.
13.     Set the bits [B+DA.Arg.Index*NF..B+NF+A_N-1] of the Destination
          Address of the IPv6 header to zero.
14.     Set DA.Arg.Index to 0.
15.   }
```

Notes:

- "DA.Arg.Next" identifies the bits "[(B+NF)..(B+NF+A_N-1)]" in the Destination Address of the IPv6 header.

- "DA.Arg.Index" identifies the bits "[(B+NF+A_N)..(B+NF+A_N+A_I-1)]" in the Destination Address of the IPv6 header.

- "Segment List[Segments Left][DA.Arg.Index]" identifies the bits "[DA.Arg.Index*NF..(DA.Arg.Index+1)*NF-1]" in the SRH Segment List entry at index Segments Left.

5. GIB, LIB, global C-SID and local C-SID

GIB: The set of IDs available for global C-SID allocation.

LIB: The set of IDs available for local C-SID allocation.

5.1. Global C-SID

A C-SID from the GIB.

A Global C-SID typically identifies a shortest-path to a node in the SRv6 domain. An IP route is advertised by the parent node to each of its global C-SID's, under the associated C-SID block. The parent node executes a variant of the END behavior.

A node can have multiple global C-SID's under the same C-SID blocks (e.g. one per IGP flexible algorithm). Multiple nodes may share the same global C-SID (anycast).

5.2. Local C-SID

A C-SID from the LIB.

A local C-SID may identify a cross-connect to a direct neighbor over a specific interface or a VPN context.

No IP route is advertised by a parent node for its local C-SID's.

If N1 and N2 are two different physical nodes of the SRv6 domain and I is a local C-SID value, then N1 and N2 may bind two different behaviors to I.

The concept of LIB is applicable to SRv6 and specifically to its NEXT-C-SID and REPLACE-C-SID flavors. The shorter the SID/C-SID, the more benefit the LIB brings.

The allocation of C-SID's from the GIB and LIB depends on the C-SID length (see Section 6.3).

6. C-SID and Block Length

6.1. C-SID Length

The NEXT-C-SID flavor supports both 16- and 32-bit C-SID lengths. A C-SID length of 16-bit is recommended.

The REPLACE-C-SID flavor supports both 16- and 32-bit C-SID lengths. A C-SID length of 32-bit is recommended.

6.2. Block Length

The recommended SRv6 SID block sizes for the NEXT-C-SID flavor are 16, 32 or 48 bits. The smaller the block, the higher the compression efficiency.

The recommended SRv6 SID block size for the REPLACE-C-SID flavor can be 48, 56, 64, 72 or 80 bits, depending on the needs of the operator.

6.3. GIB/LIB Usage

The previous block and C-SID length recommendations, call for the following GIB/LIB usage:

- o NEXT-C-SID:

  * GIB: END.NEXT-C-SID

  * LIB: END.X.NEXT-C-SID, END.DX.NEXT-C-SID, END.DT.NEXT-C-SID

  * LIB: END.DX.NEXT-C-SID for large-scale PW support

- o REPLACE-C-SID:

  * GIB: END.REPLACE-C-SID, END.X.REPLACE-C-SID, END.DX.REPLACE-C-SID, END.DT.REPLACE-C-SID

  * LIB: END.DX.REPLACE-C-SID for large-scale PW support

7. Efficient SID-list Encoding

The compressed SID-list encoding logic is a local behavior of the SR Policy headend node and hence out of the scope of this document.

8. Inter Routing Domains with the End.XPS behavior

The End.XPS behavior described in this section is OPTIONAL.

Some SRv6 traffic may need to cross multiple routing domains, such as different Autonomous Systems (ASes) or different routing areas. Different routing domains may use different addressing schema and SRv6 SID blocks.

This section defines an optional solution and SID behavior allowing for the use of different SRv6 SID blocks between routing domains.

The solution requires a new SID behavior, called "Endpoint with cross-connect to an array of layer-3 adjacencies and SRv6 Prefix Swap" (End.XPS for short) allowing for this transition of SRv6 SID block between two routing domains.

End.XPS is a variant of End.X, performing both "End.X Layer-3 Cross-Connect" and the translation of the SRv6 SID block between the two routing domains.

The processing takes as an additional parameter the prefix B2/m corresponding the SRv6 SID block in the second domain. This

parameter is a property of the (received) SID and is given as a result of the lookup on the IPv6 destination address which identifies the SRv6 SID and its properties.

The End.XPS behavior is compatible with the NEXT-C-SID, REPLACE-C-SID, and NEXT-and-REPLACE-C-SID flavors described in this document.

When a router R receives a packet whose IPv6 DA matches a local End.XPS SID with the NEXT-C-SID flavor, that is associated with a set J of one or more Layer-3 adjacencies and the SRv6 SID block B2/m of the neighbor routing domain, R processes the packet as follows.

```
 1.   If (DA.Argument != 0) {
 2.     Write B2 into the most significant bits of the Destination
          Address of the IPv6 header.
 3.     Write DA.Argument into the bits [m..(m+A-1)] of the
          Destination Address of the IPv6 header.
 4.     Set the bits [(m+A)..127] of the Destination Address
          of the IPv6 header to zero.
 5.   } Else {
 6.     Decrement Segments Left by 1.
 7.     Copy Segment List[Segments Left] from the SRH to the
          Destination Address of the IPv6 header.
 8.   }
 9.   Submit the packet to the IPv6 module for transmission to the
        new destination via a member of J.
```

When a router R receives a packet whose IPv6 DA matches a local End.XPS SID with the REPLACE-C-SID flavor, that is associated with a set J of one or more Layer-3 adjacencies and the SRv6 SID block B2/m of the neighbor routing domain, R processes the packet as follows.

```
 1.   If (DA.Argument != 0) {
 2.     Decrement DA.Argument by 1.
 3.   } Else {
 4.     Decrement Segments Left by 1.
 5.     Set DA.Argument to (128/NF - 1).
 6.   }
 7.   Write B2 into the most significant bits of the Destination
        Address of the IPv6 header.
 8.   Write Segment List[Segments Left][DA.Argument] into the bits
        [m..m+NF-1] of the Destination Address of the IPv6 header.
 9.   Write DA.Argument into the bits [m+NF..m+NF+A-1] of the
        Destination Address of the IPv6 header.
10.   Set the bits [(m+NF+A)..127] of the Destination Address
        of the IPv6 header to zero.
11.   Submit the packet to the IPv6 module for transmission to the
        new destination via a member of J.
```

Note: the way the SRv6 SID Block B2 of the next routing domain is known is out of scope of this document. As examples, it could be learnt via configuration, or using a signaling protocol either with the peer domain or with a central controller (e.g. PCE).

When End.XPS SID behavior is used, the restriction on the C-SID length for the REPLACE-C-SID and the NEXT-and-REPLACE-C-SID flavors is relaxed and becomes: all SID the are part of a C-SID sequence *within a domain* MUST have the same SID length NF.

9. Control Plane

This document does not require any control plane modification.

10. Illustrations

Illustrations will be provided in a separate document.

11. Interoperability Status

In November 2020, China Mobile successfully validated multiple interoperable implementations of the NEXT-C-SID and REPLACE-C-SID flavors defined in this document.

This testing covered two different implementations of the SRv6 endpoint flavors defined in this document:

- Hardware implementation in Cisco ASR 9000 running IOS XR

- Software implementation in Cisco IOS XRv9000 virtual appliance

- Hardware implementation in Huawei NE40E and NE5000E running VRP

The interoperability was validated for the following scenario:

- Packet forwarding through a traffic engineering segment list combining, in the same SRH ([RFC8754]), SRv6 SIDs bound to an endpoint behavior with the NEXT-C-SID flavor and SRv6 SIDs bound to an endpoint behavior with the REPLACE-C-SID flavor.

Further interoperability testing is ongoing and will be reported in this document as the work progresses.

12. Security Considerations

The security requirements and mechanisms described in [RFC8402] and [RFC8754] also apply to this document.

This document does not introduce any new security consideration.

13. Acknowledgements

The authors would like to thank Kamran Raza, Xing Jiang, YuanChao Su, Han Li and Yisong Liu.

14. References

14.1. 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>.

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

[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>.

[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>.

[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>.

14.2. Informative References

[I-D.cl-spring-generalized-srv6-for-cmpr]
Cheng, W., Li, Z., Li, C., Clad, F., Liu, A., Xie, C., Liu, Y., and S. Zadok, "Generalized SRv6 Network Programming for SRv6 Compression", draft-cl-spring-generalized-srv6-for-cmpr-03 (work in progress), April 2021.

[I-D.filsfils-spring-net-pgm-extension-srv6-usid]
Filsfils, C., Garvia, P. C., Cai, D., Voyer, D., Meilik, I., Patel, K., Henderickx, W., Jonnalagadda, P., Melman, D., Liu, Y., and J. Guichard, "Network Programming extension: SRv6 uSID instruction", draft-filsfils-spring-net-pgm-extension-srv6-usid-10 (work in progress), March 2021.

[I-D.srcompdt-spring-compression-requirement]
Cheng, W., Xie, C., Bonica, R., Dukes, D., Li, C., Shaofu, P., and W. Henderickx, "Compressed SRv6 SID List Requirements", draft-srcompdt-spring-compression-requirement-06 (work in progress), March 2021.

Authors' Addresses

Weiqiang Cheng (editor)
China Mobile
China

Email: chengweiqiang@chinamobile.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Dennis Cai
Alibaba
USA

Email: d.cai@alibaba-inc.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Francois Clad (editor)
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Shay Zadok
Broadcom
Israel

Email: shay.zadok@broadcom.com

James N Guichard
Futurewei Technologies Ltd.
USA

Email: james.n.guichard@futurewei.com

Liu Aihua
ZTE Corporation
China

Email: liu.aihua@zte.com.cn

Robert Raszuk
NTT Network Innovations
USA

Email: robert@raszuk.net

Cheng Li
Huawei Technologies
China

Email: chengli13@huawei.com