

Softwire WG
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2015

Q. Wang
China Telecom
W. Meng
C. Wang
ZTE Corporation
M. Boucadair
France Telecom
June 3, 2015

RADIUS Extensions for IPv4-Embedded Multicast and Unicast IPv6 Prefixes
draft-hu-softwire-multicast-radius-ext-08

Abstract

This document specifies a new Remote Authentication Dial-In User Service (RADIUS) attribute to carry the Multicast-Prefixes-64 information, aiming to delivery the Multicast and Unicast IPv6 Prefixes to be used to build multicast and unicast IPv4-Embedded IPv6 addresses. this RADIUS attribute is defined based on the equivalent DHCPv6 OPTION_v6_PREFIX64 option.

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 December 5, 2015.

Copyright Notice

Copyright (c) 2015 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

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

Table of Contents

1. Introduction	3
2. Convention and Terminology	4
3. Multicast-Prefixes-64 Configuration with RADIUS and DHCPv6 . . .	5
4. RADIUS Attribute	8
4.1. Multicast-Prefixes-64	8
5. Table of Attributes	11
6. Security Considerations	12
7. IANA Considerations	13
8. Acknowledgments	14
9. Normative References	15
Authors' Addresses	16

1. Introduction

The solution specified in [I-D.ietf-softwire-dslite-multicast] relies on stateless functions to graft part of the IPv6 multicast distribution tree and IPv4 multicast distribution tree, also uses IPv4-in-IPv6 encapsulation scheme to deliver IPv4 multicast traffic over an IPv6 multicast-enabled network to IPv4 receivers.

To inform the mB4 element of the PREFIX64, a PREFIX64 option may be used. [I-D.ietf-softwire-multicast-prefix-option] defines a DHCPv6 PREFIX64 option to convey the IPv6 prefixes to be used for constructing IPv4-embedded IPv6 addresses.

In broadband environments, a customer profile may be managed by Authentication, Authorization, and Accounting (AAA) servers, together with AAA for users. The Remote Authentication Dial-In User Service (RADIUS) protocol [RFC2865] is usually used by AAA servers to communicate with network elements. Since the Multicast-Prefixes-64 information can be stored in AAA servers and the client configuration is mainly provided through DHCP running between the NAS and the requesting clients, a new RADIUS attribute is needed to send Multicast-Prefixes-64 information from the AAA server to the NAS.

This document defines a new RADIUS attribute to be used for carrying the Multicast-Prefixes-64, based on the equivalent DHCPv6 option already specified in [I-D.ietf-softwire-multicast-prefix-option].

This document makes use of the same terminology defined in [I-D.ietf-softwire-dslite-multicast].

This attribute can be in particular used in the context of DS-Lite Multicast, MAP-E Multicast and other IPv4-IPv6 Multicast techniques. However it is not limited to DS-Lite Multicast.

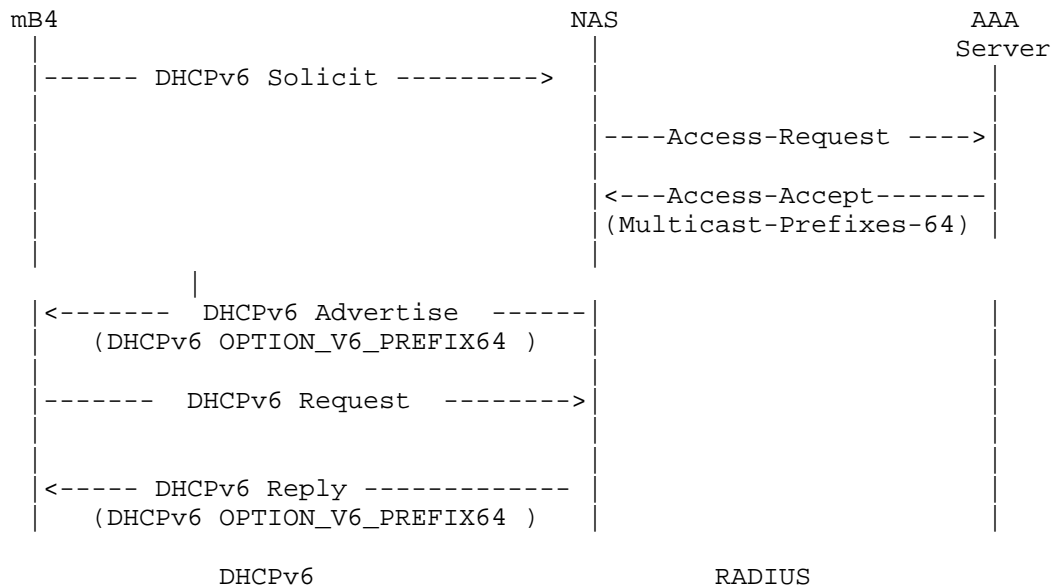
DS-Lite unicast RADIUS extensions are defined in [RFC6519] .

2. Convention and Terminology

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

The terms DS-Lite multicast Basic Bridging BroadBand element (mB4) and the DS-Lite multicast Address Family Transition Router element (mAFTR) are defined in [I-D.ietf-softwire-dslite-multicast]

Figure 1 illustrates in DS-Lite scenario how the RADIUS protocol and DHCPv6 work together to accomplish Multicast-Prefixes-64 configuration on the mB4 element for multicast service when an IP session is used to provide connectivity to the user.



The NAS operates as a client of RADIUS and as a DHCP Server/Relay for mB4. When the mB4 sends a DHCPv6 Solicit message to NAS(DHCP Server/Relay). The NAS sends a RADIUS Access-Request message to the RADIUS server, requesting authentication. Once the RADIUS server receives the request, it validates the sending client, and if the request is approved, the AAA server replies with an Access-Accept message including a list of attribute-value pairs that describe the parameters to be used for this session. This list MAY contain the Multicast-Prefixes-64 attribute (asm-length,ASM_PREFIX64,ssm-length,SSM_PREFIX64,unicast-length,U_PREFIX64). Then, when the NAS receives the DHCPv6 Request message containing the OPTION_V6_PREFIX64 option in its Option Request option,the NAS SHALL use the prefixes returned in the RADIUS Multicast-Prefixes-64 attribute to populate the DHCPv6 OPTION V6 PREFIX64 option in the DHCPv6 reply message.

NAS MAY be configured to return the configured Multicast-Prefixes-64 by the AAA Server to any requesting client without relaying each received request to the AAA Server.

Figure 2 describes another scenario, which accomplish DS-Lite Multicast-Prefixes-64 configuration on the mB4 element for multicast service when a PPP session is used to provide connectivity to the user. Once the NAS obtains the Multicast-Prefixes-64 attribute from the AAA server through the RADIUS protocol, the NAS MUST store the received Multicast-Prefixes-64 locally. When a user is online and sends a DHCPv6 Request message containing the OPTION_V6_PREFIX64 option in its Option Request option, the NAS retrieves the previously stored Multicast-Prefixes-64 and uses it as OPTION_V6_PREFIX64 option in DHCPv6 Reply message.

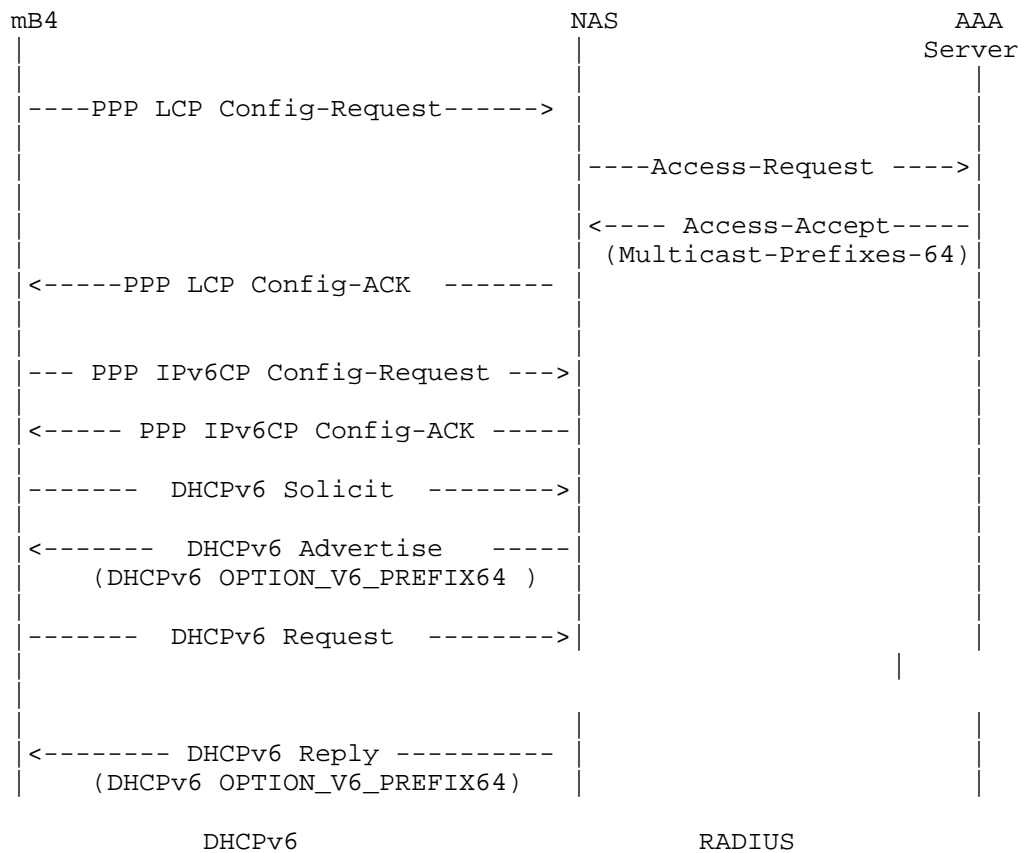


Figure 2: RADIUS and DHCPv6 Message Flow for a PPP Session

According to [RFC3315], after receiving the Multicast-Prefixes-64 attribute in the initial Access-Accept packet, the NAS MUST store the received V6_PREFIX64 locally. When the mB4 sends a DHCPv6 Renew message to request an extension of the lifetimes for the assigned address or prefix, the NAS does not have to initiate a new Access-

Request packet towards the AAA server to request the Multicast-Prefixes-64. The NAS retrieves the previously stored Multicast-Prefixes-64 and uses it in its reply.

Also, if the DHCPv6 server to which the DHCPv6 Renew message was sent at time T1 has not responded, the DHCPv6 client initiates a Rebind/Reply message exchange with any available server. In this scenario, the NAS receiving the DHCPv6 Rebind message MUST initiate a new Access-Request message towards the AAA server. The NAS MAY include the Multicast-Prefixes-64 attribute in its Access-Request message.

4. RADIUS Attribute

This section specifies the format of the new RADIUS attribute.

4.1. Multicast-Prefixes-64

The Multicast-Prefixes-64 attribute conveys the IPv6 prefixes to be used in [I-D.ietf-softwire-dslite-multicast] to synthesize IPv4-embedded IPv6 addresses. The NAS SHALL use the IPv6 prefixes returned in the RADIUS Multicast-Prefixes-64 attribute to populate the DHCPv6 PREFIX64 Option [I-D.ietf-softwire-multicast-prefix-option] .

This attribute MAY be used in Access-Request packets as a hint to the RADIUS server, for example, if the NAS is pre-configured with Multicast-Prefixes-64, these prefixes MAY be inserted in the attribute. The RADIUS server MAY ignore the hint sent by the NAS, and it MAY assign a different Multicast-Prefixes-64 attribute.

If the NAS includes the Multicast-Prefixes-64 attribute, but the AAA server does not recognize this attribute, this attribute MUST be ignored by the AAA server.

NAS MAY be configured with both ASM_PREFIX64 and SSM_PREFIX64 or only one of them. Concretely, AAA server MAY return ASM_PREFIX64 or SSM_PREFIX64 based on the user profile and service policies. AAA MAY return both ASM_PREFIX64 and SSM_PREFIX64. When SSM_PREFIX64 is returned by the AAA server, U_PREFIX64 MUST also be returned by the AAA server.

If the NAS does not receive the Multicast-Prefixes-64 attribute in the Access-Accept message, it MAY fall back to a pre-configured default Multicast-Prefixes-64, if any. If the NAS does not have any pre-configured, the delivery of multicast traffic is not supported.

If the NAS is pre-provisioned with a default Multicast-Prefixes-64 and the Multicast-Prefixes-64 received in the Access-Accept message are different from the configured default, then the Multicast-Prefixes-64 attribute received in the Access-Accept message MUST be used for the session.

A summary of the Multicast-Prefixes-64 RADIUS attribute format is shown Figure 3. The fields are transmitted from left to right.

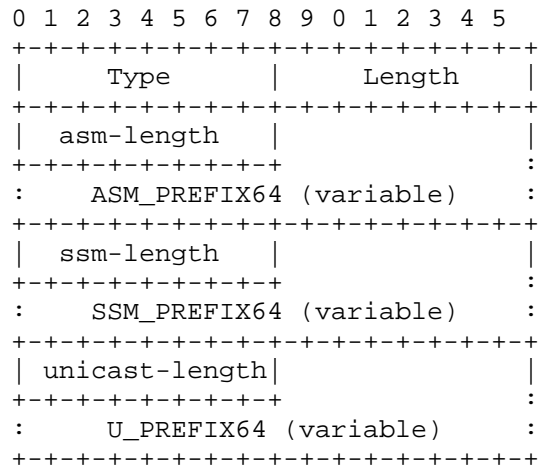


Figure 3: RADIUS attribute format for Multicast-Prefixes-64

Type:

145 for Multicast-Prefixes-64

Length:

This field indicates the total length in octets of this attribute including the Type and Length fields, and the length in octets of all PREFIX fields.

asm-length:

the prefix-length for the ASM IPv4-embedded prefix, as an 8-bit unsigned integer (0 to 128). This field represents the number of valid leading bits in the prefix.

ASM_PREFIX64:

this field identifies the IPv6 multicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast groups in the ASM mode. It is a variable size field with the length of the field defined by the asm-length field and is rounded up to the nearest octet boundary. In such case any additional padding bits must be zeroed. The conveyed multicast IPv6 prefix MUST belong to the ASM range. This prefix is likely to be a /96.

ssm-length:

the prefix-length for the SSM IPv4-embedded prefix, as an 8-bit unsigned integer (0 to 128). This field represents the number of valid leading bits in the prefix.

SSM_PREFIX64:

this field identifies the IPv6 multicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast groups in the SSM mode. It is a variable size field with the length of the field defined by the ssm-length field and is rounded up to the nearest octet boundary. In such case any additional padding bits must be zeroed. The conveyed multicast IPv6 prefix MUST belong to the SSM range. This prefix is likely to be a /96.

unicast-length:

the prefix-length for the IPv6 unicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast sources, as an 8-bit unsigned integer (0 to 128). This field represents the number of valid leading bits in the prefix.

U_PREFIX64:

this field identifies the IPv6 unicast prefix to be used in SSM mode for constructing the IPv4-embedded IPv6 addresses representing the IPv4 multicast sources in the IPv6 domain. U_PREFIX64 may also be used to extract the IPv4 address from the received multicast data flows. It is a variable size field with the length of the field defined by the unicast-length field and is rounded up to the nearest octet boundary. In such case any additional padding bits must be zeroed. The address mapping MUST follow the guidelines documented in [RFC6052].

5. Table of Attributes

The following tables provide a guide to which attributes may be found in which kinds of packets, and in what quantity.

The following table defines the meaning of the above table entries.

Access-Request	Access-Accept	Access-Reject	Challenge	Accounting-Request	#	Attribute
0-1	0-1	0	0	0-1	145	Multicast-Prefixes-64

CoA-Request	CoA-ACK	CoA-NACK	#	Attribute
0-1	0	0	145	Multicast-Prefixes-64

0 This attribute MUST NOT be present in the packet.

0+ Zero or more instances of this attribute MAY be present in the packet.

0-1 Zero or one instances of this attribute MAY be present in the packet.

1 Exactly one instances of this attribute MAY be present in the packet.

6. Security Considerations

This document has no additional security considerations beyond those already identified in [RFC2865] for the RADIUS protocol and in [RFC5176] for CoA messages.

The security considerations documented in [RFC3315] and [RFC6052] are to be considered.

7. IANA Considerations

Per this document, IANA has allocated a new RADIUS attribute type from the IANA registry "Radius Attribute Types" located at <http://www.iana.org/assignments/radius-types>.

Multicast-Prefixes-64 - 145

8. Acknowledgments

The authors would like to thank Ian Farrer, Chongfen Xie, Qi Sun, Linhui Sun and Hao Wang for their contributions to this work.

9. Normative References

- [I-D.ietf-softwire-dslite-multicast]
Qin, J., Boucadair, M., Jacquenet, C., Lee, Y., and Q. Wang, "Delivery of IPv4 Multicast Services to IPv4 Clients over an IPv6 Multicast Network", draft-ietf-softwire-dslite-multicast-09 (work in progress), March 2015.
- [I-D.ietf-softwire-multicast-prefix-option]
Boucadair, M., Qin, J., Tsou, T., and X. Deng, "DHCPv6 Option for IPv4-Embedded Multicast and Unicast IPv6 Prefixes", draft-ietf-softwire-multicast-prefix-option-08 (work in progress), March 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.
- [RFC6519] Maglione, R. and A. Durand, "RADIUS Extensions for Dual-Stack Lite", RFC 6519, February 2012.

Authors' Addresses

Qian Wang
China Telecom
No.118, Xizhimennei
Beijing 100035
China

Email: wangqian@ctbri.com.cn

Wei Meng
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: meng.wei2@zte.com.cn, vally.meng@gmail.com

Cui Wang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: wang.cuil@zte.com.cn

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange.com

Softwire WG
Internet-Draft
Intended status: Standards Track
Expires: December 10, 2019

M. Xu
Y. Cui
J. Wu
Tsinghua University
S. Yang
Shenzhen University
C. Metz
Cisco Systems
June 8, 2019

IPv4 Multicast over an IPv6 Multicast in Softwire Mesh Network
draft-ietf-softwire-mesh-multicast-25

Abstract

During the transition to IPv6, there will be scenarios where a backbone network internally running one IP address family (referred to as the internal IP or I-IP family), connects client networks running another IP address family (referred to as the external IP or E-IP family). In such cases, the I-IP backbone needs to offer both unicast and multicast transit services to the client E-IP networks.

This document describes a mechanism for supporting multicast across backbone networks where the I-IP and E-IP protocol families differ. The document focuses on IPv4-over-IPv6 scenario, due to lack of real-world use cases for IPv6-over-IPv4 scenario.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 10, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	5
3. Terminology	5
4. Scope	6
5. Mesh Multicast Mechanism	7
5.1. Mechanism Overview	8
5.2. Group Address Mapping	8
5.3. Source Address Mapping	9
5.4. Routing Mechanism	9
6. Control Plane Functions of AFBR	10
6.1. E-IP (*,G) and (S,G) State Maintenance	10
6.2. I-IP (S',G') State Maintenance	10
6.3. E-IP (S,G,rpt) State Maintenance	11
6.4. Inter-AFBR Signaling	11
6.5. SPT Switchover	13
6.6. Other PIM Message Types	13
6.7. Other PIM States Maintenance	13
7. Data Plane Functions of the AFBR	14
7.1. Process and Forward Multicast Data	14
7.2. TTL or Hop Count	14
7.3. Fragmentation	14
8. Packet Format and Translation	14
9. Softwire Mesh Multicast Encapsulation	15
10. Security Considerations	16
11. IANA Considerations	16
12. Normative References	16
Appendix A. Acknowledgements	18
Authors' Addresses	18

1. Introduction

During the transition to IPv6, there will be scenarios where a backbone network internally running one IP address family (referred to as the internal IP or I-IP family), connects client networks running another IP address family (referred to as the external IP or E-IP family).

One solution is to leverage the multicast functions inherent in the I-IP backbone to efficiently forward client E-IP multicast packets inside an I-IP core tree. The I-IP tree is rooted at one or more ingress Address Family Border Routers (AFBRs) [RFC5565] and branches out to one or more egress AFBRs.

[RFC4925] outlines the requirements for the softwire mesh scenario and includes support for multicast traffic. It is likely that client E-IP multicast sources and receivers will reside in different client E-IP networks connected to an I-IP backbone network. This requires the client E-IP source-rooted or shared tree to traverse the I-IP backbone network.

This could be accomplished by re-using the multicast VPN approach outlined in [RFC6513]. MVPN-like schemes can support the softwire mesh scenario and achieve a "many-to-one" mapping between the E-IP client multicast trees and the transit core multicast trees. The advantage of this approach is that the number of trees in the I-IP backbone network scales less than linearly with the number of E-IP client trees. Corporate enterprise networks, and by extension multicast VPNs, have been known to run applications that create too many (S,G) states, which is source specific states related with a specified multicast group [RFC7761][RFC7899]. Aggregation at the edge contains the (S,G) states for customer's VPNs and these need to be maintained by the network operator. The disadvantage of this approach is the possibility of inefficient bandwidth and resource utilization when multicast packets are delivered to a receiving AFBR with no attached E-IP receivers.

[RFC8114] provides a solution for delivering IPv4 multicast services over an IPv6 network. But it mainly focuses on the DS-lite [RFC6333] scenario, where IPv4 addresses assigned by a broadband service provider are shared among customers. This document describes a detailed solution for the IPv4-over-IPv6 softwire mesh scenario, where client networks run IPv4 and the backbone network runs IPv6.

Internet-style multicast is somewhat different to the [RFC8114] scenario in that the trees are source-rooted and relatively sparse. The need for multicast aggregation at the edge (where many customer multicast trees are mapped into one or more backbone multicast trees)

does not exist and to date has not been identified. Thus the need for alignment between the E-IP and I-IP multicast mechanisms emerges.

[RFC5565] describes the "Softwire Mesh Framework". This document provides a more detailed description of how one-to-one mapping schemes ([RFC5565], Section 11.1) for IPv4-over-IPv6 multicast can be achieved.

Figure 1 shows an example of how a softwire mesh network can support multicast traffic. A multicast source S is located in one E-IP client network, while candidate E-IP group receivers are located in the same or different E-IP client networks that all share a common I-IP transit network. When E-IP sources and receivers are not local to each other, they can only communicate with each other through the I-IP core. There may be several E-IP sources for a single multicast group residing in different client E-IP networks. In the case of shared trees, the E-IP sources, receivers and rendezvous points (RPs) might be located in different client E-IP networks. In the simplest case, a single operator manages the resources of the I-IP core, although the inter-operator case is also possible and so not precluded.

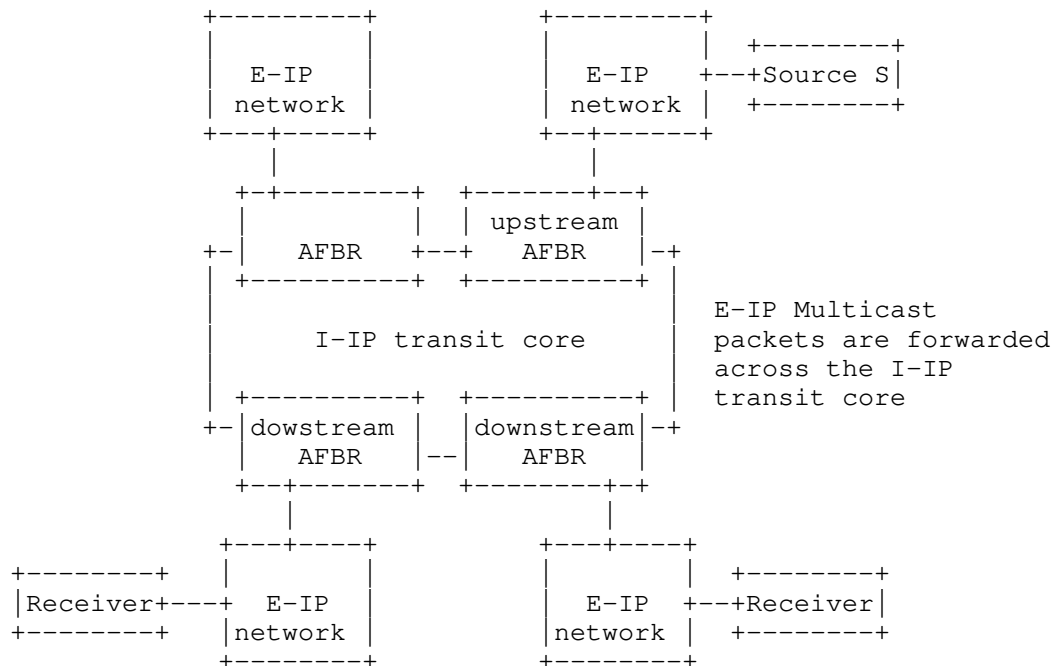


Figure 1: Software Mesh Multicast Framework

2. 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. Terminology

Terminology used in this document:

- o Address Family Border Router (AFBR) - A router interconnecting two or more networks using different IP address families. Additionally, in the context of software mesh multicast, the AFBR runs E-IP and I-IP control planes to maintain E-IP and I-IP multicast states respectively and performs the appropriate encapsulation/decapsulation of client E-IP multicast packets for transport across the I-IP core. An AFBR will act as a source and/or receiver in an I-IP multicast tree.

- o Upstream AFBR: An AFBR that is closer to the source of a multicast data flow.
- o Downstream AFBR: An AFBR that is closer to a receiver of a multicast data flow.
- o I-IP (Internal IP): This refers to IP address family that is supported by the core network. In this document, the I-IP is IPv6.
- o E-IP (External IP): This refers to the IP address family that is supported by the client network(s) attached to the I-IP transit core. In this document, the E-IP is IPv4.
- o I-IP core tree: A distribution tree rooted at one or more AFBR source nodes and branched out to one or more AFBR leaf nodes. An I-IP core tree is built using standard IP or MPLS multicast signaling protocols (in this document, we focus on IP multicast) operating exclusively inside the I-IP core network. An I-IP core tree is used to forward E-IP multicast packets belonging to E-IP trees across the I-IP core. Another name for an I-IP core tree is multicast or multipoint softwire.
- o E-IP client tree: A distribution tree rooted at one or more hosts or routers located inside a client E-IP network and branched out to one or more leaf nodes located in the same or different client E-IP networks.
- o uPrefix64: The /96 unicast IPv6 prefix for constructing an IPv4-embedded IPv6 unicast address [RFC8114].
- o mPrefix64: The /96 multicast IPv6 prefix for constructing an IPv4-embedded IPv6 multicast address [RFC8114].
- o PIMv4, PIMv6: refer to [RFC8114].
- o Inter-AFBR signaling: A mechanism used by downstream AFBRs to send PIMv6 messages to the upstream AFBR.

4. Scope

This document focuses on the IPv4-over-IPv6 scenario, as shown in the following diagram:

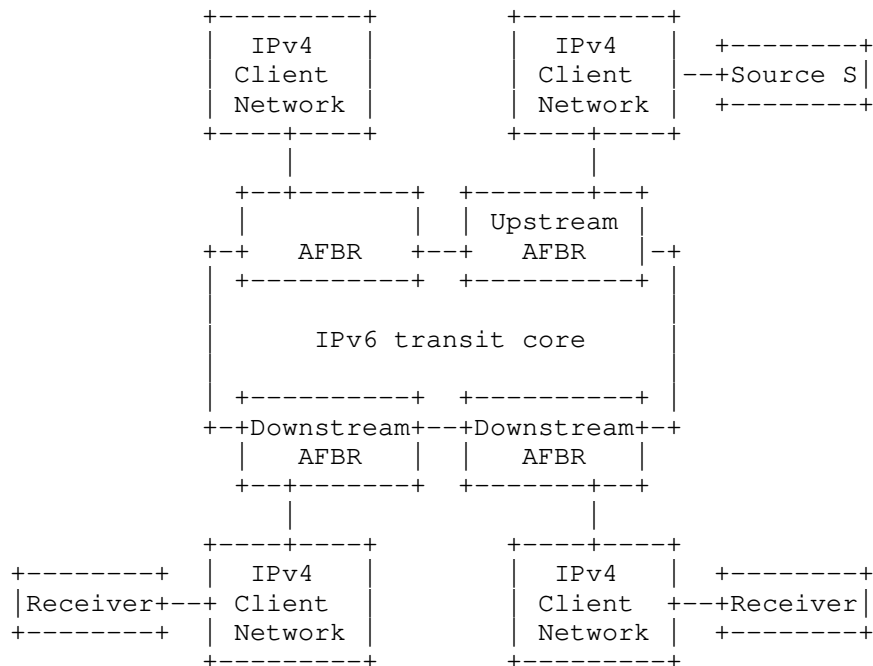


Figure 2: IPv4-over-IPv6 Scenario

In Figure 2, the E-IP client networks run IPv4 and the I-IP core runs IPv6.

Because of the much larger IPv6 group address space, the client E-IP tree can be mapped to a specific I-IP core tree. This simplifies operations on the AFBR because it becomes possible to algorithmically map an IPv4 group/source address to an IPv6 group/source address and vice-versa.

The IPv4-over-IPv6 scenario is an emerging requirement as network operators build out native IPv6 backbone networks. These networks support native IPv6 services and applications but in many cases, support for legacy IPv4 unicast and multicast services will also need to be accommodated.

5. Mesh Multicast Mechanism

5.1. Mechanism Overview

Routers in the client E-IP networks have routes to all other client E-IP networks. Through PIMv4 messages, E-IP hosts and routers have discovered or learnt of (S,G) or (*,G) [RFC7761] IPv4 addresses. Any I-IP multicast state instantiated in the core is referred to as (S',G') or (*,G') and is separated from E-IP multicast state.

Suppose a downstream AFBR receives an E-IP PIM Join/Prune message from the E-IP network for either an (S,G) tree or a (*,G) tree. The AFBR translates the PIMv4 message into an PIMv6 message with the latter being directed towards the I-IP IPv6 address of the upstream AFBR. When the PIMv6 message arrives at the upstream AFBR, it is translated back into an PIMv4 message. The result of these actions is the construction of E-IP trees and a corresponding I-IP tree in the I-IP network. An example of the packet format and translation is provided in Section 8.

In this case, it is incumbent upon the AFBRs to perform PIM message conversions in the control plane and IP group address conversions or mappings in the data plane. The AFBRs perform an algorithmic, one-to-one mapping of IPv4-to-IPv6.

5.2. Group Address Mapping

A simple algorithmic mapping between IPv4 multicast group addresses and IPv6 group addresses is performed. Figure 3 is provided as a reminder of the format:

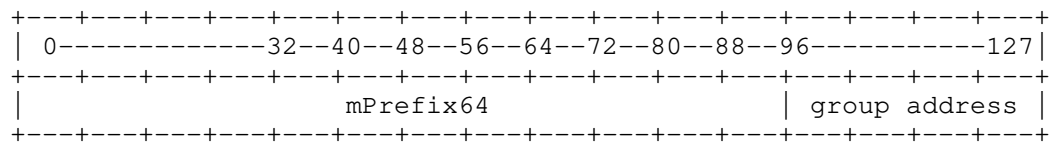


Figure 3: IPv4-Embedded IPv6 Multicast Address Format

An IPv6 multicast prefix (mPrefix64) is provisioned on each AFBR. AFBRs will prepend the prefix to an IPv4 multicast group address when translating it to an IPv6 multicast group address.

The construction of the mPrefix64 for Source-Specific Multicast (SSM) is the same as the construction of the mPrefix64 described in Section 5 of [RFC8114].

With this scheme, each IPv4 multicast address can be mapped into an IPv6 multicast address (with the assigned prefix), and each IPv6 multicast address with the assigned prefix can be mapped into an IPv4 multicast address. The group address translation algorithm can be referred in Section 5.2 of [RFC8114].

5.3. Source Address Mapping

There are two kinds of multicast: Any-Source Multicast (ASM) and SSM. Considering that the I-IP network and E-IP network may support different kinds of multicast, the source address translation rules needed to support all possible scenarios may become very complex. But since SSM can be implemented with a strict subset of the PIM-SM protocol mechanisms [RFC7761], we can treat the I-IP core as SSM-only to make it as simple as possible. There then remain only two scenarios to be discussed in detail:

- o E-IP network supports SSM

One possible way to make sure that the translated PIMv6 message reaches upstream AFBR is to set S' to a virtual IPv6 address that leads to the upstream AFBR. The unicast address translation should be achieved according to [RFC6052]

- o E-IP network supports ASM

The (S,G) source list entry and the (*,G) source list entry differ only in that the latter has both the WildCard (WC) and RPT bits of the Encoded-Source-Address set, while with the former, the bits are cleared (See Section 4.9.5.1 of [RFC7761]). As a result, the source list entries in (*,G) messages can be translated into source list entries in (S',G') messages by clearing both the WC and RPT bits at downstream AFBRs, and vice-versa for the reverse translation at upstream AFBRs.

5.4. Routing Mechanism

With mesh multicast, PIMv6 messages originating from a downstream AFBR need to be propagated to the correct upstream AFBR, and every AFBR needs the /96 prefix in "IPv4-Embedded IPv6 Source Address Format" [RFC6052].

To achieve this, every AFBR MUST announce the address of one of its E-IPv4 interfaces in the "v4" field [RFC6052] alongside the corresponding uPrefix46. The announcement MUST be sent to the other AFBRs through MBGP [RFC4760]. Every uPrefix64 that an AFBR announces

MUST be unique. "uPrefix64" is an IPv6 prefix, and the distribution mechanism is the same as the traditional mesh unicast scenario.

As the "v4" field is an E-IP address, and BGP messages are not tunneled through softwires or any other mechanism specified in [RFC5565], AFBRs MUST be able to transport and encode/decode BGP messages that are carried over the I-IP, and whose NLRI and NH are of the E-IP address family.

In this way, when a downstream AFBR receives an E-IP PIM (S,G) message, it can translate this message into (S',G') by looking up the IP address of the corresponding AFBR's E-IP interface. Since the uPrefix64 of S' is unique, and is known to every router in the I-IP network, the translated message will be forwarded to the corresponding upstream AFBR, and the upstream AFBR can translate the message back to (S,G).

When a downstream AFBR receives an E-IP PIM (*,G) message, S' can be generated with the "source address" field set to * (wildcard value). The translated message will be forwarded to the corresponding upstream AFBR. Since every PIM router within a PIM domain MUST be able to map a particular multicast group address to the same RP when the source address is set to wildcard value (see Section 4.7 of [RFC7761]), when the upstream AFBR checks the "source address" field of the message, it finds the IPv4 address of the RP, and ascertains that this is originally a (*,G) message. This is then translated back to the (*,G) message and processed.

6. Control Plane Functions of AFBR

AFBRs are responsible for the following functions:

6.1. E-IP (*,G) and (S,G) State Maintenance

E-IP (*,G) and (S,G) state maintenance for an AFBR is the same as E-IP (*,G) and (S,G) state maintenance for an mAFTR described in Section 7.2 of [RFC8114]

6.2. I-IP (S',G') State Maintenance

It is possible that the I-IP transit core runs another, non-transit, I-IP PIM-SSM instance. Since the translated source address starts with the unique "Well-Known" prefix or the ISP-defined prefix that MUST NOT be used by another service provider, mesh multicast will not influence non-transit PIM-SSM multicast at all. When an AFBR receives an I-IP (S',G') message, it MUST check S'. If S' starts with the unique prefix, then the message is actually a translated

E-IP (S,G) or (*,G) message, and the AFBR translate this message back to a PIMv4 message and process it.

6.3. E-IP (S,G,rpt) State Maintenance

When an AFBR wishes to propagate a Join/Prune(S,G,rpt) [RFC7761] message to an I-IP upstream router, the AFBR MUST operate as specified in Section 6.5 and Section 6.6.

6.4. Inter-AFBR Signaling

Assume that one downstream AFBR has joined an RPT of (*,G) and an SPT of (S,G), and decided to perform an SPT switchover (see Section 4.2.1 of [RFC7761]). According to [RFC7761], it should propagate a Prune(S,G,rpt) message along with the periodical Join(*,G) message upstream towards the RP. However, routers in the I-IP transit core do not process (S,G,rpt) messages since the I-IP transit core is treated as SSM-only. As a result, the downstream AFBR is unable to prune S from this RPT, so it will receive two copies of the same data for (S,G). In order to solve this problem, we introduce a new mechanism for downstream AFBRs to inform upstream AFBRs of pruning any given S from an RPT.

When a downstream AFBR wishes to propagate an (S,G,rpt) message upstream, it SHOULD encapsulate the (S,G,rpt) message, then send the encapsulated unicast message to the corresponding upstream AFBR, which we call "RP".

When RP' receives this encapsulated message, it MUST decapsulate the message as in the unicast scenario, and retrieve the original (S,G,rpt) message. The incoming interface of this message may be different to the outgoing interface which propagates multicast data to the corresponding downstream AFBR, and there may be other downstream AFBRs that need to receive multicast data of (S,G) from this incoming interface, so RP' should not simply process this message as specified in [RFC7761] on the incoming interface.

To solve this problem, we introduce an "interface agent" to process all the encapsulated (S,G,rpt) messages the upstream AFBR receives. The interface agent's RP' should prune S from the RPT of group G when no downstream AFBR is subscribed to receive multicast data of (S,G) along the RPT.

In this way, we ensure that downstream AFBRs will not miss any multicast data that they need. The cost of this is that multicast data for (S,G) will be duplicated along the RPT received by AFBRs affected by the SPT switch over, if at least one downstream AFBR

exists that has not yet sent Prune(S,G,rpt) messages to the upstream AFBR.

In certain deployment scenarios (e.g. if there is only a single downstream router), the interface agent function is not required.

The mechanism used to achieve this is left to the implementation. The following diagram provides one possible solution for an "interface agent" implementation:

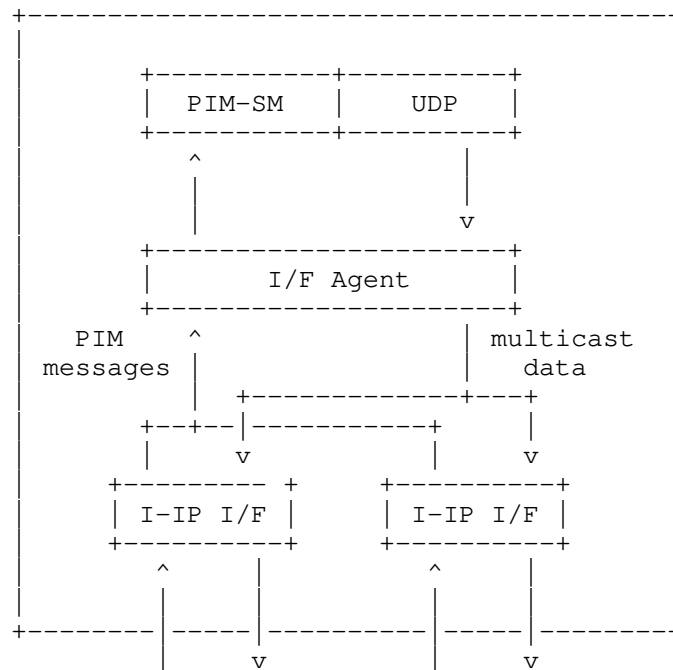


Figure 4: Interface Agent Implementation Example

Figure 4 shows an example of an interface agent implementation using UDP encapsulation. The interface agent has two responsibilities: In the control plane, it should work as a real interface that has joined $(*,G)$, representing of all the I-IP interfaces which are outgoing interfaces of the $(*,G)$ state machine, and process the (S,G,rpt) messages received from all the I-IP interfaces.

The interface agent maintains downstream (S,G,rpt) state machines for every downstream AFBR, and submits Prune (S,G,rpt) messages to the PIM-SM module only when every (S,G,rpt) state machine is in the Prune(P) or PruneTmp(P') state, which means that no downstream AFBR is subscribed to receive multicast data for (S,G) along the RPT of G. Once a (S,G,rpt) state machine changes to NoInfo(NI) state, which means that the corresponding downstream AFBR has switched to receive multicast data of (S,G) along the RPT again, the interface agent MUST send a Join (S,G,rpt) to the PIM-SM module immediately.

In the data plane, upon receiving a multicast data packet, the interface agent MUST encapsulate it at first, then propagate the encapsulated packet from every I-IP interface.

NOTICE: It is possible that an E-IP neighbor of RP' has joined the RPT of G, so the per-interface state machine for receiving E-IP Join/Prune (S,G,rpt) messages should be preserved.

6.5. SPT Switchover

After a new AFBR requests the receipt of traffic destined for a multicast group, it will receive all the data from the RPT at first. At this time, every downstream AFBR will receive multicast data from any source from this RPT, in spite of whether they have switched over to an SPT or not.

To minimize this redundancy, it is recommended that every AFBR's SwitchToSptDesired(S,G) function employs the "switch on first packet" policy. In this way, the delay in switchover to SPT is kept as small as possible, and after the moment that every AFBR has performed the SPT switchover for every S of group G, no data will be forwarded in the RPT of G, thus no more unnecessary duplication will be produced.

6.6. Other PIM Message Types

In addition to Join or Prune, other message types exist, including Register, Register-Stop, Hello and Assert. Register and Register-Stop messages are sent by unicast, while Hello and Assert messages are only used between directly linked routers to negotiate with each other. It is not necessary to translate these for forwarding, thus the processing of these messages is out of scope for this document.

6.7. Other PIM States Maintenance

In addition to states mentioned above, other states exist, including (*,*,RP) and I-IP (*,G') state. Since we treat the I-IP core as SSM-only, the maintenance of these states is out of scope for this document.

7. Data Plane Functions of the AFBR

7.1. Process and Forward Multicast Data

Refer to Section 7.4 of [RFC8114]. If there is at least one outgoing interface whose IP address family is different from the incoming interface, the AFBR MUST encapsulate this packet with mPrefix64-derived and uPrefix64-derived IPv6 address to form an IPv6 multicast packet.

7.2. TTL or Hop Count

Upon encapsulation, the TTL and hop account in the outer header SHOULD be set by policy. Upon decapsulation, the TTL and hop count in the inner header SHOULD be modified by policy, it MUST NOT be incremented and it MAY be decremented to reflect the cost of tunnel forwarding. Besides, processing of TTL and hop count information in protocol headers depends on the tunneling technology, which is out of scope of this document.

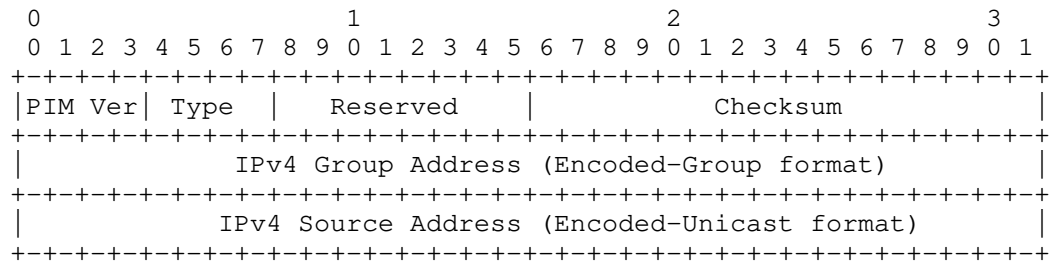
7.3. Fragmentation

The encapsulation performed by an upstream AFBR will increase the size of packets. As a result, the outgoing I-IP link MTU may not accommodate the larger packet size. It is not always possible for core operators to increase the MTU of every link, thus source fragmentation after encapsulation and reassembling of encapsulated packets MUST be supported by AFBRs [RFC5565]. PMTUD [RFC8201] SHOULD be enabled and ICMPv6 packets MUST NOT be filtered in the I-IP network. Fragmentation and tunnel configuration considerations are provided in Section 8 of [RFC5565]. The detailed procedure can be referred in Section 7.2 of [RFC2473].

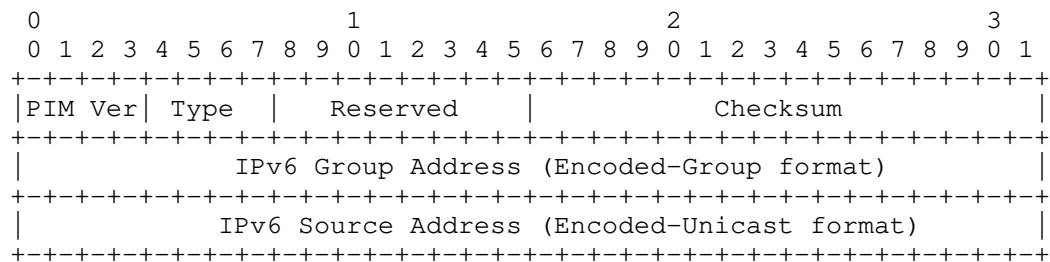
8. Packet Format and Translation

Because the PIM-SM Specification is independent of the underlying unicast routing protocol, the packet format in Section 4.9 of [RFC7761] remains the same, except that the group address and source address MUST be translated when traversing an AFBR.

For example, Figure 5 shows the register-stop message format in the IPv4 and IPv6 address families.



(1). IPv4 Register-Stop Message Format



(2). IPv6 Register-Stop Message Format

Figure 5: Register-Stop Message Format

In Figure 5, the semantics of fields "PIM Ver", "Type", "Reserved", and "Checksum" can be referred in Section 4.9 of [RFC7761].

IPv4 Group Address (Encoded-Group format): The encoded-group format of the IPv4 group address described in Section 4.9.1 of [RFC7761]

IPv4 Source Address (Encoded-Group format): The encoded-unicast format of the IPv4 source address described in Section 4.9.1 of [RFC7761]

IPv6 Group Address (Encoded-Group format): The encoded-group format of the IPv6 group address described in Section 5.2.

IPv6 Source Address (Encoded-Group format): The encoded-unicast format of the IPv6 source address described in Section 5.3.

9. Softwire Mesh Multicast Encapsulation

Softwire mesh multicast encapsulation does not require the use of any one particular encapsulation mechanism. Rather, it MUST accommodate a variety of different encapsulation mechanisms, and allow the use of encapsulation mechanisms mentioned in [RFC4925]. Additionally, all of the AFBRs attached to the I-IP network MUST implement the same

encapsulation mechanism, and follow the requirements mentioned in Section 8 of [RFC5565].

10. Security Considerations

The security concerns raised in [RFC4925] and [RFC7761] are applicable here.

The additional workload associated with some schemes, such as interface agents, could be exploited by an attacker to perform a DDOS attack.

Compared with [RFC4925], the security concerns should be considered more carefully: an attacker could potentially set up many multicast trees in the edge networks, causing too many multicast states in the core network. To defend against these attacks, BGP policies SHOULD be carefully configured, e.g., AFBRS only accept Well-Known prefix advertisements from trusted peers. Besides, cryptographic methods for authenticating BGP sessions [RFC7454] could be used.

11. IANA Considerations

This document includes no request to IANA.

12. 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>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC4925] Li, X., Ed., Dawkins, S., Ed., Ward, D., Ed., and A. Durand, Ed., "Softwire Problem Statement", RFC 4925, DOI 10.17487/RFC4925, July 2007, <<https://www.rfc-editor.org/info/rfc4925>>.
- [RFC5565] Wu, J., Cui, Y., Metz, C., and E. Rosen, "Softwire Mesh Framework", RFC 5565, DOI 10.17487/RFC5565, June 2009, <<https://www.rfc-editor.org/info/rfc5565>>.

- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <<https://www.rfc-editor.org/info/rfc6052>>.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <<https://www.rfc-editor.org/info/rfc6333>>.
- [RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <<https://www.rfc-editor.org/info/rfc6513>>.
- [RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <<https://www.rfc-editor.org/info/rfc7454>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC7899] Morin, T., Ed., Litkowski, S., Patel, K., Zhang, Z., Kebler, R., and J. Haas, "Multicast VPN State Damping", RFC 7899, DOI 10.17487/RFC7899, June 2016, <<https://www.rfc-editor.org/info/rfc7899>>.
- [RFC8114] Boucadair, M., Qin, C., Jacquenet, C., Lee, Y., and Q. Wang, "Delivery of IPv4 Multicast Services to IPv4 Clients over an IPv6 Multicast Network", RFC 8114, DOI 10.17487/RFC8114, March 2017, <<https://www.rfc-editor.org/info/rfc8114>>.
- [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>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

Appendix A. Acknowledgements

Wenlong Chen, Xuan Chen, Alain Durand, Yiu Lee, Jacni Qin and Stig Venaas provided useful input into this document.

Authors' Addresses

Mingwei Xu
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5822
Email: xumw@tsinghua.edu.cn

Yong Cui
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5822
Email: cuiyong@tsinghua.edu.cn

Jianping Wu
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5983
Email: jianping@cernet.edu.cn

Shu Yang
Shenzhen University
South Campus, Shenzhen University
Shenzhen 518060
P.R. China

Phone: +86-755-2653-4078
Email: yang.shu@szu.edu.cn

Chris Metz
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1-408-525-3275
Email: chmetz@cisco.com

Softwire Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

Q. Sun
H. Wang
Y. Cui
Tsinghua University
I. Farrer
S. Zoric
Deutsche Telekom AG
M. Boucadair
Orange
R. Asati
Cisco Systems, Inc.
July 8, 2016

A YANG Data Model for IPv4-in-IPv6 Softwires
draft-sun-softwire-yang-05

Abstract

This document defines a YANG data model for the configuration and operations (state, notification, RPC etc.) of IPv4-in-IPv6 Softwire Border Routers and Customer Premises Equipment. The model covers the Lightweight 4over6, MAP-E and MAP-T Softwire mechanisms.

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 January 9, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
1.2. Tree Diagrams	3
1.3. YANG Modelling of NAT44 Functionality	4
2. Common	4
3. Lightweight 4over6	4
4. MAP-E and MAP-T	4
5. Software YANG Tree Diagrams	4
5.1. Common Tree Diagrams	4
5.2. Lightweight 4over6 Tree Diagrams	5
5.3. MAP-E and MAP-T Tree Diagrams	8
5.4. Notifications for Software YANG	9
6. Software YANG Model	10
7. Example of Configure lw4o6 Binding-Table	26
8. Security Considerations	27
9. IANA Considerations	28
10. Acknowledgements	28
11. References	28
11.1. Normative References	28
11.2. Informative References	29
Authors' Addresses	30

1. Introduction

The IETF Softwire Working Group has developed several IPv4-in-IPv6 Softwire mechanisms to address various deployment contexts and constraints. As a companion to the architectural specification documents, this document focuses on the provisioning of A+P softwire functional elements: Border Routers (BRs) and Customer Premises Equipment (CEs).

This document defines a YANG data model [RFC6020] that can be used to configure and manage A+P Softwire elements using the NETCONF protocol [RFC6241]. DS-Lite YANG data model is defined in [I-D.boucadair-softwire-dslite-yang].

The Softwire YANG model is structured into two sub-models:

- o Lightweight 4over6 [RFC7596]
- o MAP-E [RFC7597] and MAP-T [RFC7599] (combined due to their common configuration parameters).

Two root containers are defined:

1. Container "softwire-config" holds the collection of YANG definitions common to all Softwire element configuration.
2. Container "softwire-state" holds YANG definitions for the operational state of the Softwire elements.

A NETCONF notify module is also included.

This approach has been taken so that the model can be easily extended to support additional Softwire mechanisms, if required.

1.1. Terminology

The reader should be familiar with the concepts and terms defined in [RFC7596], [RFC7597], [RFC7599], and the YANG data modelling language [RFC6020].

1.2. Tree Diagrams

The meaning of the symbols in these diagrams are as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature content.
- o Parentheses "(" and ")" enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Symbols after data node names: "?" means an optional node, and "*" denotes a list and leaf-list.
- o Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).

1.3. YANG Modelling of NAT44 Functionality

The model does not include CPE NAT-specific provisioning parameters that may be used for IPv4 address sharing other than the external IP address and port set which a softwire client may use for NAT44. NAT-specific considerations are out of scope of this document. A YANG model for the configuration and management of NAT gateways is described in [I-D.sivakumar-yang-nat].

2. Common

The following sections of the document are structured with the root of the Softwire YANG model (common to all mechanisms) described first. Subsequent sections describe the models relevant to the different softwire mechanisms. All functions are listed, but the YANG models use the "feature" statement to distinguish among the different softwire mechanisms. This document defines a new module named "ietf-softwire" for Softwire data models such that this module augments "ietf-ipv6-unicast-routing" module that is defined in [I-D.ietf-netmod-routing-cfg].

3. Lightweight 4over6

Lightweight 4over6 (binding) includes two elements: lwAFTR (BR) and lwB4 (CE). The lwAFTR holds configuration for IPv4-IPv6 address bindings which are used for the forwarding of traffic originating from lwB4s.

The lwB4 is configured with the relevant parameters for establishing the IPv4-in-IPv6 tunnel including an IPv6 address for the lwAFTR and the IPv4 configuration for NAT44.

4. MAP-E and MAP-T

MAP-E and MAP-T elements are provisioned with the MAP rules necessary for defining MAP domains and forwarding rules. For MAP-T CEs, an additional "ipv6-prefix" parameter is also included. Note that when referring to MAP-E/T (algorithm), the CE and BR shares the same model for configuration and management.

5. Softwire YANG Tree Diagrams

5.1. Common Tree Diagrams

Figure 1 describes the high level softwire YANG data model and the way tree is organized is common to all of the different softwire mechanisms listed in Section 1:


```

+--rw software-config
|   +--rw description?                string
|   +--rw binding {binding}?
|   |   +--rw br {br}?
|   |   +--rw cr {cr}?
|   +--rw algorithm {algorithm}?
+--ro software-state
|   +--ro description?                string
|   +--ro binding {binding}?
|   |   +--ro br {br}?
|   |   +--ro ce {ce}?
|   +--ro algorithm {algorithm}?

```

Figure 1: High Level Software YANG Tree Organization

5.2. Lightweight 4over6 Tree Diagrams

Figure 2 defines the software data model for lw4o6 (software binding mode) which includes lwAFTR (BR) and lwB4 (CE):

```

module: ietf-software
+--rw software-config
|   +--...
|   +--rw binding {binding}?
|   |   +--rw br {br}?
|   |   |   +--rw enable?                boolean
|   |   |   +--rw br-instances
|   |   |   |   +--rw br-instance* [id]
|   |   |   |   |   +--rw binding-table-versioning
|   |   |   |   |   |   +--rw binding-table-version?  uint64
|   |   |   |   |   |   +--rw binding-table-date?     yang:date-and-time
|   |   |   |   |   +--rw id                    uint32
|   |   |   |   |   +--rw name?                  string
|   |   |   |   |   +--rw software-num-threshold  uint32
|   |   |   |   |   +--rw tunnel-payload-mtu      uint16
|   |   |   |   |   +--rw tunnel-path-mru        uint16
|   |   |   |   +--rw binding-table
|   |   |   |   |   +--rw binding-entry* [binding-ipv6info]
|   |   |   |   |   |   +--rw binding-ipv6info      union
|   |   |   |   |   |   |   +--rw binding-ipv4-addr  inet:ipv4-address
|   |   |   |   |   |   |   +--rw port-set
|   |   |   |   |   |   |   |   +--rw psid-offset    uint8
|   |   |   |   |   |   |   |   +--rw psid-len      uint8
|   |   |   |   |   |   |   |   +--rw psid          uint16
|   |   |   |   |   |   +--rw br-ipv6-addr          inet:ipv6-address
|   |   |   |   |   +--rw lifetime?                 uint32
|   |   +--rw ce {ce}?

```

```

|         +--rw enable?                               boolean
|         +--rw ce-instances
|           +--rw ce-instance* [binding-ipv6info]
|             +--rw name?                               string
|             +--rw tunnel-payload-mtu                 uint16
|             +--rw tunnel-path-mru                   uint16
|             +--rw b4-ipv6-addr-format                boolean
|             +--rw binding-ipv6info                   union
|             +--rw binding-ipv4-addr                  inet:ipv4-address
|             +--rw port-set
|               +--rw psid-offset                     uint8
|               +--rw psid-len                         uint8
|               +--rw psid                             uint16
|             +--rw br-ipv6-addr                       inet:ipv6-address
|             +--rw lifetime?                           uint32
+--ro software-state
+--...
+--ro binding {binding}?
+--ro br {br}?
+--ro br-instances
+--ro br-instance* [id]
+--ro id                                             uint32
+--ro name?                                         string
+--ro sentPacket?                                yang:zero-based-counter64
+--ro sentByte?                                 yang:zero-based-counter64
+--ro rcvdPacket?                               yang:zero-based-counter64
+--ro rcvdByte?                                 yang:zero-based-counter64
+--ro droppedPacket?                             yang:zero-based-counter64
+--ro droppedByte?                              yang:zero-based-counter64
+--ro active-software-num?                       uint32
+--ro binding-table
+--ro binding-entry* [binding-ipv6info]
+--ro binding-ipv6info                           union
+--ro active?                                     boolean
+--ro ce {ce}?
+--ro ce-instances
+--ro ce-instance* [binding-ipv6info]
+--ro name?                                         string
+--ro binding-ipv6info                           union
+--ro sentPacket?                                yang:zero-based-counter64
+--ro sentByte?                                 yang:zero-based-counter64
+--ro rcvdPacket?                               yang:zero-based-counter64
+--ro rcvdByte?                                 yang:zero-based-counter64
+--ro droppedPacket?                             yang:zero-based-counter64
+--ro droppedByte?                              yang:zero-based-counter64

```

Figure 2: Softwire Lightweight 4over6 Data Model Tree Structure

The data model assumes that each CE/BR instance can: be enable/disabled, be provisioned with a dedicated configuration data, and maintain its own binding table.

Additional information on some of the important lwAFTR nodes is provided below:

- o binding-table-versioning: optionally used to add a incremental version number and/or timestamp to the binding table. This can be used for logging/data retention purposes. The version number is incremented and a new timestamp value written whenever a change is made to the contents of the binding table or a new binding table list is created.
- o binding-entry: used to define the binding relationship between 3-tuples, which contains the lwB4's IPv6 address/prefix, the allocated IPv4 address and restricted port-set. For detail information, please refer to [RFC7596].
- o tunnel-payload-mtu: used to set the IPv4 MTU for the lw4o6 tunnel.
- o tunnel-path-mru: used to set the maximum lw4o6 IPv6 encapsulating packet size that can be received.
- o psid-offset: used to set the number of offset bits.
- o psid-len: defines the number of ports that will be allocated for the software.
- o psid: used to identify the set of ports allocated for a specific software.
- o tunnel-num-threshold: used to set the maximum number of tunnels that can be created on the lw4o6 device simultaneously.
- o active-tunnel-num (ro): used to present the number of tunnels currently provisioned on the device.
- o active (ro): used to show the status of particular binding-entry.

Additional information on some of the important lwB4 nodes is provided below:

- o b4-ipv6-addr-format: indicates the format of lwB4 IPv6 address. If set to true, it indicates that the IPv6 source address of the lwB4 is constructed according to the description in Section 6 of [RFC7597]; if set to false, the lwB4 can use any /128 address from the assigned IPv6 prefix.

- o binding-ipv6info: used to set the IPv6 address type which is combined in a binding entry, for a complete address or a prefix.

5.3. MAP-E and MAP-T Tree Diagrams

Figure 3 defines the softwire data model for MAP-E and MAP-T:

```

module: ietf-softwire
  +--rw softwire-config
  |   +--...
  |   +--rw algorithm {algorithm}?
  |   |   +--rw enable?                               boolean
  |   |   +--rw algorithm
  |   |   |   +--rw algo-instance* [id]
  |   |   |   |   +--rw algo-versioning
  |   |   |   |   |   +--rw algo-version?           uint64
  |   |   |   |   |   +--rw algo-date?              yang:date-and-time
  |   |   |   |   +--rw id                           uint32
  |   |   |   |   +--rw name?                          string
  |   |   |   |   +--rw data-plane                     enumeration
  |   |   |   |   +--rw ea-len                          uint8
  |   |   |   |   +--rw rule-ipv6-prefix               inet:ipv6-prefix
  |   |   |   |   +--rw rule-ipv4-prefix               inet:ipv4-prefix
  |   |   |   |   +--rw forwarding                     boolean
  |   |   |   |   +--rw psid-offset                     uint8
  |   |   |   |   +--rw psid-len                       uint8
  |   |   |   |   +--rw tunnel-payload-mtu             uint16
  |   |   |   |   +--rw tunnel-path-mru                uint16
  |   |   |   |   +--rw br-ipv6-addr                   inet:ipv6-address
  |   |   |   |   +--rw dmr-ipv6-addr                   inet:ipv6-prefix
  |   +--ro softwire-state
  |   |   +--...
  |   +--ro algorithm {algorithm}?
  |   |   +--ro algo-instances
  |   |   |   +--ro algo-instance* [id]
  |   |   |   |   +--ro id                             int32
  |   |   |   |   +--ro name?                           string
  |   |   |   |   +--ro sentPacket?                      yang:zero-based-counter64
  |   |   |   |   +--ro sentByte?                       yang:zero-based-counter64
  |   |   |   |   +--ro rcvdPacket?                      yang:zero-based-counter64
  |   |   |   |   +--ro rcvdByte?                       yang:zero-based-counter64
  |   |   |   |   +--ro droppedPacket?                   yang:zero-based-counter64
  |   |   |   |   +--ro droppedByte?                    yang:zero-based-counter64

```

Figure 3: Softwire MAP-E and MAP-T Data Model Structure

Additional information on some of the important MAP-E and MAP-T nodes is provided below:

- o algo-versioning: optionally used to add a incremental version number and/or timestamp to the algorithm. This can be used for logging/data retention purposes. The version number is incremented and a new timestamp value written whenever a change is made to the algorithm or a new instance is created.
- o forwarding: specifies whether the rule can be used as a Forward Mapping Rule (FMR). If not set, this rule is a Basic Mapping Rule (BMR) only and must not be used for forwarding. See Section 4.1 of [RFC7598].
- o ea-len: used to set the length of the Embedded-Address (EA), which defined in the mapping rule for a MAP domain.
- o dmr-ipv6-prefix: defines the Default Mapping Rule (DMR) for MAP-T. This parameter is optional when configuring a MAP-T BR.
- o stat-count (ro): use to show the numbers of packets and bytes information of specific device respectively.

5.4. Notifications for Softwire YANG

This section describes the tree structure for notifications. These notifications pertain to the configuration and monitoring portions of the specific Softwire mechanisms. The logic is that the softwire instance notifies the NETCONF client with the index for a mapping entry and the NETCONF client retrieves the related information from the operational datastore of that instance.

```

module: ietf-softwire
notifications:
  +---n softwire-binding-br-event {binding,br}?
  |   +--ro br-id?                  -> /softwire-state/binding/br/.../id
  |   +--ro invalid-entry*          -> /softwire-config/binding/br/.../binding-table/b
inding-entry/binding-ipv6info
  |   +--ro added-entry*            inet:ipv6-address
  |   +--ro modified-entry*         -> /softwire-config/binding/br/.../binding-table/b
inding-entry/binding-ipv6info
  +---n softwire-binding-ce-event {binding,ce}?
  |   +--ro ce-binding-ipv6-addr-change  inet:ipv6-address
  +---n softwire-algorithm-instance-event {algorithm}?
  |   +--ro algo-id                  -> /softwire-config/algorithm/.../id
  |   +--ro invalid-entry*           -> /softwire-config/algorithm/.../id
  |   +--ro added-entry*             -> /softwire-config/algorithm/.../id
  |   +--ro modified-entry*          -> /softwire-config/algorithm/.../id

```

Figure 4: Softwire Notifications Data Model Structure

Additional information on some of the important notification nodes is listed below:

- o invalid-entry, added-entry, modified-entry: used to notify the client that a specific binding entry or MAP rule is expired or invalidated, added, or modified.
- o ce-binding-ipv6-addr-change: used to notify that the lwB4's binding-ipv6-address has been changed or the value of the 'b4-ipv6-addr-format' is "False".

6. Softwire YANG Model

This module imports typedefs from [RFC6991].

<CODE BEGINS> file "ietf-softwire@2016-06-04.yang"

```
module ietf-softwire {
  namespace "urn:ietf:params:xml:ns:yang:ietf-softwire";
  prefix "softwire";

  import ietf-inet-types {prefix inet; }
  import ietf-yang-types {prefix yang; }

  organization "Softwire Working Group";

  contact
    "
    Qi Sun <sunqi.ietf@gmail.com>
    Hao Wang <wangh13@mails.tsinghua.edu.cn>
    Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
    Ian <Farrer ian.farrer@telekom.de>
    Sladjana Zoric <sladjana.zoric@telekom.de>
    Mohamed Boucadair <mohamed.boucadair@orange.com>
    Rajiv <Asati rajiva@cisco.com>
    ";

  description
    "This document defines a YANG data model for the configuration and
    management of A+P Softwire Border Routers (BRs) and Customer
    Premises Equipment (CEs). It covers Lightweight 4over6,
    MAP-E and MAP-T mechanisms.

    Copyright (c) 2016 IETF Trust and the persons identified
    as authors of the code. All rights reserved.
    This version of this YANG module is part of RFC XXX; see the RFC
    itself for full legal notices.";

  revision 2016-06-04 {
    description
      "Version-05: Combined MAP-E/MAP-T into a single tree. Added binding
```

```
        table/algorithm versioning";
        reference "-05";
    }

    revision 2015-09-30 {
        description
            "Version-04: Fix YANG syntax; Add flags to map-rule; Remove
            the map-rule-type element. ";
        reference "-04";
    }

    revision 2015-04-07 {
        description
            "Version-03: Integrate lw4over6; Update state nodes; Correct
            grammar errors; Reuse groupings; Update descriptions.
            Simplify the model.";
        reference "-03";
    }

    revision 2015-02-10 {
        description
            "Version-02: Add notifications.";
        reference "-02";
    }

    revision 2015-02-06 {
        description
            "Version-01: Correct grammar errors; Reuse groupings; Update
            descriptions.";
        reference "-01";
    }

    revision 2015-02-02 {
        description
            "Initial revision.";
        reference "-00";
    }

/*
 * Features
 */

    feature binding {
        description
            "Lightweight 4over6 (binding) is an IPv4-over-IPv6 tunnelling
            transition mechanism. Lightweight 4over6 is a solution designed
            specifically for complete independence between IPv6 subnet
```

prefix (and /128 IPv6 address) and IPv4 address with or without IPv4 address sharing.

This is accomplished by maintaining state for each softwire (per-subscriber state) in the central lwAFTR and a hub-and-spoke forwarding architecture. In order to delegate the NAPT function and achieve IPv4 address sharing, port-restricted IPv4 addresses needs to be allocated to CEs.

Besides lw4o6, this feature also covers MAP in 1:1 mode (offset=0, PSID explicit)";

```
reference
  "RFC7596";
}

feature br {
  if-feature binding;
  description
    "The AFTR for Lightweight 4over6, so-called lwAFTR (BR). This
    feature indicates that a instance functions as a lwAFTR (BR).
    A lwAFTR (BR) is an IPv4-in-IPv6 tunnel concentrator that
    maintains per-subscriber IPv4-IPv6 address binding.";
}

feature ce {
  if-feature binding;
  description
    "The B4 for Lightweight 4over6, so-called lwB4 (CE). This
    feature indicates that a instance functions as a lwB4 (CE). A
    lwB4 (ce) is an IPv4-in-IPv6 tunnel initiator. It is
    dual-stack capable node, either a directly connected end-host
    or a CE. It sources IPv4 connections using the configured
    port-set and the public IPv4 address.";
}

feature algorithm {
  description
    "MAP-E is an IPv6 transition mechanism for transporting IPv4
    packets across an IPv6 network using IP encapsulation. MAP-E
    allows for a reduction of the amount of centralized state using
    rules to express IPv4/IPv6 address mappings. This introduces an
    algorithmic relationship between the IPv6 subnet
    and IPv4 address.
    The Mapping of Address and Port - Translation (MAP-T)
    architecture is a double stateless NAT64 based solution. It uses
    the stateless algorithmic address & transport layer port mapping
    scheme defined in MAP-E. The MAP-T solution differs from MAP-E in
```



```
        the use of IPv4-IPv6 translation, rather than encapsulation, as
        the form of IPv6 domain transport.
        This feature indicates the instance functions as a MAP-E or
        MAP-T instance.";
    reference
        "RFC7597 & RFC7599";
}

/*
 * Grouping
 */

grouping port-set {
    description
        "Use the PSID algorithm to represent a range of transport layer
        ports.";
    leaf psid-offset {
        type uint8 {
            range 0..16;
        }
        mandatory true;
        description
            "The number of offset bits. In Lightweight 4over6, the default
            value is 0 for assigning one contiguous port range. In MAP-E/T,
            the default value is 6, which excludes system ports by default
            and assigns distributed port ranges. If the this parameter is
            larger than 0, the value of offset MUST be greater than 0.";
    }
    leaf psid-len {
        type uint8 {
            range 0..15;
        }
        mandatory true;
        description
            "The length of PSID, representing the sharing ratio for an
            IPv4 address.";
    }
    leaf psid {
        type uint16;
        mandatory true;
        description
            "Port Set Identifier (PSID) value, which identifies a set
            of ports algorithmically.";
    }
}

grouping binding-entry {
    description
```

```
"The lwAFTR maintains an address binding table that contains
the binding between the lwB4's IPv6 address, the allocated IPv4
address and restricted port-set.";
leaf binding-ipv6info {
  type union {
    type inet:ipv6-address;
    type inet:ipv6-prefix;
  }
  mandatory true;
  description
    "The IPv6 information for a binding entry.
    If it's an IPv6 prefix, it indicates that
    the IPv6 source address of the lwB4 is constructed
    according to the description in RFC7596;
    if it's an IPv6 address, it means the lwB4 uses
    any /128 address from the assigned IPv6 prefix.
    ";
}
leaf binding-ipv4-addr {
  type inet:ipv4-address;
  mandatory true;
  description
    "The IPv4 address assigned to the lwB4, which is
    used as the IPv4 external address
    for lwB4 local NAPT44.";
}
container port-set {
  description
    "For Lightweight 4over6, the default value
    of offset should be 0, to configure one contiguous
    port range.";
  uses port-set {
    refine psid-offset {
      default "0";
    }
  }
}
leaf br-ipv6-addr {
  type inet:ipv6-address;
  mandatory true;
  description
    "The IPv6 address for lwaftr.";
}
leaf lifetime {
  type uint32;
  units seconds;
  description "The lifetime for the binding entry";
}
```

```
    }

/*
    grouping nat-table {

        description
            "Grouping 'nat-table' is not extended. The current mechanism
            is focusing on the provisioning of external IP address and
            port set; other NAT-specific considerations are out of scope.";
    }
*/

    grouping traffic-stat {
        description "Traffic statistics";
        leaf sentPacket {
            type yang:zero-based-counter64;
            description "Number of packets sent.";
        }
        leaf sentByte {
            type yang:zero-based-counter64;
            description "Traffic sent, in bytes";
        }
        leaf rcvdPacket {
            type yang:zero-based-counter64;
            description "Number of packets received.";
        }
        leaf rcvdByte {
            type yang:zero-based-counter64;
            description "Traffic received, in bytes";
        }
        leaf droppedPacket {
            type yang:zero-based-counter64;
            description "Number of packets dropped.";
        }
        leaf droppedByte {
            type yang:zero-based-counter64;
            description "Traffic dropped, in bytes";
        }
    }

/*
    * Configuration Data Nodes
    */

    container softwire-config {
        description
```

```
    "The configuration data for Softwire instances. And the shared
    data describes the softwire data model which is common to all of
    the different softwire mechanisms, such as description.";
  leaf description {
    type string;
    description
      "A textual description of Softwire.";
  }
  container binding {
    if-feature binding;
    description
      "lw4over6 (binding) configuration.";
    container br {
      if-feature br;
      description
        "Indicate this instance supports the lwAFTR (BR) function.
        The instances advertise the BR feature through the
        capability exchange mechanism when a NETCONF session is
        established.";
      leaf enable {
        type boolean;
        description
          "Enable/disable the lwAFTR (BR) function.";
      }
    }
    container br-instances {
      description
        "A set of BRs to be configured.";
      list br-instance {
        key "id";
        description
          "A set of lwAFTRs to be configured.";
        container binding-table-version {
          description "binding table's version";
          leaf binding-table-version {
            type uint64;
            description "Incremental version number
              to the binding table";
          }
          leaf binding-table-date {
            type yang:date-and-time;
            description "Timestamp to the binding
              table";
          }
        }
      }
      leaf id {
        type uint32;
        mandatory true;
        description "An instance identifier.";
      }
    }
  }
}
```

```
    }
    leaf name {
        type string;
        description "The name for the lwaftr.";
    }
    leaf software-num-threshold {
        type uint32;
        mandatory true;
        description
            "The maximum number of tunnels that can be created on
            the lwAFTR.";
    }
    leaf tunnel-payload-mtu {
        type uint16;
        mandatory true;
        description
            "The payload MTU for Lightweight 4over6 tunnel.";
    }
    leaf tunnel-path-mru {
        type uint16;
        mandatory true;
        description
            "The path MRU for Lightweight 4over6 tunnel.";
    }
    container binding-table {
        description "binding table";
        list binding-entry {
            key "binding-ipv6info";
            description "binding entry";
            uses binding-entry;
        }
    }
}

container ce {
    if-feature ce;
    description
        "Indicate this instance supports the lwb4 (CE) function.
        The instances advertise the CE feature through the
        capability exchange mechanism when a NETCONF session is
        established.";
    leaf enable {
        type boolean;
        description
            "Enable/disable the lwb4 (CE) function.";
    }
}
```

```
    container ce-instances {
      description
        "A set of CEs to be configured.";
      list ce-instance {
        key "binding-ipv6info";
        description "instances for CE";
        leaf name {
          type string;
          description "The CE's name.";
        }
        leaf tunnel-payload-mtu {
          type uint16;
          mandatory true;
          description
            "The payload MTU for Lightweight 4over6 tunnel.";
        }
        leaf tunnel-path-mru {
          type uint16;
          mandatory true;
          description
            "The path MRU for Lightweight 4over6 tunnel.";
        }
        leaf b4-ipv6-addr-format {
          type boolean;
          mandatory true;
          description
            "The format of lwB4 (CE) IPv6 address. If set to true,
            it indicates that the IPv6 source address of the lwB4
            is constructed according to the description in
            [RFC7596]; if set to false, the lwB4 (CE)
            can use any /128 address from the assigned IPv6
            prefix.";
        }
        uses binding-entry;
      }
    }
  }
}

container algorithm {
  if-feature algorithm;
  description
    "Indicate the instances support the MAP-E and MAP-T function.
    The instances advertise the map-e feature through the
    capability exchange mechanism when a NETCONF session is
    established.";
  leaf enable {
    type boolean;
```

```
    description
      "Enable/disable the MAP-E or MAP-T function.";
  }
  container algo-instances {
    description
      "A set of MAP-E or MAP-T instances to be configured,
       applying to BRs and CEs. A MAP-E/T instance defines a MAP
       domain comprising one or more MAP-CE and MAP-BR";
    list algo-instance {
      key "id";
      description "instance for MAP-E/MAP-T";
      container algo-versioning {
        description "algorithm's version";
        leaf algo-version {
          type uint64;
          description "Incremental version number to
            the algorithm";
        }
        leaf algo-date {
          type yang:date-and-time;
          description "Timestamp to the algorithm";
        }
      }
      leaf id {
        type uint32;
        mandatory true;
        description "Algorithm Instance ID";
      }
      leaf name {
        type string;
        description "The name for the instance.";
      }
      leaf data-plane {
        type enumeration {
          enum "encapsulation" {
            description "encapsulation for MAP-E";
          }
          enum "translation" {
            description "translation for MAP-T";
          }
        }
        description
          "Encapsulation is for MAP-E while translation is
           for MAP-T";
      }
      leaf ea-len {
        type uint8;
        mandatory true;
      }
    }
  }
```

```
description
  "Embedded Address (EA) bits are the IPv4 EA-bits
  in the IPv6 address identify an IPv4
  prefix/address (or part thereof) or
  a shared IPv4 address (or part thereof)
  and a port-set identifier.
  The length of the EA-bits is defined as
  part of a MAP rule for a MAP domain.";
}
leaf rule-ipv6-prefix {
  type inet:ipv6-prefix;
  mandatory true;
  description
    "The Rule IPv6 prefix defined in the mapping rule.";
}
leaf rule-ipv4-prefix {
  type inet:ipv4-prefix;
  mandatory true;
  description
    "The Rule IPv4 prefix defined in the mapping rule.";
}
leaf forwarding {
  type boolean;
  mandatory true;
  description
    "This parameter specifies whether the rule may be used for
    forwarding (FMR). If set, this rule is used as an FMR;
    if not set, this rule is a BMR only and must not be used
    for forwarding.";
}
leaf psid-offset {
  type uint8 {
    range 0..16;
  }
  mandatory true;
  description
    "The number of offset bits. In Lightweight 4over6, the default
    value is 0 for assigning one contiguous port range. In MAP-E/T,
    the default value is 6, which excludes system ports by default
    and assigns distributed port ranges. If the this parameter is
    larger than 0, the value of offset MUST be greater than 0.";
}
leaf psid-len {
  type uint8 {
    range 0..15;
  }
  mandatory true;
  description
```



```

        "The length of PSID, representing the sharing ratio for an
        IPv4 address.";
    }
    leaf tunnel-payload-mtu {
        type uint16;
        description
            "The payload MTU for MAP-E tunnel.";
    }
    leaf tunnel-path-mru {
        type uint16;
        description
            "The path MRU for MAP-E tunnel.";
    }
    leaf br-ipv6-addr {
        type inet:ipv6-address;
        mandatory true;
        description
            "The IPv6 address of the MAP-E BR.";
    }
    leaf dmr-ipv6-prefix {
        type inet:ipv6-prefix;
        description
            "The IPv6 prefix of the MAP-T BR. ";
    }
    }
}
}
}

/*
 * Operational state Data Nodes
 */

container software-state {
    config false;
    description
        "The operational state data for Softwire instances. ";
    leaf description {
        type string;
        description
            "A textual description of the softwire instances.";
    }
    container binding {
        if-feature binding;
        description
            "lw4over6 (binding) state.";
        container br {
            if-feature br;

```

```
config false;
description
  "Indicate this instance supports the lwAFTR (BR) function.
  The instances advertise the lwaftr (BR) feature through the
  capability exchange mechanism when a NETCONF session is
  established.";
container br-instances {
  description
    "A set of BRs.";
  list br-instance {
    key "id";
    description "instances for BR";
    leaf id {
      type uint32;
      mandatory true;
      description "id";
    }
    leaf name {
      type string;
      description "The name for this lwaftr.";
    }
  }
  uses traffic-stat;
  leaf active-software-num {
    type uint32;
    description
      "The number of currently active tunnels on the
      lw4over6 (binding) instance.";
  }
  container binding-table {
    description "id";
    list binding-entry {
      key "binding-ipv6info";
      description "An identifier of the binding entry.";
      leaf binding-ipv6info {
        type union {
          type inet:ipv6-address;
          type inet:ipv6-prefix;
        }
        mandatory true;
        description
          "The IPv6 information used to identify
          a binding entry. ";
      }
      leaf active {
        type boolean;
        description
          "Status of a specific tunnel.";
      }
    }
  }
}
```

```

    }
  }
}

container ce {
  if-feature ce;
  config false;
  description
    "Indicate this instance supports the lwb4 (CE) function.
    The instances advertise the lwb4 (CE) feature through the
    capability exchange mechanism when a NETCONF session is
    established.";
  container ce-instances {
    description
      "Status of the configured CEs.";
    list ce-instance {
      key "binding-ipv6info";
      description "a lwb4 (CE) instance.";
      leaf name {
        type string;
        description "The CE's name.";
      }
      leaf binding-ipv6info {
        type union {
          type inet:ipv6-address;
          type inet:ipv6-prefix;
        }
        mandatory true;
        description
          "The IPv6 information used to identify
          a binding entry. ";
      }
      uses traffic-stat;
    }
  }
}

container algorithm {
  if-feature algorithm;
  config false;
  description
    "Indicate the instances support the MAP-E and MAP-T function.
    The instances advertise the map-e/map-t feature through the
    capability exchange mechanism when a NETCONF session is

```

```

        established.";
    container algo-instances {
        description
            "Status of MAP-E instance(s).";
        list algo-instance {
            key "id";
            description "Instances for algorithm";
            leaf id {
                type uint32;
                mandatory true;
                description "id";
            }
            leaf name {
                type string;
                description "The map-e instance name.";
            }
            uses traffic-stat;
        }
    }
}

/*
 * Notifications
 */
notification software-br-event {
    if-feature binding;
    if-feature br;
    description "Notification for BR.";

    leaf br-id {
        type leafref {
            path
                "/software-state/binding/br/br-instances/"
                + "br-instance/id";
        }
        description "...";
    }
}
leaf-list invalid-entry {
    type leafref {
        path
            "/software-config/binding/br/br-instances/"
            + "br-instance[id=current()../br-id]/"
            + "binding-table/binding-entry/binding-ipv6info";
    }
    description
        "Notify the client that a specific binding entry has been

```

```
        expired/invalid. The binding-ipv6info identifies an entry.";
    }
    leaf-list added-entry {
        type inet:ipv6-address;
        description
            "Notify the client that a binding entry has been added.
            The ipv6 address of that entry is the index. The client
            get other information from the lwaftr about the entry
            indexed by that ipv6 address.
            ";
    }
    leaf-list modified-entry {
        type leafref {
            path
                "/software-config/binding/br/br-instances/"
                + "br-instance[id=current()/../br-id]/"
                + "binding-table/binding-entry/binding-ipv6info";
        }
        description "...";
    }
}

notification software-ce-event {
    if-feature binding;
    if-feature ce;
    description "CE notification";
    leaf ce-binding-ipv6-addr-change {
        type inet:ipv6-address;
        mandatory true;
        description
            "The source tunnel IPv6 address of the lwb4.
            If 'b4-ipv6-addr-format' is false, or the lwb4's
            binding-ipv6-address changes for any reason,
            it SHOULD notify the NETCONF client.";
    }
}

notification software-algorithm-instance-event {
    if-feature algorithm;
    description "Notifications for MAP-E or MAP-T.";
    leaf algo-id {
        type leafref {
            path
                "/software-config/algorithm/algo-instances/algo-instance/id";
        }
        mandatory true;
        description "MAP-E or MAP-T event.";
    }
}
```

```
leaf-list invalid-entry-id {
  type leafref {
    path
      "/softwire-config/algorithm/algo-instances/algo-instance/id";
  }
  description "Invalid entry event.";
}
leaf-list added-entry {
  type leafref {
    path
      "/softwire-config/algorithm/algo-instances/algo-instance/id";
  }
  description "Added entry.";
}
leaf-list modified-entry {
  type leafref {
    path
      "/softwire-config/algorithm/algo-instances/algo-instance/id";
  }
  description "Modified entry.";
}
}
}
<CODE ENDS>
```

7. Example of Configure lw4o6 Binding-Table

The lwAFTR maintains an address binding table which contains the following 3-tuples:

- o IPv6 Address for a single lwB4
- o Public IPv4 Address
- o Restricted port-set

The entry has two functions: the IPv6 encapsulation of inbound IPv4 packets destined to the lwB4 and the validation of outbound IPv4-in-IPv6 packets received from the lwB4 for de-capsulation.

Let's consider an example to add an entry that maintains the relationship between 3-tuples of lwB4 (2001:db8::1), '192.0.2.1' and '1234' in the binding table of the lwAFTR (2001:db8::2). Here is the example binding-table configuration xml:

```
<rpc message-id="101"
  xmlns:nc="urn:params:xml:ns:yang:ietf-softwire:1.0">
<!-- replace with IANA namespace when assigned. -->
  <edit-config>
    <target>
      <running/>
    </target>
  <softwire-config>
    <lw4o6-aftr>
      <lw4o6-aftr-instances>
        <lw4o6-aftr-instance>
          <aftr-ipv6-addr>2001:db8::2</aftr-ipv6-addr>
          <binding-table>
            <binding-entry>
              <binding-ipv4-addr>192.0.2.1</binding-ipv4-addr>
              <port-set>
                <psid>1234</psid>
              </port-set>
              <binding-ipv6-addr>2001:db8::1</binding-ipv6-addr>
              <active>1</active>
            </binding-entry>
          </binding-table>
        </lw4o6-aftr-instance>
      </lw4o6-aftr-instances>
    </lw4o6-aftr>
  </softwire-config>
```

Figure 5: lw4o6 Binding-Table Configuration XML

8. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

All data nodes defined in the YANG module which can be created, modified and deleted (i.e., config true, which is the default). These data nodes are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations.

9. IANA Considerations

This document requests IANA to register the following URI in the "IETF XML Registry" [RFC3688].

URI: urn:ietf:params:xml:ns:yang:softwire
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [RFC6020].

name: ietf-dslite-aftr
namespace: urn:ietf:params:xml:ns:yang:softwire
prefix: softwire
reference: RFC XXXX

10. Acknowledgements

The authors would like to thank Lishan Li, Bert Wijnen, Giles Heron, and Ole Troan for their contributions to this work.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<http://www.rfc-editor.org/info/rfc6242>>.

- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", RFC 7596, DOI 10.17487/RFC7596, July 2015, <<http://www.rfc-editor.org/info/rfc7596>>.
- [RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", RFC 7597, DOI 10.17487/RFC7597, July 2015, <<http://www.rfc-editor.org/info/rfc7597>>.
- [RFC7598] Mrugalski, T., Troan, O., Farrer, I., Perreault, S., Dec, W., Bao, C., Yeh, L., and X. Deng, "DHCPv6 Options for Configuration of Softwire Address and Port-Mapped Clients", RFC 7598, DOI 10.17487/RFC7598, July 2015, <<http://www.rfc-editor.org/info/rfc7598>>.
- [RFC7599] Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <<http://www.rfc-editor.org/info/rfc7599>>.

11.2. Informative References

- [I-D.boucadair-softwire-dslite-yang]
Boucadair, M., Jacquenet, C., and S. Sivakumar, "YANG Data Model for the DS-Lite Address Family Transition Router (AFTR)", draft-boucadair-softwire-dslite-yang-04 (work in progress), June 2016.
- [I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.
- [I-D.sivakumar-yang-nat]
Sivakumar, S., Boucadair, M., and S. <>, "YANG Data Model for Network Address Translation (NAT)", draft-sivakumar-yang-nat-04 (work in progress), March 2016.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.

Authors' Addresses

Qi Sun
Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5822
Email: sunqi.ietf@gmail.com

Hao Wang
Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5822
Email: wangh13@mails.tsinghua.edu.cn

Yong Cui
Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6260-3059
Email: yong@csnet1.cs.tsinghua.edu.cn

Ian Farrer
Deutsche Telekom AG
CTO-ATI, Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: ian.farrer@telekom.de

Sladjana Zoric
Deutsche Telekom AG
CTO-IPT, Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: sladjana.zoric@telekom.de

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Rajiv Asati
Cisco Systems, Inc.
7025 Kit Creek Rd.
RTP, NC 27709
USA

Email: Rajiva@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

S. Vinapamula
Juniper Networks
M. Boucadair
France Telecom
October 19, 2015

Recommendations for Prefix Binding in the Softwire DS-Lite Context
draft-vinapamula-softwire-dslite-prefix-binding-12

Abstract

This document discusses issues induced by the change of the Dual-Stack Lite (DS-Lite) Basic Bridging BroadBand (B4) IPv6 address and sketches a set of recommendations to solve those issues.

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 21, 2016.

Copyright Notice

Copyright (c) 2015 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	2
2. The Problem	3
3. Introducing Subscriber-Mask	4
4. Recommendations	4
5. Security Considerations	6
6. Privacy Considerations	6
7. IANA Considerations	7
8. References	7
8.1. Normative references	7
8.2. Informative references	8
Acknowledgments	9
Authors' Addresses	9

1. Introduction

IPv6 deployment models assume IPv6 prefixes are delegated by Service Providers to the connected CPEs (Customer Premises Equipments) or hosts, which in turn derive IPv6 addresses from that prefix. In the case of Dual-Stack Lite (DS-Lite) [RFC6333], which is an IPv4 service continuity mechanism over an IPv6 network, the Basic Bridging BroadBand (B4) element derives an IPv6 address for the IPv4-in-IPv6 software setup purposes.

The B4 element might obtain a new IPv6 address, for a variety of reasons that include (but are not limited to) a reboot of the CPE, power outage, DHCPv6 lease expiry, or other actions undertaken by the Service Provider. If this occurs, traffic forwarded to a B4's previous IPv6 address may never reach its destination or be delivered to another B4 that now uses the address formerly assigned to the original B4. This situation affects all mapping types, both implicit (e.g., by sending a TCP SYN) and explicit (e.g., using Port Control Protocol (PCP) [RFC6887]). The problem is further elaborated in Section 2.

This document proposes recommendations to soften the impact of such renumbering issues (Section 4).

This document complements [RFC6908].

1.1. Requirements Language

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

2. The Problem

Since private IPv4 addresses assigned to hosts serviced by a B4 element overlap across multiple CPEs, the IPv6 address of a B4 element plays a key role in de-multiplexing connections, enforcing policies, and in identifying associated resources assigned for each of the connections maintained by the Address Family Transition Router (AFTR, [RFC6333]). For example, these resources maintain state of Endpoint-Independent Mapping (EIM, Section 4.1 of [RFC4787]), Endpoint-Independent Filtering (EIF, Section 5 of [RFC4787]), preserve the external IPv4 address assigned in the AFTR (i.e., "IP address pooling" behavior as defined in Section 4.1 of [RFC4787]), PCP mappings, etc.

However, the IPv6 address used by the B4 element may change for some reason, e.g., because of a change in the CPE itself or maybe because of privacy extensions enabled for generating the IPv6 address (e.g., [RFC7217] or [RFC4941]). Whenever the B4's IPv6 address changes, the associated mappings created in the AFTR are no longer valid. This may result in the creation of a new set of mappings in the AFTR.

Furthermore, a misbehaving user may be tempted to change the B4's IPv6 address in order to "grab" more ports and resources at the AFTR side. This behavior can be seen as a potential Denial of Service (DoS) attack from misbehaving users. Note that this DoS attack can be achieved whatever the port assignment policy enforced by the AFTR (individual ports, port sets, randomized port bulks, etc.).

Service Providers may want to enforce policies in order to limit the usage of the AFTR resources on a per-subscriber basis for fairness of resource usage (see REQ-4 of [RFC6888]). These policies are used for dimensioning purposes and also to ensure that AFTR resources are not exhausted. If the derived B4's IPv6 address can change, resource tracking using that address will give incomplete results. Also, whenever the B4's IPv6 address changes, enforcing policies based on this address doesn't resolve stale mappings hanging around in the system, consuming not only system resources, but also reducing the available quota of resources per subscriber. Clearing those mappings can be envisaged, but that will cause a lot of churn in the AFTR and could be disruptive to existing connections, which is not desirable. More concretely, if stale mappings have not be migrated to the new B4's IPv6 address so that packets can be forwarded to the appropriate B4, all incoming packets that are associated with those mappings will be rejected by the AFTR. Such behavior is not desirable from a service quality of experience.

When application servers are hosted behind a B4 element, and when there is a change of the B4's IPv6 address which results in a change

of the external IPv4 address and/or the external port number at the AFTR side, these servers have to advertise about their change (see Section 1.1 of [RFC7393]). Means to discover the change of B4's IPv6 address, the external IPv4 address and/or the external port are therefore required. Latency issues are likely to be experienced when an application server has to advertise its newly assigned external IPv4 address and port, and the application clients have to discover that newly assigned address and/or port and re-initiate connections with the application server.

A solution to these problems is to enforce policies based on the IPv6 prefix assigned to DS-Lite serviced subscribers instead of the B4's IPv6 address. Section 3 introduces the subscriber-mask that is meant to derive the IPv6 prefix assigned to a subscriber's CPE from the source IPv6 address of a packet received from a B4 element.

3. Introducing Subscriber-Mask

The subscriber-mask is defined as an integer that indicates the length of significant bits to be applied on the source IPv6 address (internal side) to identify unambiguously a CPE.

Subscriber-mask is an AFTR system-wide configuration parameter that is used to enforce generic per-subscriber policies. Applying these generic policies does not require configuring every subscriber's prefix.

Subscriber-mask must be configurable; the default value is 56. The default value is motivated by current practices to assign IPv6 prefix lengths of /56 to end-sites (e.g., [RIPE][LACNIC]).

Example: suppose the 2001:db8:100:100::/56 prefix is assigned to a DS-Lite enabled CPE. Suppose also that the 2001:db8:100:100::1 address is the IPv6 address used by the B4 element that resides in that CPE. When the AFTR receives a packet from this B4 element (i.e., the source address of the IPv4-in-IPv6 packet is 2001:db8:100:100::1), the AFTR applies the subscriber-mask (e.g., 56) on the source IPv6 address to compute the associated prefix for this B4 element (that is 2001:db8:100:100::/56). Then, the AFTR enforces policies based on that prefix (2001:db8:100:100::/56), not on the exact source IPv6 address.

4. Recommendations

In order to mitigate the issues discussed in Section 2, the following recommendations are made:

1. A policy SHOULD be enforced at the AFTR to limit the number of active DS-Lite softwires per subscriber. The default value MUST be 1.

This policy aims to prevent a misbehaving subscriber from mounting several DS-Lite softwires that would consume additional AFTR resources (e.g., get more external ports if the quota were enforced on a per-softwire basis, consume extra processing induced by a large number of active softwires).

2. Resource contexts created and maintained by the AFTR SHOULD be based on the delegated IPv6 prefix instead of the B4's IPv6 address. The AFTR derives the delegated prefix from the B4's IPv6 address by means of a configured subscriber-mask (Section 3). Administrators SHOULD configure per-prefix limits of resource usage, instead of per-tunnel limits. These resources include the maximum number of active flows, the maximum number of PCP-created mappings, NAT pool resources, etc.
3. In the event a new IPv6 address is assigned to the B4 element, the AFTR SHOULD migrate existing state to be bound to the new IPv6 address. This operation ensures that traffic destined to the previous B4's IPv6 address will be redirected to the newer B4's IPv6 address. The destination IPv6 address for tunneling return traffic from the AFTR SHOULD be the last seen as the B4's IPv6 source address from the CPE.

This recommendation avoids stale mappings at the AFTR and minimizes the risk of service disruption for subscribers.

The AFTR uses the subscriber-mask to determine whether two IPv6 addresses belong to the same CPE (e.g., if the subscriber-mask is set to 56, the AFTR concludes that 2001:db8:100:100::1 and 2001:db8:100:100::2 belong to the same CPE assigned with 2001:db8:100:100::/56).

As discussed in Section 5, changing the source B4's IPv6 address may be used as an attack vector. Packets with new B's IPv6 address from the same prefix SHOULD be rate limited. It is RECOMMENDED to set this rate limit to 30 minutes; other values can be set on a per deployment basis.

One side effect of migrating mapping state is that a server deployed behind an AFTR does not need to update its DNS records (if any) by means of dynamic DNS, for example. As far as a dedicated mapping is instantiated, migrating the state during its validity lifetime will ensure that the same external IP address and port are assigned to that server.

4. In the event of change of the CPE WAN's IPv6 prefix, unsolicited PCP ANNOUNCE messages SHOULD be sent by the B4 element to internal hosts connected to the PCP-capable CPE so that they update their mappings accordingly.

This allows internal PCP clients to update their mappings with the new B4's IPv6 address and to trigger updates to rendezvous servers (e.g., dynamic DNS). A PCP-based dynamic DNS solution is specified in [RFC7393].

5. When a new prefix is assigned to the CPE, stale mappings may exist in the AFTR. This will consume both implicit and explicit resources. In order to avoid such issues, stable IPv6 prefix assignment is RECOMMENDED.
6. In case for any reason an IPv6 prefix has to be reassigned, it is RECOMMENDED to reassign an IPv6 prefix (that was previously assigned to a given CPE) to another CPE only when all the resources in use associated with that prefix are cleared from the AFTR. Doing so avoids redirecting traffic, destined to the previous prefix owner, to the new one.

5. Security Considerations

Security considerations related to DS-Lite are discussed in [RFC6333].

Enforcing the recommendations documented in Section 4 together with rate limiting softwares with new source IPv6 addresses from the same prefix defend against DoS attacks that would result in varying the B4's IPv6 address to exhaust AFTR resources. A misbehaving CPE can be blacklisted by enforcing appropriate policies based on the prefix derived from the subscriber-mask.

6. Privacy Considerations

A CPE connected to a DS-Lite network is identified by a set of information that is specific to each network domain (e.g., service credentials, device identifiers, etc.). This document does not make any assumption nor introduce new requirements on how such identification is implemented network-wide.

This document adheres to Sections 6 and 8 of [RFC6333] for handling IPv4-in-IPv6 packets and IPv4 translation operations. In particular, this document does not leak extra information in packets exiting a DS-Lite network domain.

The recommendations in Section 4 (bullet 6, in particular) ensure the traffic is forwarded to a legitimate CPE. If those recommendations are not implemented, privacy concerns may arise (e.g., If an IPv6 prefix is reassigned while mapping entries associated with that prefix are still active in the AFTR, sensitive data that belong to a previous prefix owner may be disclosed to the new prefix owner).

These recommendations do not interfere with privacy extensions for generating IPv6 addresses (e.g., [RFC7217] or [RFC4941]). These recommendations allow a CPE to generate new IPv6 addresses with privacy extensions without experiencing DS-Lite service degradation. Even if activating privacy extensions makes it more difficult to track a CPE over time when compared to using a permanent Interface Identifier, tracking a CPE is still possible based on the first 64 bits of the IPv6 address. This is even exacerbated for deployments relying on stable IPv6 prefixes.

This document does not nullify the privacy effects that may motivate the use of non-stable IPv6 prefixes. Particularly, the subscriber-mask does not allow to identify a CPE across renumbering (even within a DS-Lite network domain). This document mitigates some of the undesired effects of reassigning an IPv6 prefix to another CPE (e.g., update a rendezvous service, clear stale mappings).

7. IANA Considerations

This document does not require any action from IANA.

8. References

8.1. Normative references

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <<http://www.rfc-editor.org/info/rfc6333>>.
- [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <<http://www.rfc-editor.org/info/rfc6887>>.

8.2. Informative references

- [LACNIC] LACNIC, "IPv6 Address Allocation and Assignment Policies", July 2015, <<http://www.lacnic.net/en/web/lacnic/manual-4>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<http://www.rfc-editor.org/info/rfc4787>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <<http://www.rfc-editor.org/info/rfc6888>>.
- [RFC6908] Lee, Y., Maglione, R., Williams, C., Jacquenet, C., and M. Boucadair, "Deployment Considerations for Dual-Stack Lite", RFC 6908, DOI 10.17487/RFC6908, March 2013, <<http://www.rfc-editor.org/info/rfc6908>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.
- [RFC7393] Deng, X., Boucadair, M., Zhao, Q., Huang, J., and C. Zhou, "Using the Port Control Protocol (PCP) to Update Dynamic DNS", RFC 7393, DOI 10.17487/RFC7393, November 2014, <<http://www.rfc-editor.org/info/rfc7393>>.
- [RIPE] RIPE, "IPv6 Address Allocation and Assignment Policy", August 2015, <<https://www.ripe.net/publications/docs/ripe-650>>.

Acknowledgments

G. Krishna, C. Jacquenet, I. Farrer, Y. Lee, Q. Sun, R. Weber, T. Taylor, D. Harkins, D. Gillmor, S. Sivakumar, A. Cooper, and B. Campbell provided useful comments. Many thanks to them.

Authors' Addresses

Suresh Vinapamula
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA

Phone: +1 408 936 5441
EMail: sureshk@juniper.net

Mohamed Boucadair
France Telecom
Rennes 35000
France

EMail: mohamed.boucadair@orange.com