Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 27, 2018

M. Smith
July 26, 2017

Skinny IPv6 in IPv6 Tunnelling
draft-smith-skinny-ipv6-in-ipv6-tunnelling-00

Abstract

This memo proposes a method of tunnelling IPv6 packets inside IPv6 packets with a reduced tunnelling overhead.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There have been some recent proposals to insert additional information via Extension Headers (EHs) into IPv6 packets that are currently "in-flight", with in-flight meaning that they have left the source host that originated the packet and are on their way across the network towards the packet's destination host [I-D.voyer-6man-extension-header-insertion][I-D.francois-dots-ipv6-signal-option].

The fundamental drawback of inserting EHs into existing packets is that while the inserted EH is present, the IPv6 header's Source Address field's semantics are incorrect, as the device with the recorded source address is not the source of the inserted EH.

If the inserted EH causes the packet to trigger a mechanism that relies on IPv6 header Source Address semantics, such as Path MTU Discovery [RFC1981], the triggered mechanism may fail. This is because the mechanism's messages will be sent to the source address indicated, rather than the device that inserted the EH. The device inserting the EH is not identified in the packet so that messages can be sent to it if necessary.

Troubleshooting is also impacted, in particular when packets are received over long Autonomous System paths across the Internet. If an inserted EH is received and causes failures, it isn't possible to identify which AS inserted the EH from the received packet. Significant time will need to be spent contacting each AS that may have inserted the EH, to then inform them that they aren't removing the EHs they're inserting.

The alternative to inserting EHs is to add EHs using IPv6-in-IPv6 tunnelling e.g., via [RFC2423]. A new IPv6 packet is created with the new EH, with the original packet then being the new IPv6 packet's payload. As the source of the added EH is now identified in the outer IPv6 packet header, mechanisms such as PMTUD will work correctly if addition of the EH triggers the mechanism.

The drawback of conventional IPv6-in-IPv6 tunnelling is the overhead of adding another IPv6 header to the original IPv6 packet, a minimum of an additional 40 octets. It is likely that insertion of EHs into existing in-flight packets was motivated by trying to avoid this tunnelling encapsulation overhead.

From the perspective of the original and inner IPv6 packet, IPv6-in-IPv6 tunnelling is treating the outer or underlying IPv6 network as though it was a link-layer. The inner IPv6 packet is treated by the outer IPv6 packet as a link-layer opaque payload [I-D.ietf-intarea-tunnels].

A property unique to IPv6-in-IPv6 tunnelling, unlike other link-layer encapsulations of IPv6 packets, is that all fixed header fields of the outer IPv6 "link-layer" packet have the same semantics as all fixed header fields of the inner and being carried IPv6 packet.

If opaque treatment of the inner IPv6 packet by the outer IPv6 packet is compromised, matching inner and outer IPv6 packet field semantics provides the opportunity to reduce the IPv6-in-IPv6 tunnelling overhead. This can be achieved by partially or fully using a number of the outer IPv6 packet's fields as proxies for the inner packet's fields while the inner packet is in-flight over the outer IPv6 "link-layer". These inner IPv6 packet's fields are removed during link-layer encapsulation and then restored during link-layer decapsulation, using the values of the fields from the outer IPv6 packet.

This memo describes this method of IPv6-in-IPv6 tunnelling, giving it the name "Skinny IPv6-in-IPv6 Tunnelling". It preserves source address semantics for both the inner and outer packet source addresses, allowing mechanisms such as PMTUD to function, while also reducing tunnelling overhead.

In essence, this memo is a specification for a method of using IPv6 as a link-layer, adding to others in that set such as [RFC2423].

A number of the techniques and concepts have been taken from or inspired by the previous draft [I-D.smith-enhance-vne-with-ipv6].

2. Terminology

Terminology used in this memo is from "IP Tunnels in the Internet Architecture" [I-D.ietf-intarea-tunnels].

3. Skinny IPv6-in-IPv6 Virtual Link Characteristics

A virtual link created using this method has the following characteristics:

- o Limited to carrying IPv6 packets as link-layer payload. As many IPv4 packet fields have similar if not the same semantics as IPv6 packet fields, a similar method could be used for carrying IPv4 in IPv6, however this is out of scope for this memo.

- o Point-to-multipoint transmission via IPv6 multicast. A link node (tunnel endpoint) can send a single packet, have it duplicated by the link (the underlying IPv6 network) and then have the duplicates arrive at multiple destination link nodes. The set of destination link nodes or tunnel endpoints are identified by an IPv6 multicast address.

- o Can be used to create a single link-layer instance (e.g., a single point-to-point connection between two routers, hosts, or a host and a router, or a single multi-access link interconnecting a set of routers), or as one of a set of component links used to create a link-layer instance (i.e., a multi-link link-layer topology, where links are bridged together.). In this latter case, a link-layer loop prevention mechanism may be needed.

4. Addressing Tunnel Endpoints

Initially suggested in [I-D.smith-enhance-vne-with-ipv6], Skinny IPv6-in-IPv6 tunnelling uses individual /64s to identify individual tunnel endpoints, rather than using individual IPv6 addresses (i.e., /128s).

For Skinny IPv6-in-IPv6 unicast tunnelling to a tunnel endpoint, this allows the lower 64 bits of the outer packet's source and destination addresses (the Interface Identifier or IID parts) to carry the corresponding lower 64 bits of the inner packet's source and destination addresses. These two sets of 64 bits are removed from

the inner IPv6 packet's addresses, effectively lowering the tunnelling overhead by 16 octets.

Multicast Skinny IPv6-in-IPv6 tunnel endpoints cannot be identified by /64s, as IPv6 multicast groups cannot be uniquely created at the /64 boundary. When multicast tunnelling towards a multicast group of tunnel endpoints, only the inner packet source address lower 64 bits are carried in the outer packet's source address lower 64 bits, and removed from the inner source address. The inner packet's full 128 bit destination address is carried in the Skinny IPv6-in-IPv6 tunnel header. The removal of the inner packet's source address lower 64 bits reduces the tunnelling overhead by 8 octets.

An emergent benefit of this technique of carrying inner packet IIDs in the outer packet's addresses between tunnel endpoints is that inner packet address entropy useful for load balancing is exposed to the network carrying the outer packets.

Skinny IPv6-in-IPv6 tunnel endpoints are expected to be represented as virtual interfaces in routers or hosts, and therefore will have at least a Link-Local unicast /128 address per the [RFC4291] interface address requirements, and perhaps other individual IPv6 addresses.

The use of /64s to identify tunnel endpoints means that for other non-Skinny IPv6-in-IPv6 traffic originated by or sent to the tunnel endpoint, any of the individual IPv6 addresses within the /64 are valid as local source or destination addresses for this traffic. Examples of other types of traffic originated by or sent to tunnel endpoints could be ICMPv6, TCP, or UDP.

When receiving non-Skinny IPv6-in-IPv6 packets, a tunnel endpoint is to consider all destination addresses within its assigned /64 valid and local, passing the received packets' payload to the appropriate upper layer protocol if supported, or dropping if not and possibly emitting an appropriate ICMPv6 message.

When originating non-Skinny IPv6-in-IPv6 packets from a source address within the assigned /64, a common individual address within the /64 should be used for all packets, to avoid confusing a receiver that does not expect a single device to be assigned an entire /64. Alternatively, per connection or per protocol common individual source address could be used if useful. Consistent and expected source address usage by the sender is the requirement at the receiver that needs to be met.

If the tunnel endpoint exists on a router (more specifically, is a forwarding interface), then the source address for tunnel endpoint

originated packets could be the Subnet-Router anycast address [RFC4291], the endpoint's /64 plus an IID of all zeros.

Alternatively, if the tunnel endpoint exists on a host (more specifically, is a non-forwarding interface), then the source address should not be the Subnet-Router anycast address. Some other address within the /64, such as a statically configured address, or an address generated by the SLAAC procedure [RFC4862][RFC7217][RFC8064] should be used. Non-Subnet Router anycast source addresses could also be used by tunnel endpoints on routers.

In any of these cases, source and destination address selection will occur per the standard IPv6 default address selection procedure [RFC6724], unless a specific source and/or destination address is provided.

5. Proxied Inner IPv6 Packet Header Fields

While the inner IPv6 packet is being tunnelled, the following inner packet header fields are proxied by the fields in the outer IPv6 packet header [RFC2460]:

- o Version
- o Traffic Class
- o Flow Label
- o Source Address lower 64 bits for both unicast and multicast tunnel destination endpoints
- o Destination Address lower 64 bits for unicast tunnel destination endpoints

6. Non-Proxied Inner IPv6 Packet Header Fields

While the inner IPv6 packet is being tunnelled, values of the following inner packet header fields [RFC2460] are carried in fields in the Skinny IPv6-in-IPv6 Extension Header. This allows them to be restored upon decapsulation, as the outer packet's header field values cannot or may not be used.

- o Payload Length
- o Hop Limit
- o Source Address upper 64 bits

- Destination Address upper 64 bits

- Destination Address lower 64 bits for multicast tunnel destination endpoints

7. Hop Limit Handling

Conventional link-layers commonly do not have a hop count or hop limit field in their frames to mitigate the effects of link-layer forwarding loops. A separate mechanism or method must be used to prevent forwarding loops occurring.

When IPv6 is being used as a link-layer, the packet's Hop Limit field and processing acts as a link-layer forwarding loop mitigation mechanism.

It could be useful to have the number of hops the outer packet took included in the inner packet's recorded Hop Limit after the inner packet has traversed the tunnel. This would expose the outer IPv6 packet hops that occurred during tunnelling to the inner IPv6 packet's network. This may be useful for troubleshooting, or mitigating forwarding loops sooner.

Alternatively, the number of hops the outer IPv6 packet took could be hidden from the inner IPv6 packet's network, by restoring the Hop Limit value from the Hop Limit field value in the Skinny IPv6-in-IPv6 Extension Header. Restoring the inner packet's pre-Skinny Ipv6-in-IPv6 Hop Limit value is critical when the Neighbor Discovery protocol [RFC4861] is used across the Skinny IPv6-in-IPv6 virtual link. This is necessary as to ensure these ND packets have not come from an off-link source, their Hop Limit is set to 255 upon transmission and the recipient will only accept ND packets that arrive with a Hop Limit value of 255.

Whether to use the outer packet's Hop Limit value or to use the inner Skinny IPv6-in-IPv6 EH preserved Hop Limit value during inner IPv6 packet header reconstruction is indicated by the value of the Skinny IPv6-in-IPv6 EH preserved Hop Limit field value when it arrives at the destination tunnel endpoint. If the value is 0, the outer IPv6 packet header's current Hop Limit value is used during reconstruction, where as if the value is non-zero (e.g., 255 for ND packets), then that value is used during inner IPv6 packet reconstruction.

8. Skinny IPv6-in-IPv6 Tunnel Extension Header

While the original IPv6 packet is being tunnelled, the Skinny IPv6 Extension Header carries the particles of the original inner IPv6 packet's header that are or may not be being proxied in the outer IPv6 packet header.

8.1. Extension Header rather than Destination Option

[RFC6564] requires justification for the creation of a new Extension Header rather than the creation of a new Destinations Options Header option.

The creation of a Skinny IPv6-in-IPv6 Extension Header is requested for the following reasons:

- o Simplified header processing, which will be beneficial at high packet per second rates. The receiving device will only need to look up a single Next Header value in either the outer IPv6 header or the previous Extension Header to determine that a Skinny IPv6-in-IPv6 EH follows. This is in comparison to a two stage lookup to process a Destination Option in a Destination Option EH.

- o Consistent with other IPv6-in-IPv6 tunnelling encapsulations that use Extension Headers such as conventional IPv6-in-IPv6 [RFC2423] and GRE [RFC7676].

8.2. Extension Header Format

(ASCII art to come!)

- o Next Header - 8 bits.

- o Hdr Ext Len - 8 bits.

- o Reserved - 24 bits.

- o Inner Payload Length - 16 bits.

- o Inner Hop Limit - 8 bits.

- o Inner SA 64 bit Prefix - 64 bits / 8 octets.

- o Inner DA 64 bit Prefix - 64 bits / 8 octets.

- o Inner DA 64 bit IID - 64 bits / 8 octets.

8.3. Extension Header Fields

- o Next Header - 8 bits field, specifying the header type of the header that immediately followed the inner IPv6 packet's original IPv6 fixed header. May be another extension header value, or an upper layer protocol value such as UDP or TCP.

- o Hdr Ext Len - 8 bit field, specifying the length of this Skinny IPv6-in-IPv6 extension header in units of 8 octets, minus the first unit of 8 octets. The valid values are 2 or 3.

- o Reserved - 24 bits field reserved for future use, to 8 octet align the extension header. Set to zero upon transmission, ignored upon receipt. A possible future use of one or more of these octets is as a flag field.

- o Inner Payload Length - 16 bits Payload Length field value from the inner IPv6 packet before Skinny IPv6-in-IPv6 encapsulation occurred.

- o Inner Hop Limit - 8 bits Hop Limit field value from inner IPv6 packet before Skinny IPv6-in-IPv6 encapsulation occurred, or zero.

- o Inner SA 64 bit Prefix - upper 64 bits / 8 octets of the inner IPv6 packet's source address, prior to Skinny IPv6-in-IPv6 encapsulation.

- o Inner DA 64 bit Prefix - upper 64 bits / 8 octets of the inner IPv6 packet's destination address, prior to Skinny IPv6-in-IPv6 encapsulation.

- o Inner DA 64 bit IID - optional lower 64 bits / 8 octets of the inner IPv6 packet's destination address when the outer Skinny IPv6-in-IPv6 packet's destination is a multicast group of tunnel endpoints.

9. Encapsulation Procedure

The following steps describe the encapsulation procedure performed by the ingress Skinny IPv6-in-IPv6 tunnel endpoint upon a IPv6 packet that is being tunnelled. Note that it is a functional description, meaning that implementations may perform different and more performance oriented steps that achieve the same functional outcome.

After receiving the packet to be Skinny IPv6-in-IPv6 tunnelled from the upper IPv6 network layer:

9.1. Skinny IPv6-in-IPv6 EH Construction

1. Create a new instance of the Skinny IPv6-in-IPv6 Extension Header. If the destination tunnel endpoint is a multicast destination, the new EH instance will need to provide the optional Inner DA 64 bit IID field.

2. Set the new EH's Next Header value to the Next Header value in the original IPv6 packet header's Next Header field value.

3. If the destination Skinny-IPv6-in-IPv6 tunnel endpoint is an individual /64, set the new EH's Hdr Ext Len field value to 2. If the destination Skinny-IPv6-in-IPv6 tunnel endpoint is a multicast address, set the new EH's Hdr Ext Len field value to 3.

4. Set the new EH's Reserved field bits all to zeros.

5. Set the new EH's Inner Payload Length field to the original IPv6 packet header's Payload Length field value.

6. If the tunnel endpoint is configured to use the outer IPv6 packet header's Hop Limit value after tunnelling when reconstructing the inner IPv6 packet during decapsulation, set the new EH's Inner Hop Limit field value to zero.

7. If the tunnel endpoint is not configured to use the original IPv6 packet header's Hop Limit value after tunnelling, the default configuration, set the new EH's Inner Hop Limit field value to the original inner IPv6 packet's Hop Limit value (note that this will be after the Hop Limit has been decremented if the IPv6 packet is being forwarded per [RFC2460]).

8. Set the new EH's Inner SA 64 bit Prefix field to the high order 8 octets of the original IPv6 packet header's source address high order 8 octets.

9. Set the new EH's Inner DA 64 bit Prefix field to the high order 8 octets of the original IPv6 packet header's destination address high order 8 octets.

10. If the destination Skinny-IPv6-in-IPv6 tunnel endpoint is a multicast address, set the new EH's Inner DA 64 bit IID field to the low order 8 octets of the original IPv6 packet header's destination address low order 8 octets.

9.2. Outer IPv6 Packet Construction

1. Create a new instance of an IPv6 packet header, and set its Version field value to 6.

2. Set the new IPv6 packet header's Traffic Class field value to the original IPv6 packet header's Traffic Class field value.

3. Set the new IPv6 packet header's Flow Label field value to the original IPv6 packet header's Flow Label field value.

4. If the tunnel endpoint is configured to use the outer IPv6 packet header's Hop Limit value after tunnelling when reconstructing the inner IPv6 packet during decapsulation, set the new IPv6 packet header's Hop Limit to the original inner IPv6 packet header's Hop Limit field value.

5. If the tunnel endpoint is not configured to use the original IPv6 packet header's Hop Limit value after tunnelling, the default configuration, set the new IPv6 packet header's Hop Limit field value to the device's current Hop Limit default value when originating packets.

6. Set the new IPv6 packet header's Source Address high order 8 octets/64 bits to the tunnel endpoint's /64 value.

7. Set the new IPv6 packet header's Source Address low order 8 octets/64 bits to the original IPv6 packet's Source Address low order 8 octets.

8. Set the new IPv6 packet header's Destination Address high order 8 octets/64 bits to the destination tunnel endpoint's /64 value, or for a multicast tunnel destination, the high order 8 octets of the destination multicast address.

9. Set the new IPv6 packet header's Destination Address low order 8 octets/64 bits to the original IPv6 packet's Destination Address lower order 8 octets, or for a multicast tunnel destination, the low order 8 octets of the destination multicast address.

10. Append any additional tunnel endpoint specific EHs to the new IPv6 packet header.

11. Set the new IPv6 packet header's Next Header field value to the first of the tunnel endpoint specific EHs selector value.

12. Append the Skinny IPv6-in-IPv6 EH to the end of the existing set of EHs.

13. Append the original IPv6 packet's set of EHs after the Skinny IPv6-in-IPv6 EH, if any exist.

14. Append the original IPv6 packet's payload.

15. Set the new IPv6 packet header's Payload Length value based on the tunnel endpoint specific EHs, the Skinny IPv6-in-IPv6 EH and the original IPv6 packet's payload.

16. Discard the original IPv6 packet, as it has now been fully Skinny IPv6-in-IPv6 encapsulated.

17. Send the new IPv6 Skinny IPv6-in-IPv6 packet.

10. Decapsulation Procedure

The following steps describe the decapsulation procedure performed by the egress Skinny IPv6-in-IPv6 tunnel endpoint upon a IPv6 packet that has been tunnelled. Note that it is a functional description, meaning that implementations may perform different and more performance oriented steps that achieve the same functional outcome. (An obvious optimisation is to utilise the received packet header when reconstructing the original Skinny IPv6-in-IPv6 tunnelled packet's header, avoiding a number of copy operations among other things.)

After receiving the Skinny IPv6-in-IPv6 packet from the network:

1. Perform any Extension Header processing for the EHs that reside after the received fixed IPv6 header and before the Skinny IPv6-in-IPv6 Extension Header.

2. Create a new instance of an IPv6 packet header, and set its Version field value to 6. This is the IPv6 packet header that is being used to reconstruct the original inner IPv6 packet.

3. From the received Skinny IPv6-in-IPv6 outer packet header, copy the Traffic Class field value into the new IPv6 header's Traffic Class field.

4. From the received Skinny IPv6-in-IPv6 outer packet header, copy the Flow Label field value into the new IPv6 header's Flow Label field.

5. From the Skinny IPv6-in-IPv6 Extension Header, copy the Inner Payload Length field value into the new IPv6 header's Payload Length field.

6. From the Skinny IPv6-in-IPv6 Extension Header, copy the Next Header field value into the new IPv6 header's Next Header field.

7. If the Skinny IPv6-in-IPv6 Extension Header's Inner Hop Limit field value is 0, copy the received Skinny IPv6-in-IPv6 outer packet header's Hop Limit field value into the new IPv6 header's Hop Limit field.

8. If the Skinny IPv6-in-IPv6 Extension Header's Inner Hop Limit field value is not 0, copy the Skinny IPv6-in-IPv6 Extension Header's Inner Hop Limit field value into the new IPv6 header's Hop Limit field.

9. From the Skinny IPv6-in-IPv6 Extension Header, copy the SA 64 bit Prefix into the high order 64 bits of the new IPv6 header's Source Address field.

10. From the received Skinny IPv6-in-IPv6 outer packet header, copy the low order 64 bits of the Source Address into the low order 64 bits of the the new IPv6 header's Source Address field.

11. From the Skinny IPv6-in-IPv6 Extension Header, copy the DA 64 bit Prefix into the high order 64 bits of the new IPv6 header's Destination Address field.

12. If the Skinny IPv6-in-IPv6 Extension Header's Hdr Ext Len field value is 2, copy the low order 64 bits of the Destination Address from the received Skinny IPv6-in-IPv6 outer packet header into the low order 64 bits of the new IPv6 header's Destination Address field.

13. If the Skinny IPv6-in-IPv6 Extension Header's Hdr Ext Len field value is 3, copy the Skinny IPv6-in-IPv6 Extension Header's Inner DA 64 bit IID field value into the low order 64 bits of the new IPv6 header's Destination Address field.

14. If the Skinny IPv6-in-IPv6 Extension Header's Hdr Ext Len field value is not 2 or 3, discard the new IPv6 packet header and the received Skinny IPv6-in-IPv6 packet; if required, record them as discarded (e.g., in a tunnel interface discarded packet counter); and abort the decapsulation procedure.

15. Append any Extension Headers that followed the Skinny IPv6-in-IPv6 Extension Header to the new IPv6 header.

16. Append the Skinny IPv6-in-IPv6's packet's payload to the new IPv6 packet, after the appended Extension Headers.

17. Discard the received Skinny IPv6-in-IPv6 packet.

18. Submit the new IPv6 packet to the device's upper IPv6 layer for further processing (such as forwarding or local processing based on the destination address).

11. Reduction in Overhead

Conventional IPv6 tunnelling described in [RFC2423] adds at least another 40 octets to the original packet being tunnelled, as that is the size of the fixed IPv6 header.

Using the method described in this memo, for packets destined to a single (unicast) tunnel endpoint, this 40 octet overhead is reduced to 24 octets, a saving of 16 octets. This is a 40 percent saving.

Using the method described in this memo, for packets destined to multiple tunnel endpoint identified by an IPv6 multicast address, this 40 octet overhead is reduced to 32 octets, a saving of 8 octets. This is a 20 percent saving.

12. IANA Considerations

IANA are requested to allocate an IPv6 Next Header value for use by the Skinny IPv6-in-IPv6 Extension header, per [RFC5237].

13. Security Considerations

There are no security considerations specific to Skinny IPv6-in-IPv6 tunnelling. General tunnelling security considerations apply [RFC6169].

14. Acknowledgements

Review and comments were provided by YOUR NAME HERE!

This memo was prepared using the xml2rfc tool.

15. Change Log [RFC Editor please remove]

draft-smith-skinny-ipv6-in-ipv6-tunnelling-00, initial version, 2017-07-13

draft-smith-skinny-ipv6-in-ipv6-tunnelling-01, first revision, 2017-07-13

o Forgot Abstract in 00. Facepalm!

draft-smith-skinny-ipv6-in-ipv6-tunnelling-02, second revision, 2017-07-13

o Typo in Abstract. Facepalm again!

draft-smith-skinny-ipv6-in-ipv6-tunnelling-01, third revision, 2017-0x-xx

o Revert revision back to 00, as IETF ID submission tool complains. Previous were private revisions, waiting for IETF ID tool to become available again (two weeks pre-IETF99 meeting)

16. Informative References

[I-D.francois-dots-ipv6-signal-option]
Francois, J., Lahmadi, A., and M. Davids, "IPv6 DOTS Signal Option", draft-francois-dots-ipv6-signal-option-02 (work in progress), May 2017.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-07 (work in progress), June 2017.

[I-D.smith-enhance-vne-with-ipv6]
Smith, M., "Enhancing Virtual Network Encapsulation with IPv6", draft-smith-enhance-vne-with-ipv6-07 (work in progress), October 2015.

[I-D.voyer-6man-extension-header-insertion]
daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Leddy, J., Filsfils, C., Previdi, S., Salsano, S., Cianfrani, A., Lebrun, D., Bonaventure, O., Jonnalagadda, P., Sharif, M., Elmalky, H., Abdelsalam, A., Raszuk, R., Ayyangar, A., Steinberg, D., and W. Henderickx, "Insertion of IPv6 Segment Routing Headers in a Controlled Domain", draft-voyer-6man-extension-header-insertion-00 (work in progress), March 2017.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC2423] Vaudreuil, G. and G. Parsons, "VPIM Voice Message MIME Sub-type Registration", RFC 2423, DOI 10.17487/RFC2423, September 1998, <http://www.rfc-editor.org/info/rfc2423>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC5237] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, <http://www.rfc-editor.org/info/rfc5237>.

[RFC6169] Krishnan, S., Thaler, D., and J. Hoagland, "Security Concerns with IP Tunneling", RFC 6169, DOI 10.17487/RFC6169, April 2011, <http://www.rfc-editor.org/info/rfc6169>.

[RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <http://www.rfc-editor.org/info/rfc6564>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[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>.

[RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC 7676, DOI 10.17487/RFC7676, October 2015, <http://www.rfc-editor.org/info/rfc7676>.

[RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <http://www.rfc-editor.org/info/rfc8064>.

Author's Address

Mark Smith
PO BOX 521
HEIDELBERG, VIC 3084
AU

Email: markzzzsmith@gmail.com