

behave
Internet-Draft

Intended status: Standards Track
Expires: April 21, 2011

C. Bao
X. Li
CERNET Center/Tsinghua University
October 18, 2010

Extended IPv6 Addressing for Encoding Port Range
draft-bcx-address-fmt-extension-00

Abstract

This document discusses an extension of the algorithmic translation between IPv4 and IPv4-translatable IPv6 addresses. The extended address format contains transport-layer port range information which allows several IPv6 nodes to share a single IPv4 address with each node managing a different range of ports. This address format extension can be used for IPv4/IPv6 translation, as well as IPv4 over IPv6 tunneling.

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, 2011.

Copyright Notice

Copyright (c) 2010 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Applicability Scope	3
1.2. Conventions	3
2. Extended IPv4-translatable IPv6 Address	3
2.1. Port-range Coding Algorithm	4
2.2. Extended Address Format	5
2.3. Suffix for Port Range Encoding	6
2.4. Stateless Suffix Translation Algorithm	7
2.5. Partial-state Suffix Translation Algorithm	8
2.6. ICMP Packet Handling	8
3. IANA Considerations	8
4. Security Considerations	8
5. Acknowledgements	9
6. References	9
6.1. Normative References	9
6.2. Informative References	9
Authors' Addresses	9

1. Introduction

This document discusses an extension of the address format defined in [I-D.ietf-behave-address-format]. The original address format document specifies how an individual IPv6 address is translated to a corresponding IPv4 address, and vice versa, in cases where an algorithmic mapping is used. To face the IPv4 public address exhaustion, it is desirable to assign fractional IPv4 addresses to IPv6 nodes which can share a single IPv4 address with each node managing a different range of ports.

In [I-D.ietf-behave-address-format] Section 3.5, it states:

"There have been proposals to complement stateless translation with a port-range feature. Instead of mapping an IPv4 address to exactly one IPv6 prefix, the options would allow several IPv6 nodes to share an IPv4 address, with each node managing a different range of ports. If a port range extension is needed, it could be defined later, using bits currently reserved as null in the suffix."

This document defines one of such a suffix encoding scheme and the corresponding port-range algorithm.

1.1. Applicability Scope

The address format extension can be used for both IPv4/IPv6 translation and IPv4 over IPv6 tunneling. However, in this document we will use IPv4-translatable addresses in the stateless translation to discuss this specific address format extension. The descriptions of the other algorithms for their specific use case can be defined later.

1.2. Conventions

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. Extended IPv4-translatable IPv6 Address

In Section 2.2 of [I-D.ietf-behave-address-format], IPv4-embedded IPv6 address format is defined which composed of a variable length prefix, the embedded IPv4 address, and a variable length suffix, as presented in the following diagram, in which PL designates the prefix length:

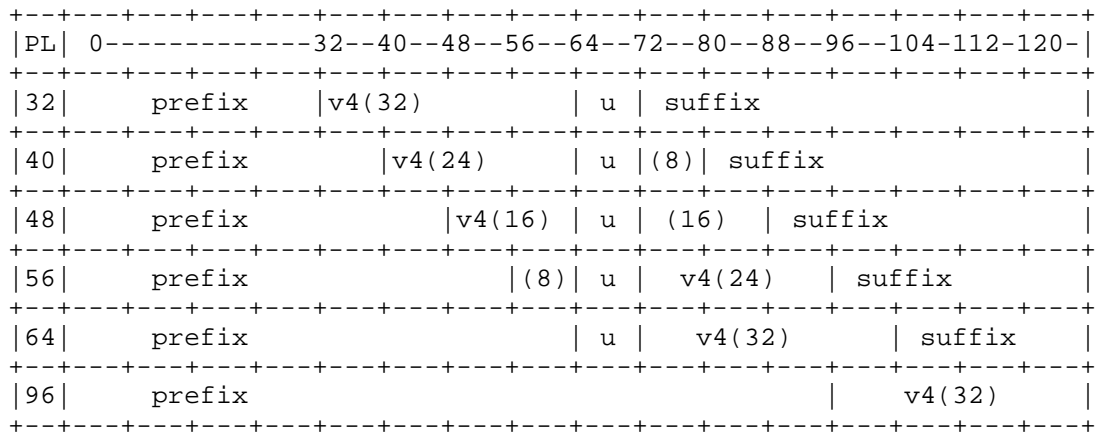


Figure 1: Address Format

It is clear that only Prefix lengths (PL) with 32, 40, 48, 56 and 64 have suffix portions, with maximum suffix lengths of 56, 48, 40, 32, 24, respectively. In order to use different Prefix length and unify the port range encoding method, the suffix should be 24 bits or less. Furthermore, we choose the port-range coding suffix which is directly following the embedded IPv4 address and padding zeros after the suffix.

2.1. Port-range Coding Algorithm

The address-sharing scheme is shown in the following figure.

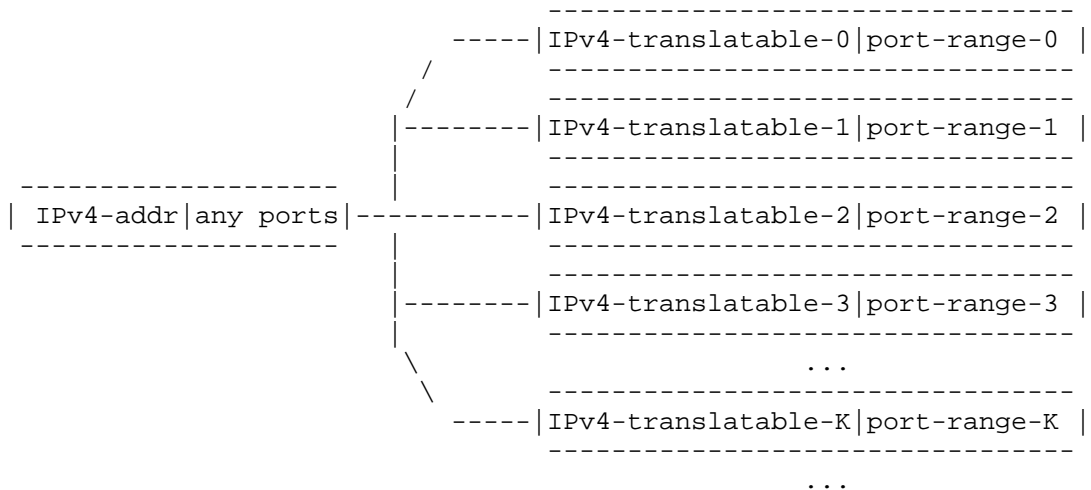


Figure 2: Address Sharing Scheme

In the above figure, the IPv4-translatable-0, IPv4-translatable-1, IPv4-translatable-2, ..., IPv4-translatable-K are sharing the same IPv4 address IPv4-addr, but port number range for different IPv4-translatable addresses (i.e. port-range-0, port-range-1, port-range-2, ..., port-range-K) are not overlapped. When an IPv6 node using IPv4-translatable addresses communicates with the IPv4 Internet via a translator, it looks like a single host using IPv4 address (IPv4-addr) communicating with the IPv4 Internet.

There exist many port-range coding schemes and each one may have its advantages and disadvantages, as well as has its best application scenario. In this document, we will introduce a simple one, which we believe is suitable for the IPv4/IPv6 stateless translation. This encoding scheme uses the modulus operator to define the port number range.

If the sharing ratio is N, then:

- o Given K (K=0, 1, ..., N-1), the allowed port number (P) are $P=j*N + K-1$, where $j=0, 1, \dots, (65536-N)/N$.
- o Given P, the IPv6 node index (K) is $K=(P\%N)$ (% is the Modulus Operator).

For example, If N=128, then IPv6 node K=5 is only allowed to use port numbers 5, 133, 261, 389, 517, 645, 773, 901, ... 65,413 as the source port, while the packets with these port numbers as the destination port number will be send to IPv6 node K=5.

The modulus operator has several features:

1. The port ranges are not overlapped for different IPv6 nodes.
2. The individual ports for each IPv6 node are not continues and the whole 65536 port range is equally shared by IPv6 nodes.
3. The port number range can be uniquely determined by given sharing ratio (N) and the IPv6 node index (K).

Based on the modulus operator, We need to encode N and K in the suffix as an extended IPv4-translatable IPv6 address.

2.2. Extended Address Format

Since the transport port number is a 16 bit integer, the sharing ratio (N) and the IPv6 node index (K) can both have the value from 0

to 65535. In theory, 32 bits (16 bits for sharing ratio and 16 bits for IPv6 node index) are required for encoding the port range based on the modulus operation. In order to fit into 24 bit or less suffix range, we need to do compression.

First, we can use number of bits to represent the sharing ratio when the sharing ratio is bit-wise, hence 4 bits is enough for N.

Secondly, if sharing ratio N is very high, each IPv6 node can only use a small number of concurrent sessions. For example, if $N=4096$, each IPv6 node will have 16 concurrent sessions, which may be too small for most of the applications. Therefore, if we set the maximum sharing ratio $N=4096$, then 12 bits are enough for the IPv6 node index. In this case, we can design suffix which consists of 16 bits for encoding the port range.

Based on the above discussion, the extended address format is shown in the following figure.

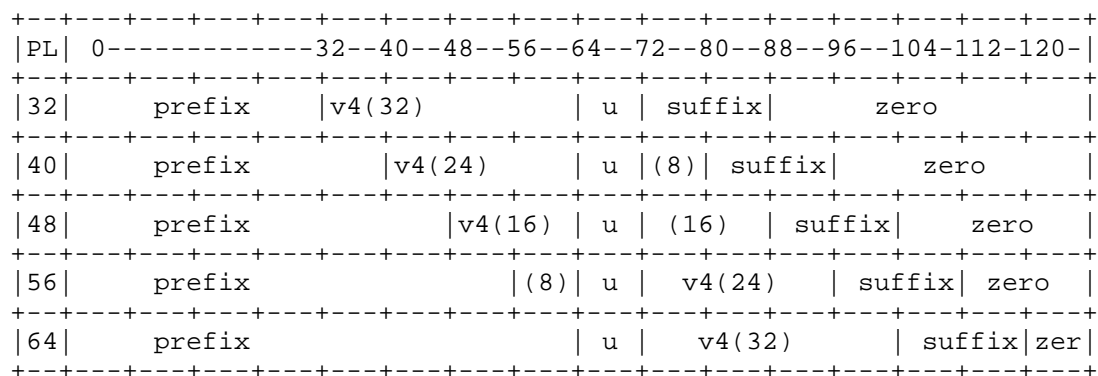


Figure 3: Extended Address Format

Since different suffixes are more specifics in the original address format defined in [I-D.ietf-behave-address-format], the routing considerations in that document are also applied here. Furthermore, the port range is embedded in the extended IPv4-translatable IPv6 addresses and bound to the IPv6 node index (K), therefore the packets containing extended IPv4-translatable IPv6 addresses as the destination can be routed to different IPv6 nodes.

2.3. Suffix for Port Range Encoding

The most significant 4 bits define the multiplexing ratio and the least significant 12 bits define the IPv6 node index. The multiplexing ratio, the suffix range and the number of corresponding

concurrent ports are as shown in the following figure.

ratio	suffix range	# of Ports
1	0000 - 0000	65,536
2	1000 - 1001	32,768
4	2000 - 2003	16,384
8	3000 - 3007	8,192
16	4000 - 400f	4,096
32	5000 - 501f	2,048
64	6000 - 603f	1,024
128	7000 - 707f	512
256	8000 - 80ff	256
512	9000 - 91ff	128
1,024	a000 - a3ff	64
2,048	b000 - b7ff	32
4,096	c000 - cfff	16

Figure 4: Suffix for Port Range Encoding

2.4. Stateless Suffix Translation Algorithm

For the stateless translation, the IPv6 nodes are required to follow the port number range defined by the extended IPv4- translatable address format when communicating with the IPv4 Internet. The port number handling algorithm is:

- o If the packets are from IPv4 to IPv6, the IPv4 source addresses are translated to the IPv4-converted addresses and the source port numbers are unchanged before and after translation; the IPv4 destination addresses are translated to the extended IPv4- translatable addresses based on the destination port number and the destination port numbers are unchanged before and after translation. Note that this means that only a specific IPv6 node can receive the packets for a specific port number. When it is port 80, that specific IPv6 node can setup http redirect service for other IPv6 nodes which also provide web services with non-standard port numbers (e.g. 81, 82, etc.).
- o If the packets are from IPv6 to IPv4, the IPv6 source addresses and the source port numbers are checked, if the source port number matches the port number range defined by the extended IPv4- translatable address format, the IPv6 source addresses (which are the IPv4-translatable addresses) are translated to the IPv4 addresses and the source port numbers are unchanged before and after translation; the destination IPv6 addresses (which are the IPv4-converted addresses) are translated to the IPv4 destination

addresses and the destination port numbers are unchanged before and after translation. However, if the source port number does not match the port number range defined by the extended IPv4-translatable address format, the packets will be dropped.

2.5. Partial-state Suffix Translation Algorithm

Stateless translation requires that IPv6 nodes generate source port number in the range defined by the extended IPv4-translatable address. If this condition does not hold, the partial-state suffix translation algorithm can be used.

The reason we call this partial-state is that:

- o The address mapping is fully algorithm based, as defined in section 3.4. The states are used for port number mapping only.
- o There will be no port mapping table created if the the source port number from IPv6 to IPv4 is in the range defined by the extended IPv4-translatable address.
- o For the destination port number of the packet from the IPv4 to IPv6, there will be no port mapping table created.

The partial-state suffix translation algorithm can be defined later.

2.6. ICMP Packet Handling

The ICMP errors should be translatable using the same algorithm (that is, an error such as Destination Unreachable includes the original TCP (or UDP) header in the ICMP payload, and the that TCP (or UDP) port number can be used to translate the ICMP packet into an IPv6-encoded packet and back again.

Since ICMP echo-request/echo-reply packets only contain identification field, not the transport port numbers, similar to NAT, special actions can be taken for translating the ICMP echo-request/echo-reply packets, which can be defined later.

3. IANA Considerations

This memo adds no new IANA considerations.

4. Security Considerations

There is no special security consideration.

5. Acknowledgements

6. References

6.1. Normative References

[I-D.ietf-behave-address-format]

Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-10 (work in progress), August 2010.

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

6.2. Informative References

[I-D.ietf-behave-v6v4-framework]

Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-10 (work in progress), August 2010.

Authors' Addresses

Congxiao Bao
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China

Phone: +86 10-62785983
Email: congxiao@cernet.edu.cn

Xing Li
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China

Phone: +86 10-62785983
Email: xing@cernet.edu.cn

behave
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2012

C. Bao, Ed.
X. Li
CERNET Center/Tsinghua
University
October 23, 2011

Extended IPv6 Addressing for Encoding Port Range
draft-bcx-address-fmt-extension-02

Abstract

This document discusses an extension of the algorithmic translation between IPv4 and IPv4-translatable IPv6 addresses. The extended address format contains transport-layer port range information which allows several IPv6 nodes to share a single IPv4 address with each node managing a different range of ports. This address format extension can be used for IPv4/IPv6 translation, as well as IPv4 over IPv6 tunneling.

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 October 14, 2011.

Copyright Notice

Copyright (c) 2011 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Applicability Scope	3
1.2. Conventions	3
2. Extended IPv4-translatable IPv6 Address	3
2.1. Port-range Coding Algorithm	4
2.2. Extended Address Format	5
2.3. Suffix for Port Range Encoding	6
2.4. Stateless Suffix Translation Algorithm	7
2.5. Partial-state Suffix Translation Algorithm	8
2.6. ICMP Packet Handling	8
3. IANA Considerations	8
4. Security Considerations	8
5. Acknowledgements	9
6. References	9
6.1. Normative References	9
6.2. Informative References	9
Authors' Addresses	9

1. Introduction

This document discusses an extension of the address format defined in [I-D.ietf-behave-address-format]. The original address format document specifies how an individual IPv6 address is translated to a corresponding IPv4 address, and vice versa, in cases where an algorithmic mapping is used. To face the IPv4 public address exhaustion, it is desirable to assign fractional IPv4 addresses to IPv6 nodes which can share a single IPv4 address with each node managing a different range of ports.

In [I-D.ietf-behave-address-format] Section 3.5, it states:

"There have been proposals to complement stateless translation with a port-range feature. Instead of mapping an IPv4 address to exactly one IPv6 prefix, the options would allow several IPv6 nodes to share an IPv4 address, with each node managing a different range of ports. If a port range extension is needed, it could be defined later, using bits currently reserved as null in the suffix."

This document defines one of such a suffix encoding scheme and the corresponding port-range algorithm.

1.1. Applicability Scope

The address format extension can be used for both IPv4/IPv6 translation and IPv4 over IPv6 tunneling. However, in this document we will use IPv4-translatable addresses in the stateless translation to discuss this specific address format extension. The descriptions of the other algorithms for their specific use case can be defined later.

1.2. Conventions

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. Extended IPv4-translatable IPv6 Address

In Section 2.2 of [I-D.ietf-behave-address-format], IPv4-embedded IPv6 address format is defined which composed of a variable length prefix, the embedded IPv4 address, and a variable length suffix, as presented in the following diagram, in which PL designates the prefix length:

PL	0	32	40	48	56	64	72	80	88	96	104	112	120	
32	prefix	v4(32)					u		suffix					
40	prefix	v4(24)					u	(8)	suffix					
48	prefix	v4(16)					u	(16)	suffix					
56	prefix						(8)	u	v4(24)	suffix				
64	prefix							u	v4(32)	suffix				
96	prefix									v4(32)				

Figure 1: Address Format

It is clear that only Prefix lengths (PL) with 32, 40, 48, 56 and 64 have suffix portions, with maximum suffix lengths of 56, 48, 40, 32, 24, respectively. In order to use different Prefix length and unify the port range encoding method, the suffix should be 24 bits or less. Furthermore, we choose the port-range coding suffix which is directly following the embedded IPv4 address and padding zeros after the suffix.

2.1. Port-range Coding Algorithm

The address-sharing scheme is shown in the following figure.

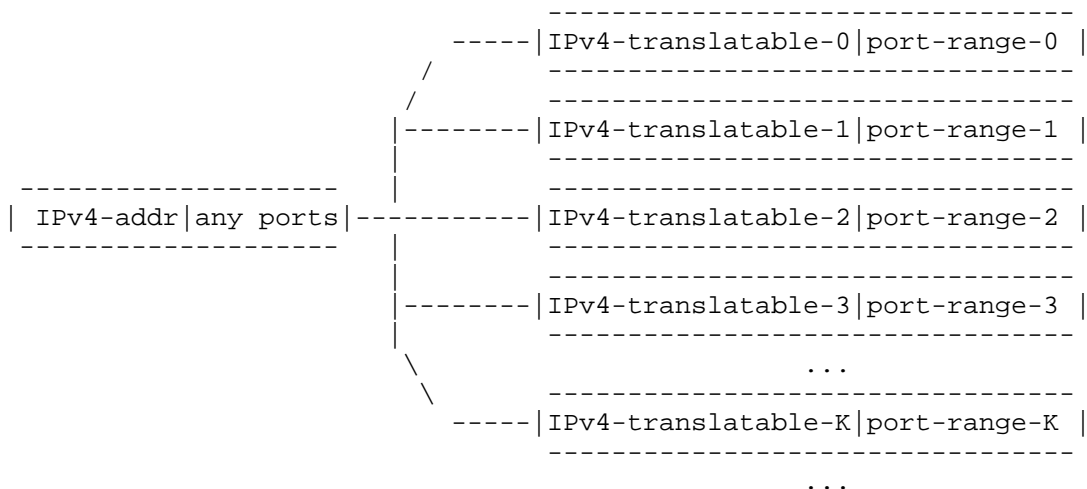


Figure 2: Address Sharing Scheme

In the above figure, the IPv4-translatable-0, IPv4-translatable-1, IPv4-translatable-2, ..., IPv4-translatable-K are sharing the same IPv4 address IPv4-addr, but port number range for different IPv4-translatable addresses (i.e. port-range-0, port-range-1, port-range-2, ..., port-range-K) are not overlapped. When an IPv6 node using IPv4-translatable addresses communicates with the IPv4 Internet via a translator, it looks like a single host using IPv4 address (IPv4-addr) communicating with the IPv4 Internet.

There exist many port-range coding schemes and each one may have its advantages and disadvantages, as well as has its best application scenario. In this document, we will introduce a simple one, which we believe is suitable for the IPv4/IPv6 stateless translation. This encoding scheme uses the modulus operator to define the port number range.

If the sharing ratio is N, then:

- o Given K (K=0, 1, ..., N-1), the allowed port number (P) are $P=j*N + K-1$, where $j=0, 1, \dots, (65536-N)/N$.
- o Given P, the IPv6 node index (K) is $K=(P\%N)$ (% is the Modulus Operator).

For example, If N=128, then IPv6 node K=5 is only allowed to use port numbers 5, 133, 261, 389, 517, 645, 773, 901, ... 65,413 as the source port, while the packets with these port numbers as the destination port number will be send to IPv6 node K=5.

The modulus operator has several features:

1. The port ranges are not overlapped for different IPv6 nodes.
2. The individual ports for each IPv6 node are not continues and the whole 65536 port range is equally shared by IPv6 nodes.
3. The port number range can be uniquely determined by given sharing ratio (N) and the IPv6 node index (K).

Based on the modulus operator, We need to encode N and K in the suffix as an extended IPv4-translatable IPv6 address.

2.2. Extended Address Format

Since the transport port number is a 16 bit integer, the sharing ratio (N) and the IPv6 node index (K) can both have the value from 0

to 65535. In theory, 32 bits (16 bits for sharing ratio and 16 bits for IPv6 node index) are required for encoding the port range based on the modulus operation. In order to fit into 24 bit or less suffix range, we need to do compression.

First, we can use number of bits to represent the sharing ratio when the sharing ratio is bit-wise, hence 4 bits is enough for N.

Secondly, if sharing ratio N is very high, each IPv6 node can only use a small number of concurrent sessions. For example, if N=4096, each IPv6 node will have 16 concurrent sessions, which may be too small for most of the applications. Therefore, if we set the maximum sharing ratio N=4096, then 12 bits are enough for the IPv6 node index. In this case, we can design suffix which consists of 16 bits for encoding the port range.

Based on the above discussion, the extended address format is shown in the following figure.

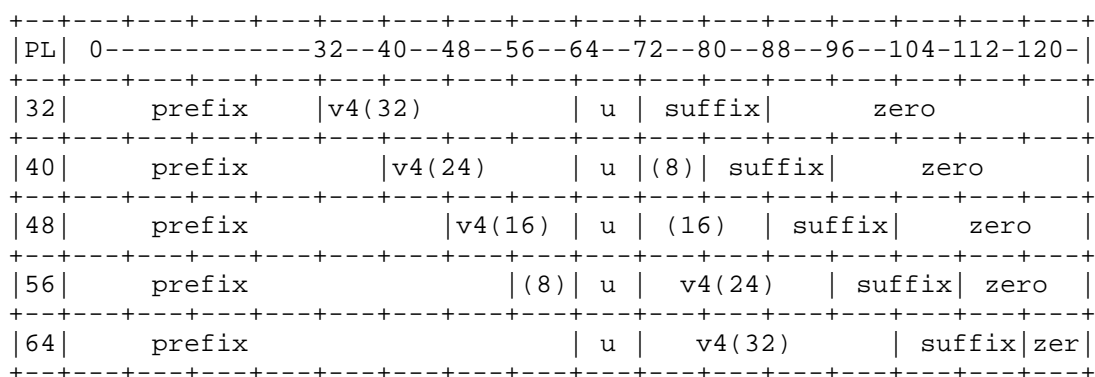


Figure 3: Extended Address Format

Since different suffixes are more specifics in the original address format defined in [I-D.ietf-behave-address-format], the routing considerations in that document are also applied here. Furthermore, the port range is embedded in the extended IPv4-translatable IPv6 addresses and bound to the IPv6 node index (K), therefore the packets containing extended IPv4-translatable IPv6 addresses as the destination can be routed to different IPv6 nodes.

2.3. Suffix for Port Range Encoding

The most significant 4 bits define the multiplexing ratio and the least significant 12 bits define the IPv6 node index. The multiplexing ratio, the suffix range and the number of corresponding

concurrent ports are as shown in the following figure.

ratio	suffix range	# of Ports
1	0000 - 0000	65,536
2	1000 - 1001	32,768
4	2000 - 2003	16,384
8	3000 - 3007	8,192
16	4000 - 400f	4,096
32	5000 - 501f	2,048
64	6000 - 603f	1,024
128	7000 - 707f	512
256	8000 - 80ff	256
512	9000 - 91ff	128
1,024	a000 - a3ff	64
2,048	b000 - b7ff	32
4,096	c000 - cfff	16

Figure 4: Suffix for Port Range Encoding

2.4. Stateless Suffix Translation Algorithm

For the stateless translation, the IPv6 nodes are required to follow the port number range defined by the extended IPv4- translatable address format when communicating with the IPv4 Internet. The port number handling algorithm is:

- o If the packets are from IPv4 to IPv6, the IPv4 source addresses are translated to the IPv4-converted addresses and the source port numbers are unchanged before and after translation; the IPv4 destination addresses are translated to the extended IPv4- translatable addresses based on the destination port number and the destination port numbers are unchanged before and after translation. Note that this means that only a specific IPv6 node can receive the packets for a specific port number. When it is port 80, that specific IPv6 node can setup http redirect service for other IPv6 nodes which also provide web services with non-standard port numbers (e.g. 81, 82, etc.).
- o If the packets are from IPv6 to IPv4, the IPv6 source addresses and the source port numbers are checked, if the source port number matches the port number range defined by the extended IPv4- translatable address format, the IPv6 source addresses (which are the IPv4-translatable addresses) are translated to the IPv4 addresses and the source port numbers are unchanged before and after translation; the destination IPv6 addresses (which are the IPv4-converted addresses) are translated to the IPv4 destination

addresses and the destination port numbers are unchanged before and after translation. However, if the source port number does not match the port number range defined by the extended IPv4-translatable address format, the packets will be dropped.

2.5. Partial-state Suffix Translation Algorithm

Stateless translation requires that IPv6 nodes generate source port number in the range defined by the extended IPv4-translatable address. If this condition does not hold, the partial-state suffix translation algorithm can be used.

The reason we call this partial-state is that:

- o The address mapping is fully algorithm based, as defined in section 3.4. The states are used for port number mapping only.
- o There will be no port mapping table created if the the source port number from IPv6 to IPv4 is in the range defined by the extended IPv4-translatable address.
- o For the destination port number of the packet from the IPv4 to IPv6, there will be no port mapping table created.

The partial-state suffix translation algorithm can be defined later.

2.6. ICMP Packet Handling

The ICMP errors should be translatable using the same algorithm (that is, an error such as Destination Unreachable includes the original TCP (or UDP) header in the ICMP payload, and the that TCP (or UDP) port number can be used to translate the ICMP packet into an IPv6-encoded packet and back again.

Since ICMP echo-request/echo-reply packets only contain identification field, not the transport port numbers, similar to NAT, special actions can be taken for translating the ICMP echo-request/echo-reply packets, which can be defined later.

3. IANA Considerations

This memo adds no new IANA considerations.

4. Security Considerations

There is no special security consideration.

5. Acknowledgements

6. References

6.1. Normative References

[I-D.ietf-behave-address-format]

Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-10 (work in progress), August 2010.

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

6.2. Informative References

[I-D.ietf-behave-v6v4-framework]

Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-10 (work in progress), August 2010.

Authors' Addresses

Congxiao Bao (editor)
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China

Phone: +86 10-62785983
Email: congxiao@cernet.edu.cn

Xing Li
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China

Phone: +86 10-62785983
Email: xing@cernet.edu.cn

Network working group
Internet Draft
Category: Standards Track
Expires: September 10, 2011

D. Cheng
Huawei Technologies

March 11, 2011

NAT44 with Pre-allocated Ports

draft-cheng-behave-nat44-pre-allocated-ports-02

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on September 10, 2011.

Copyright Notice

Copyright (c) 2011 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.

Abstract

This document specifies a NAT44 operation model where external ports are pre-allocated per subscriber. The NAT44 function is deployed in carrier's networks also known as CGN. Three new RADIUS attributes are proposed to support that operation for configuration and billing purpose. Translation logging on a NAT44 device is significantly reduced with this operational model.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

Table of Contents

1. Introduction.....	2
2. Terminology.....	4
3. Operation.....	4
3.1. An Example.....	6
4. RADIUS Attributes.....	7
4.1. Nat-Max-Port-Count Attribute.....	8
4.2. Nat-Alloc-Port-Range Attribute.....	8
4.3. Nat-Alloc-Port-Range Attribute.....	9
5. Table of Attributes.....	11
6. Security.....	11
7. IANA Considerations.....	11
8. Acknowledgements.....	11
9. References.....	11
9.1. Normative References.....	11
9.2. Informative References.....	12
10. Authors' Addresses.....	12

1. Introduction

There are currently several IPv4 address sharing mechanisms such as NAT444([I-D.shirasaki-nat444-isp-shared-addr]) and DS-Lite ([I-D.ietf-softwire-dual-stack-lite]) because of shortage of IPv4 addresses. When an IP flow is initiated from a user side and its source IPv4 address is replaced by a shared IPv4 address at a NAT44 device, the source TCP/UDP port in the IPv4 packet is also replaced by one dynamically allocated by the NAT44 device, most likely from a

single port pool, and as such, the IP flow can be uniquely identified end-to-end. A NAT44 device usually randomizes the port selection, and this practice helps enhance the security in the network.

However, such practice increases the translation logging task on a NAT44 device. Note each NAT44 translation log entry corresponds to a unique IP flow and typically includes the destination address and port, translated source address (the shared IPv4 address), translated source port (the one allocated by the NAT44 device), original source address and port, etc. It requires large volume of storage and also processing capacity on a NAT44 device for such operation (see Section 11 of [I-D.ietf-intarea-shared-addressing-issues]).

This document proposes an operation model for NAT44, where a maximum number of TCP/UDP ports is configured on a RADIUS server for each subscriber as part of the user profile. This information is passed to the NAT44 device where a subscriber is attached. The NAT44 device dynamically pre-allocates a port range for a given subscriber, and the number of ports in that range must be less or equal to the maximum number of ports that has been assigned. When a new IP flow arrives from the subscriber, the NAT44 device then allocates dynamically a port from the pre-allocated port range for the subscriber, also assigns a shared IPv4 address. The NAT44 device would need to pass the pre-allocated port range along with the shared IPv4 address to the RADIUS server and the information may be used for billing purpose. For any contiguous ports that are no longer needed for a subscriber, the NAT44 device would pass the associated port range to the RADIUS server to update the actual port range allocated for the specific user. The communication between the NAT44 device and RADIUS server uses RADIUS protocol ([RFC2865]), requiring NAS co-locates with the NAT44 device.

Note a NAT44 may pre-allocate more than one port ranges for any given subscriber, as long as the total number of ports is less or equal to the maximum number of ports configured on the RADIUS server for that subscriber. The actual port pre-allocation is entirely based on necessity during the operation, e.g., the first or a new port range may be allocated when receiving the first IP packet of a new IP flow sent by a subscriber, but the actual algorithm behind this is out of the scope of this document.

Note also that the new configuration parameter, i.e., the maximum number of ports on a RADIUS server for a subscriber only imposes a limit for that subscriber on the usage of ports, and the actual allocation and de-allocation of any port for that subscriber is

entirely performed by the NAT44 device, as always so today.

This NAT44 port pre-allocation model using RADIUS service could hopefully reduce substantially the intensity on a NAT44 device when performing session based logging, while still able to conduct port allocation with randomization.

To support this operation model, three new RADIUS attributes are defined in this document as follows:

- 1) The Nat-Max-Port-Count attribute carries the maximum number of TCP/UDP ports that a NAT44 device can use for a given subscriber during the translation. The information is configured on a RADIUS server and passed to a NAT44 device.
- 2) The Nat-Alloc-Port-Range attribute carries a block of contiguous ports that a NAT44 device pre-allocates for a given subscriber along with a shared IPv4 address. This information is passed to a RADIUS server from the NAT44 device.
- 3) The Nat-Dealloc-Port-Range attribute carries a block of contiguous ports that a NAT44 device previously allocated for a given subscriber but no longer in use, along with a shared IPv4 address. This information is passed to a RADIUS server from the NAT44 device.

2. Terminology

This document introduces two terms as follows:

. Max port count

This is the maximum number of TCP/UDP ports for a given subscriber, which can be used at a NAT44 device.

. Port range

This specifies a contiguous TCP/UDP ports, indicated by the port with the smallest numerical number and the port with the largest numerical number.

3. Operation

The per-subscriber based maximum port count is configured on a RADIUS server, along with other user information such as credentials. The value of the port count is based on service

agreement and its specification is out of the scope of this document.

A Network Access Server (NAS), located on a NAT44 device, operates as a RADIUS client.

A subscriber initiates a service request, which is sent to the NAT44 device that hosts a NAS, which in turn sends a RADIUS Access-Request message to a RADIUS server. If the server approves the request after validation, it replies with an Access-Accept message back to the NAS, where the message includes a list of parameters for the associated IP session but also the maximum number of ports as defined in this document. While some parameters are passed to the subscriber, the maximum port count for that subscriber is recorded on the NAT44 device.

Upon obtaining the maximum port count for a subscriber, the NAT44 device pre-allocates some ports for the subscriber that are used during NAT44 procedure when receiving IPv4 flows sent from that subscriber. The NAT44 may allocate one or more port ranges, where each range contains a contiguous ports, and the total number of ports must be less or equal to the maximum port count that it records for that subscriber. A NAT44 device may choose to allocate a small range of ports, and allocate more at a later time as needed; such practice is good because its randomization in nature.

At the same time, the NAT44 device also needs to decide the shared IPv4 address for that subscriber. The shared IPv4 address and the pre-allocated port range are then passed to the RADIUS server.

When a subscriber initiates an IPv4 flow, the NAT44 device randomly selects a port from the pre-allocated port range for that subscriber to replace the original source port, along with the replacement of the source IPv4 address by the shared IPv4 address.

At anytime, a NAT44 device may decide to "free" some contiguous ports that have been allocated for a specific subscriber but not currently in use, and with that, the NAT44 device must send the information of the de-allocated port range along with the shared IPv4 address to the RADIUS server.

Figure-1 illustrates how RADIUS protocol is used to configure the maximum number of ports for a given subscriber on a NAT44 device, and obtains the shared IPv4 address and pre-allocated port range determined by the NAT44 device for that subscriber. The NAT44 device later de-allocates a range of ports that no longer used by the subscriber. The NAT44 device sends the

information about pre-allocated port range and de-allocated port range, respectively, to the RADIUS server.

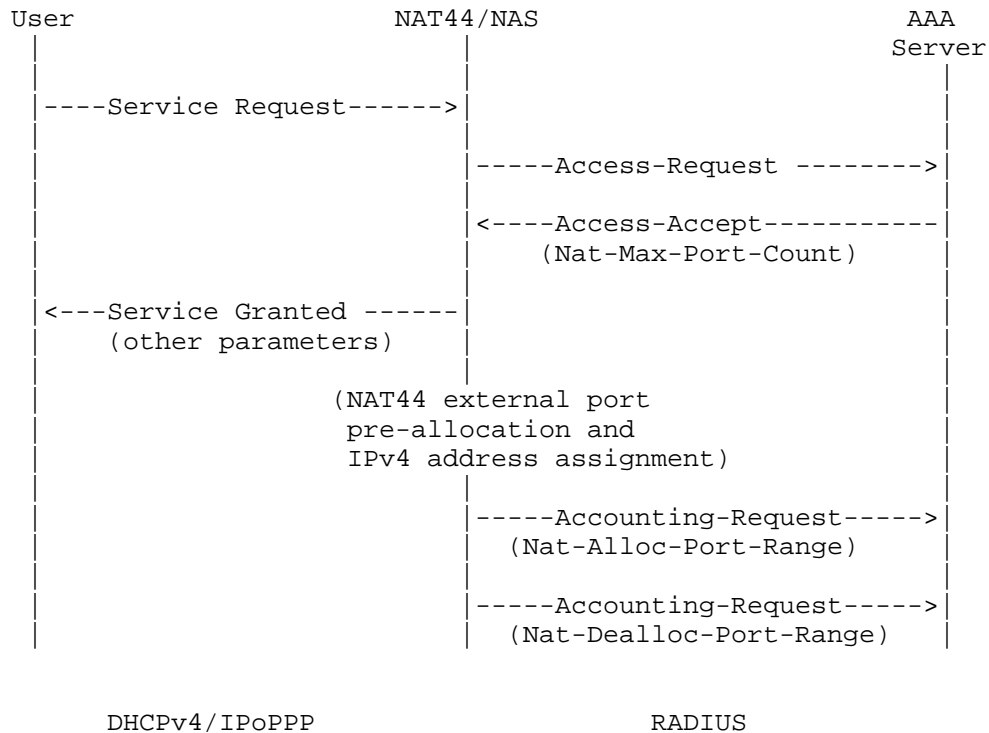


Figure 1: RADIUS Message Flow

3.1. An Example

An ISP assigns 1000 ports for the subscriber A based on a service agreement. This number is the maximum number of ports that can be used during NAT44 on a NAT44 device for A, and is configured on a RADIUS server. When A registers with the ISP's Internet service after required AAA procedure, the RADIUS server passes the maximum port count (1000) to the NAT44 device that is co-located with a BNG that connects A to the Internet.

The NAT44 device decides to pre-allocate a small port pool that contains 40 contiguous ports, from 3500 to 3540, inclusively, and also assign a shared IPv4 address 192.0.2.15, for A.

The NAT44 device passes this port range (3500-3540) and the shared IPv4 address 192.0.2.15 together to the RADIUS server using Nat-Alloc-Port-Range attribute.

When A initiates a new IP flow that reaches the NAT44 device, the original source IP address is replaced by 192.0.2.15, and the source port is replaced by an available one that is randomly selected in the port range, say port 3521.

If at a later time, the port pool (3500-3540) is close to exhaustion, the NAT44 device pre-allocates a second port range in a similar fashion, with the same or different number of ports as in the first one, say from 8500 to 8800, inclusively. The NAT44 then passes this port range (8500-8800) and IPv4 address 192.0.2.15 together to the RADIUS server using Nat-Alloc-Port-Range attribute.

The NAT44 device may decide to de-allocate a pre-allocated port pool (or a sub-range of it) based on any algorithm that is outside of this document, but when that occurs, it needs to send an update to the RADIUS server. In this example, suppose at a later time, all ports in the port pool (8500-8800) are not used and the NAT44 device decides to de-allocate all of them, the NAT44 device sends this port range (8500-8800) along with the IP address 192.0.2.15 together to the RADIUS server using Nat-Dealloc-Port-Range attribute.

4. RADIUS Attributes

Three new RADIUS attributes are defined in this document in order to achieve the NAT44 operational model as described in Section 3.

The Nat-Max-Port-Count attribute carries the maximum number of TCP/UDP ports that a NAT44 device can use during NAT44 procedure for a given subscriber. This maximum number of ports for a given subscriber is configured on a RADIUS server as part of the user's profile, and conveyed to the NAT44 device during the user registration.

The Nat-Alloc-Port-Range attribute carries contiguous TCP/UDP ports that a NAT44 device has pre-allocated to be used during NAT44 procedure for a given subscriber, along with an IPv4 address, which may be shared with other subscribers. This information is sent by the NAT44 device to the RADIUS server, reflecting the actual ports currently allocated for the specific subscriber.

The Nat-Dealloc-Port-Range attribute carries contiguous TCP/UDP ports that a NAT44 device has previously allocated, along with an IPv4 address. All ports in the range are not used anymore by the specific subscriber. This information is sent by the NAT44 device to the RADIUS server to update the information of the actual ports currently allocated for the specific subscriber.

4.1. Nat-Max-Port-Count Attribute

Description

This attribute specifies the maximum number of TCP/UDP ports that is assigned to a NAT44 device corresponding to a specific subscriber for NAT44 operation.

The Nat-Max-Port-Count MAY appear in an Access-Accept packet, and it MAY also appear in an Access-Request packet as a hint by the NAS to the server as a preference, although the RADIUS server is not required to honor the hint.

The Nat-Max-Port-Count MAY appear in an Accounting-Request packet.

The Nat-Max-Port-Count MUST NOT appear in any other RADIUS packets.

0										1										2										3																																							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9																														
Type																																								Length										Max NAT Port Count																			

Type

TBD

Length

4 octets.

Max NAT Port Count

The maximum number of TCP/UDP ports that can be used during NAT44 operation for a given subscriber.

4.2. Nat-Alloc-Port-Range Attribute

Description

This attribute contains a range of contiguous TCP/UDP ports, which is pre-allocated by a NAT44 device for a given subscriber, and an IPv4 address that may be shared with other subscribers.

The Nat-Alloc-Port-Range MAY appear in an Accounting-Request packet.

The Nat-Alloc-Port-Range MUST NOT appear in any other RADIUS packets.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Type      |      Length      | External Port Range Start |
+-----+-----+-----+-----+-----+-----+-----+-----+
| External Port Range End | External IPv4 Address |
+-----+-----+-----+-----+-----+-----+-----+-----+
| External IPv4 Address (cont.) |
+-----+-----+-----+-----+-----+-----+-----+

```

Type

TBD

Length

10 octets.

External Port Range Start

The smallest port number in a range of contiguous TCP/UDP ports.

External Port Range End

The largest port number in a range of contiguous TCP/UDP ports.

External IPv4 Address

The IPv4 address assigned to the associated user to be used in the external realm.

4.3. Nat-Alloc-Port-Range Attribute

Description

This attribute contains a range of contiguous TCP/UDP ports, which was pre-allocated by a NAT44 device for a given subscriber but will no longer be used, along with the IPv4 address previously used together with ports in the range.

The Nat-Dealloc-Port-Range MAY appear in an Accounting-Request packet.

The Nat-Dealloc-Port-Range MUST NOT appear in any other RADIUS packets.

0										1										2										3											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type										Length										External Port Range Start																					
External Port Range End										External IPv4 Address																															
External IPv4 Address (cont.)																																									

Type

TBD

Length

10 octets.

External Port Range Start

The smallest port number in a range of contiguous TCP/UDP ports.

External Port Range End

The largest port number in a range of contiguous TCP/UDP ports.

External IPv4 Address

The IPv4 address assigned to the associated user to be used in the external realm.

5. Table of Attributes

The following table provides a guide as the attributes may be found in which kinds of packets, and in what quantity.

Request	Accept	Reject	Challenge	Accounting #	Attribute
0-1	0-1	0	0	0-1	TBD Nat-Max-Port-Count
0	0	0	0	0+	TBD NAT-Alloc-Port-Range
0	0	0	0	0+	TBD NAT-Dealloc-Port-Range

The meaning of the above table entries is as follows:

0 This attribute MUST NOT be present.

0-1 Zero or one instance of this attribute MAY be present.

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

6. Security

Security problems of the RADIUS protocol are discussed in [RFC2865].

7. IANA Considerations

This document requires the assignment of three new RADIUS attribute numbers for the following attribute:

Nat-Max-Port-Count

Nat-Alloc-Port-Range

Nat-Dealloc-Port-Range

8. Acknowledgements

Many thanks to Dan Wing who provided useful suggestions and comments.

9. References

9.1. Normative References

[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)", RFC2865, June 2000.

9.2. Informative References

[I-D.ietf-softwire-dual-stack-lite] Durand, A., "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", draft-ietf-softwire-dual-stack-lite-07, March 2011.

[I-D.shirasaki-nat444-isp-shared-addr] Shirasaki, Y., Miyakawa, S., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444 addressing models", draft-shirasaki-nat444-isp-shared-addr-05, January 2011.

[I-D.ietf-intarea-shared-addressing-issues] M. Ford, M. Boucadair, A. Durand, P. Levis, P. Roberts, draft-ietf-intarea-shared-addressing-issues-05, March 2011.

10. Authors' Addresses

Dean Cheng
Huawei Technologies,
2330 Central Expressway, CA 95050, USA
Phone: +1 408 330 4754
Email: dean.cheng@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 28, 2011

C. Donley
CableLabs
L. Howard
Time Warner Cable
V. Kuarsingh
Rogers Communications
A. Chandrasekaran
V. Ganti
University of Colorado
October 25, 2010

Assessing the Impact of NAT444 on Network Applications
draft-donley-nat444-impacts-01

Abstract

NAT444 is an IPv4 extension technology being considered by Service Providers to continue offering IPv4 service to customers while transitioning to IPv6. This technology adds an extra Large-Scale NAT ("LSN") in the Service Provider network, often resulting in two NATs. CableLabs, Time Warner Cable, and Rogers Communications independently tested the impacts of NAT444 on many popular Internet services using a variety of test scenarios, network topologies, and vendor equipment. This document identifies areas where adding a second layer of NAT disrupts the communication channel for common Internet applications.

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 24, 2011.

Copyright Notice

Copyright (c) 2010 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. NAT444 Findings	3
2.1. NAT444 Additional Challenges	3
3. Test Cases	4
3.1. Case1: Single Client, Single Home Network, Single Service Provider	4
3.2. Case2: Two Clients, Single Home Network, Single Service Provider	5
3.3. Case3: Two Clients, Two Home Networks, Single Service Provider	6
3.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP	7
4. Summary of Results	7
4.1. Case1: Single Client, Single Home Network, Single Service Provider	8
4.2. Case2: Two Clients, Single Home Network, Single Service Provider	9
4.3. Case3: Two Clients, Two Home Networks, Single Service Provider	9
4.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP	10
5. IANA Considerations	10
6. Security Considerations	10
7. Informative References	10
Appendix A. Acknowledgements	11
Authors' Addresses	11

1. Introduction

Current projections suggest that IANA will exhaust its free pool of IPv4 addresses in 2011. IPv6 is the solution to the IPv4 depletion problem; however, the transition to IPv6 will not be completed prior to IPv4 exhaustion. NAT444 [I-D.shirasaki-nat444] is one transition mechanism that will allow Service Providers to multiplex customers behind a single IPv4 address, which will allow many legacy devices and applications some IPv4 connectivity without requiring a home router upgrade. While NAT444 does provide basic IPv4 connectivity, it breaks a number of advanced applications. This document describes suboptimal behaviors of NAT444 in our test environments.

2. NAT444 Findings

Overall, NAT444 was able to provide IPv4 connectivity for many basic operations conducted by consumers; however, there are several areas of concern with respect to the nested NAT environments. In particular, many advanced tasks (e.g. peer-to-peer seeding, video streaming, some Internet gaming, and IPv6 transition technologies such as 6to4 [RFC3056] and Teredo [RFC4380]) fail outright or are subject to severe service degradation. We observed that performance often differs from vendor to vendor and from test environment to test environment, and the results are somewhat difficult to predict.

2.1. NAT444 Additional Challenges

There are other challenges that arise when using shared IPv4 address space, as with NAT444. Some of these challenges include:

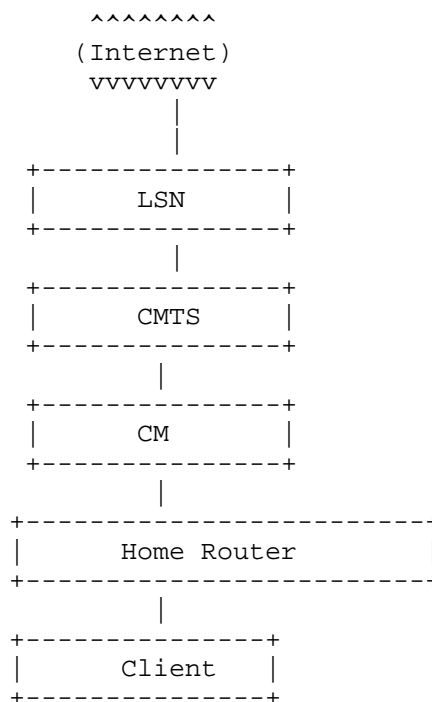
- o Loss of geolocation information - Often, translation zones will cross traditional geographic boundaries. Since the source addresses of packets traversing an LSN are set to the external address of the LSN, it is difficult for external entities to associate IP/Port information to specific locations/areas.
- o Lawful Intercept/Abuse Response - Due to the nature of NAT444 address sharing, it will be hard to determine the customer/endpoint responsible for initiating a specific IPv4 flow based on source IP address alone. Content providers, service providers, and law enforcement agencies will need to use new mechanisms (e.g., logging source port and timestamp in addition to source IP address) to potentially mitigate this new problem. This may impact the timely response to various identification requests. See [I-D.ietf-intarea-shared-addressing-issues]

- o Antispoofing - Multiplexing users behind a single IP address can lead to situations where traffic from that address triggers antispoofing/DDoS protection mechanisms, resulting in unintentional loss of connectivity for some users.

3. Test Cases

The test cases illustrated below are designed to simulate an average home user experience for various combinations of clients behind a single or multiple LSN devices.

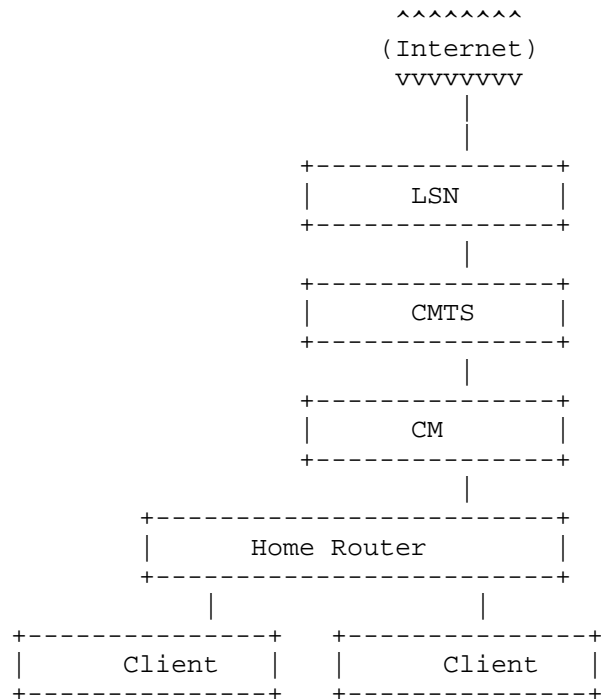
3.1. Case1: Single Client, Single Home Network, Single Service Provider



This is a typical case for a client accessing content on the Internet. For this case, we focused on basic web browsing, voice and video chat, instant messaging, video streaming (using YouTube, Google Videos , etc.), Torrent leeching and seeding, FTP, and gaming. Applications used in this case generally worked better than other topologies. However, Netflix streaming performance was generally slow and erratic. Also, large FTP downloads experienced issues when translation mappings timed out. Bittorrent seeding also failed during some tests. Finally, when a feature on XBOX is used to

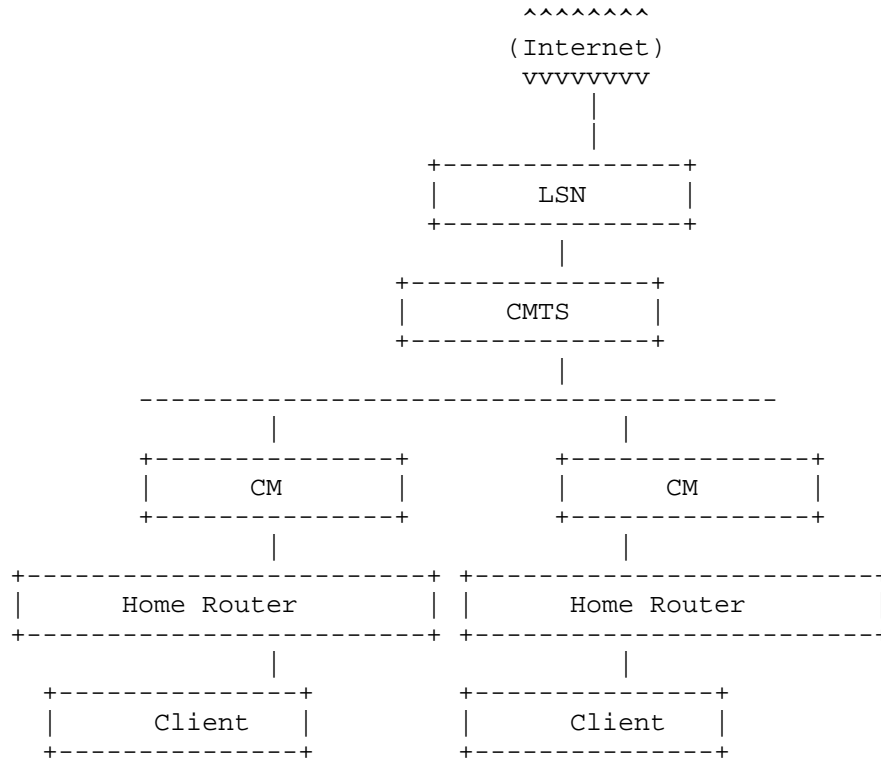
determine the Network Settings, it generates a warning that NAT settings are not ideal and may slow down the experience when more than one client is connected. Gaming generally worked, but had connectivity problems behind one specific LSN platform. Slingcatcher video streaming failed.

3.2. Case2: Two Clients, Single Home Network, Single Service Provider



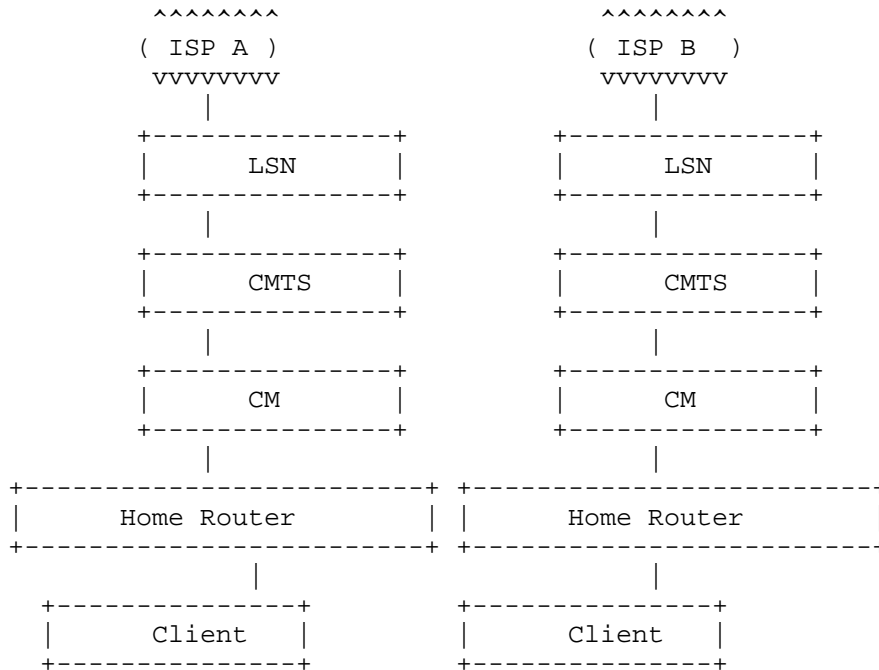
This is similar to Case 1, except that two clients are behind the same LSN and in the same home network. This test case was conducted to observe any change in speed in basic web browsing and video streaming. It is generally noted that the performance decreases in bandwidth intensive applications. Torrent leeching was performed from the two clients to a public server in the Internet. The observed speed was considerably slower than with only one client connected to the home network. Torrent seeding fails. Netflix video streaming is also observed to be considerably choppy. When streaming starts on one client, it does not start on the other, generating a message saying that the Internet connection is too slow.

3.3. Case3: Two Clients, Two Home Networks, Single Service Provider



In this scenario, the two clients are under the same LSN but behind two different gateways. This simulates connectivity between two residential subscribers on the same ISP. We tested peer-to-peer applications. utorrent leeching and limewire leeching passed, while utorrent seeding and limewire seeding failed.

3.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP



This test case is similar to Case 1 but with the addition of another identical ISP. This topology allows us to test traffic between two residential customers connected across the Internet. We focused on client-to-client applications such as IM and peer-to-peer. Instant messaging applications including Skype and Google Talk perform well. Skype video and voice chat also performed well. However, FTP transfers and peer-to-peer seeding failed.

4. Summary of Results

4.1. Case1: Single Client, Single Home Network, Single Service Provider

Test Case	Results	Notes
Web browsing	pass	performance degraded on very large downloads
Email	pass	
FTP download	pass	
Bittorrent leeching	pass	
Bittorrent seeding	fail	Blocked by some LSNs. Your NAT type is moderate. For best online experience you need an open NAT configuration. You should enable UPnP on the router.
Video streaming	pass	
Voice chat	pass	
Netflix streaming	pass	
Instant Messaging	pass	
Ping	pass	
Traceroute	pass	
Remote desktop	pass	
VPN	pass	
Xbox live	pass	
Xbox online	pass	
Xbox network test	fail	
Nintendo Wii	pass behind one LSN, fail behind another	
Playstation 3	pass	
Team Fortress 2	fail	
Starcraft II	pass	
World of Warcraft	pass	
Call of Duty	pass	performance degraded behind one LSN
Slingcatcher	fail	pass behind one LSN
Netflix	fail	
Party (Xbox)		

Hulu	pass	performance degraded behind one LSN
AIM File Tranfer	pass	performance degraded
Webcam	fail	
6to4	fail	
Teredo	fail	

Case1

4.2. Case2: Two Clients, Single Home Network, Single Service Provider

Test Case	Results	Notes
Bittorrent leeching	pass	
Bittorrent seeding	fail	
Video streaming	fail	
Voice chat	pass	
Netflix streaming	pass	performance severely impacted, eventually failed
IM	pass	
Limewire leeching	pass	
Limewire seeding	fail	

Case2

4.3. Case3: Two Clients, Two Home Networks, Single Service Provider

Test Case	Results	Notes
Limewire leeching	pass	
Limewire seeding	fail	
Utorrent leeching	pass	
Utorrent seeding	fail	

Case3

4.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP

Test Case	Results	Notes
Skype voice call	pass	
IM	pass	
FTP	fail	
Facebook chat	pass	
Skype video	pass	

Case4

5. IANA Considerations

This document has no IANA considerations.

6. Security Considerations

Security considerations are described in [I-D.shirasaki-nat444].

7. Informative References

- [I-D.ietf-intarea-shared-addressing-issues]
Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", draft-ietf-intarea-shared-addressing-issues-02 (work in progress), October 2010.
- [I-D.nishitani-cgn]
Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for IP address sharing schemes", draft-nishitani-cgn-05 (work in progress), July 2010.
- [I-D.shirasaki-nat444]
Yamagata, I., Shirasaki, Y., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444", draft-shirasaki-nat444-02 (work in progress), July 2010.
- [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through

Network Address Translations (NATs)", RFC 4380,
February 2006.

Appendix A. Acknowledgements

Thanks to the following people (in alphabetical order) for their
guidance and feedback:

Paul Eldridge

John Berg

Lane Johnson

Authors' Addresses

Chris Donley
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
USA

Email: c.donley@cablelabs.com

Lee Howard
Time Warner Cable
13241 Woodland Park Rd
Herndon, VA 20171
USA

Email: william.howard@twcable.com

Victor Kuarsingh
Rogers Communications
8200 Dixie Road
Brampton, ON L6T 0C1
Canada

Email: victor.kuarsingh@rci.rogers.com

Abishek Chandrasekaran
University of Colorado

Email: abishek.chandrasekaran@colorado.edu

Vivek Ganti
University of Colorado

Email: vivek.ganti@colorado.edu

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 4, 2013

C. Donley, Ed.
CableLabs
L. Howard
Time Warner Cable
V. Kuarsingh
Rogers Communications
J. Berg
CableLabs
J. Doshi
University of Colorado
April 2, 2013

Assessing the Impact of Carrier-Grade NAT on Network Applications
draft-donley-nat444-impacts-06

Abstract

NAT444 is an IPv4 extension technology being considered by Service Providers to continue offering IPv4 service to customers while transitioning to IPv6. This technology adds an extra Carrier-Grade NAT ("CGN") in the Service Provider network, often resulting in two NATs. CableLabs, Time Warner Cable, and Rogers Communications independently tested the impacts of NAT444 on many popular Internet services using a variety of test scenarios, network topologies, and vendor equipment. This document identifies areas where adding a second layer of NAT disrupts the communication channel for common Internet applications. This document was updated to also include Dual-Stack Lite impacts.

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 October 4, 2013.

Copyright Notice

Copyright (c) 2013 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	4
2.	Testing Scope	5
2.1.	Test Cases	5
2.1.1.	Case1: Single Client, Single Home Network, Single Service Provider	5
2.1.2.	Case2: Two Clients, Single Home Network, Single Service Provider	6
2.1.3.	Case3: Two Clients, Two Home Networks, Single Service Provider	7
2.1.4.	Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP	8
2.2.	General Test Environment	8
2.3.	Test Metrics	10
2.4.	Test Scenarios Executed	11
2.5.	General Test Methodologies	11
3.	Observed CGN Impacts	12
3.1.	Dropped Services	13
3.2.	Performance Impacted Services	14
3.3.	Improvements since 2010	15
3.4.	Additional CGN Challenges	16
4.	2011 Summary of Results	16
4.1.	NAT444	17
4.2.	DS-Lite	19
5.	2010 Summary of Results	21
5.1.	Case1: Single Client, Single Home Network, Single Service Provider	22
5.2.	Case2: Two Clients, Single Home Network, Single Service Provider	24
5.3.	Case3: Two Clients, Two Home Networks, Single Service Provider	24
5.4.	Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP	25

6. CGN Mitigation	25
7. IANA Considerations	26
8. Security Considerations	26
9. Informative References	26
Appendix A. Acknowledgements	27
Authors' Addresses	28

1. Introduction

IANA, APNIC, and RIPE exhausted their IPv4 address space in 2011-2012. Current projections suggest that ARIN may exhaust its free pool of IPv4 addresses in 2013. IPv6 is the solution to the IPv4 depletion problem; however, the transition to IPv6 will not be completed prior to IPv4 exhaustion. NAT444 [I-D.shirasaki-nat444] and Dual-Stack Lite ([RFC6333]) are transition mechanisms that will allow Service Providers to multiplex customers behind a single IPv4 address, which will allow many legacy devices and applications some IPv4 connectivity. While both NAT444 and Dual-Stack Lite do provide basic IPv4 connectivity, they impact a number of advanced applications. This document describes suboptimal behaviors of NAT444 and DS-Lite in our test environments.

In July-August 2010, CableLabs, Time Warner Cable, and Rogers Communications tested the impact of NAT444 on common applications using Carrier Grade NAT (CGN) devices. This testing was focused on a wide array of real time usage scenarios designed to evaluate the user experience over the public Internet using NAT444, in both single ISP and dual ISP environments. The purpose of this testing was to identify applications where the technology either breaks or significantly impacts the user experience. The outcome of the testing revealed that applications such as video streaming, video gaming and peer-to-peer file sharing are impacted by NAT444.

From June - October 2011, CableLabs conducted additional testing of CGN technologies, including both NAT444 and Dual-Stack Lite. The testing focused on working with several vendors including A10, Alcatel-Lucent, and Juniper to optimize the performance of those applications that experienced negative impacts during earlier CGN testing and to expand the testing to DS-Lite.

Applications that were tested included but were not necessarily limited to the following:

1. Video/Audio streaming, e.g. Silverlight-based applications, Netflix, YouTube, Pandora 2.
2. Peer-to-peer applications, e.g. video gaming, uTorrent
3. On line gaming, e.g. Xbox
4. Large file transfers using File Transfer Protocol (FTP)
5. Session Initiation Protocol (SIP) calls via X-Lite, Skype

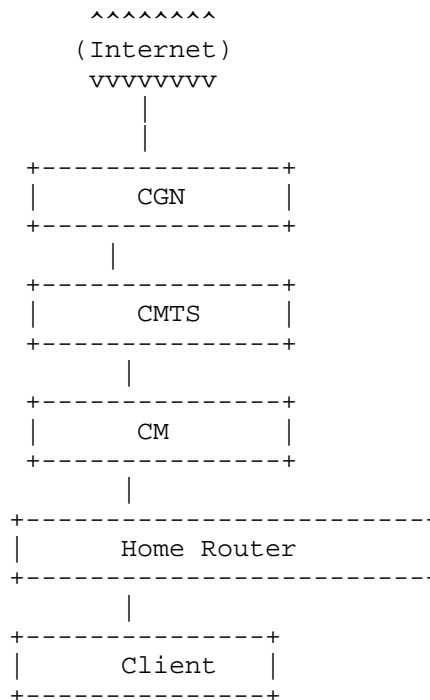
6. Social Networking, e.g. Facebook, Webkinz
7. Video chat, e.g. Skype
8. Web conferencing

2. Testing Scope

2.1. Test Cases

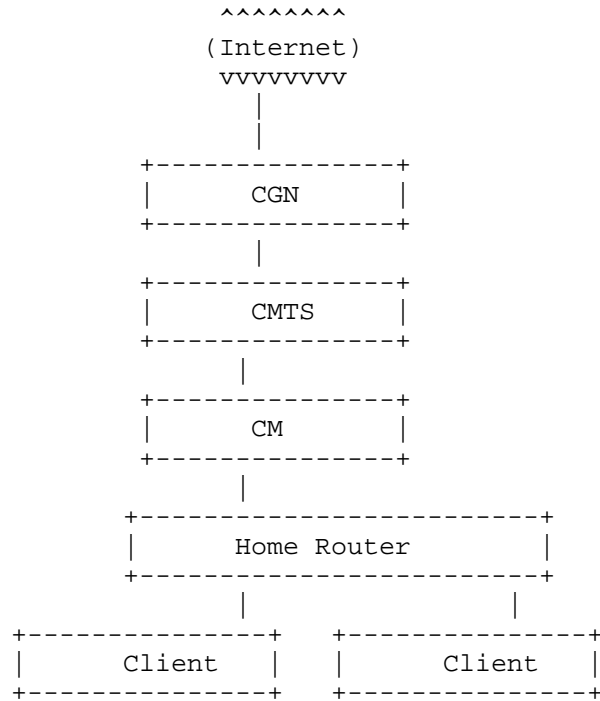
The diagrams below depict the general network architecture used for testing NAT444 and Dual Stack-Lite co-existence technologies at CableLabs.

2.1.1. Case1: Single Client, Single Home Network, Single Service Provider



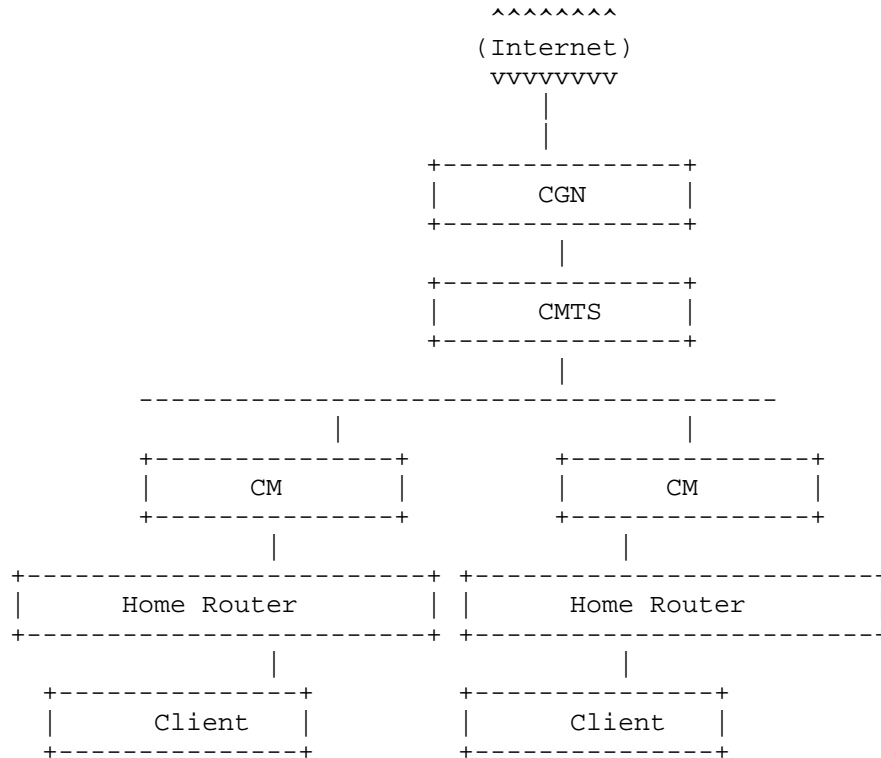
This is a typical case for a client accessing content on the Internet. For this case, we focused on basic web browsing, voice and video chat, instant messaging, video streaming (using YouTube, Google Videos , etc.), Torrent leeching and seeding, FTP, and gaming.

2.1.1.2. Case2: Two Clients, Single Home Network, Single Service Provider



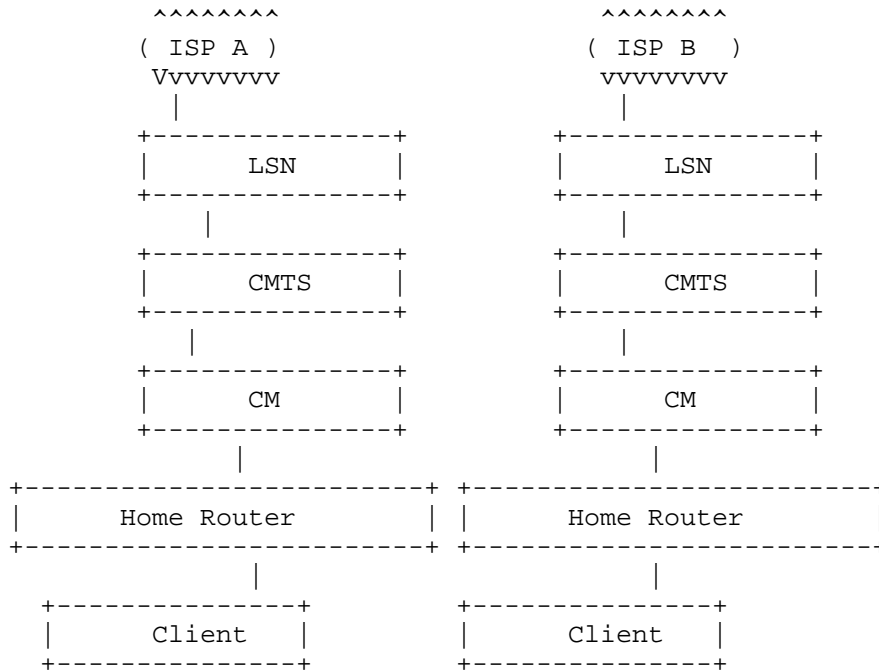
This is similar to Case 1, except that two clients are behind the same LSN and in the same home network. This test case was conducted to observe any change in speed in basic web browsing and video streaming.

2.1.3. Case3: Two Clients, Two Home Networks, Single Service Provider



In this scenario, the two clients are under the same LSN but behind two different gateways. This simulates connectivity between two residential subscribers on the same ISP. We tested peer-to-peer applications.

2.1.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP



This test case is similar to Case 1 but with the addition of another identical ISP. This topology allows us to test traffic between two residential customers connected across the Internet. We focused on client-to-client applications such as IM and peer-to-peer.

2.2. General Test Environment

The lab environment was intended to emulate multiple service provider networks with a CGN deployed, and with connectivity to the public IPv4 or IPv6 internet (as dictated by the co-existence technology under test). This was accomplished by configuring a CGN behind multiple CMTSes and setting up multiple home networks for each ISP. Testing involved sending traffic to and from the public internet in both single and dual ISP environments, using both single and multiple home networks. The following equipment was used for testing:

- o CGN
- o CMTS

- o IP sniffer
- o RF sniffer
- o Metrics tools (for network performance)
- o CPE gateway devices
- o Laptop or desktop computers (multiple OS used)
- o Gaming consoles
- o iPad or tablet devices
- o other CE equipment, e.g. BluRay players supporting miscellaneous applications

One or more CPE gateway devices were configured in the home network. One or more host devices behind the gateways were also configured in order to test conditions such as multiple users on multiple home networks in the CGN architecture, both in single and dual ISP environments.

The scope of testing was honed down to the specific types of applications and network conditions that demonstrated a high probability of diminishing user experience based on prior testing. The following use cases were tested:

1. Video streaming over Netflix
2. Video streaming over YouTube
3. Video streaming over Joost
4. On line gaming with Xbox (one user)
5. Peer to Peer gaming with Xbox (two users)
6. Bit Torrent/uTorrent file seeding/leeching
7. Pandora internet radio
8. FTP server
9. Web conferencing (GTM, WebEx)
10. Social Networking - Facebook, Webkinz (chat, YouTube, file transfer)

11. Internet Archive - Video and Audio streaming; large file downloads
12. Video streaming using iClips
13. SIP Calls - X-Lite, Skype, PJSIP
14. MS Smooth Streaming (Silverlight)
15. Video chat - Skype, OoVoo

The following CPE devices were used for testing these applications on one or more home networks:

1. Windows 7, XP and Vista based laptops
2. MAC OS X laptop
3. iPad
4. Xbox gaming consoles
5. iPhone and Android smartphones
6. LG Blu-Ray player (test applications such as Netflix, Vudu, etc.)
7. Home routers - Netgear, Linksys, D-Link, Cisco, Apple

2.3. Test Metrics

Metrics data that were collected during the course of testing were related to throughput, latency, and jitter. These metrics were evaluated under three conditions:

1. Initial finding on the CGN configuration used for testing
2. Retest of the same test scenario with the CGN removed from the network
3. Retest with a new configuration (optimized) on the CGN (when possible)

In our testing, we found only slight differences with respect to latency or jitter when the CGN was in the network versus when it was not present in the network. It should be noted that we did not conduct any performance testing and metrics gathered were limited to single session scenarios. Also, bandwidth was not restricted on the DOCSIS network. Simulated homes shared a single DOCSIS upstream and

downstream channel.

Case	Avg Latency	Min Latency	Max Latency	RFC4689 Absolute Avg Jitter	Max Jitter
With CGN	240.32 us	233.77 us	428.40 us	1.86 us	191.22 us
Without CGN	211.88 us	190.39 us	402.69 us	0.07 us	176.16 us

CGN Performance

Note: Performance testing as defined by CableLabs includes load testing, induction of impairments on the network, etc. This type of testing was out of scope for CGN testing.

2.4. Test Scenarios Executed

The following test scenarios were executed using the aforementioned applications and test equipment:

1. Single ISP, Single Home Network with Single User
2. Single ISP, Two Home Networks With One User on Each Network
3. Dual ISPs, Single Home Network with Single User on each ISP
4. Dual ISPs, One Home Network With One User ISP-A; Two Home Networks with one user on each for ISP-B

These test scenarios were executed for both NAT444 and DS-Lite technologies.

2.5. General Test Methodologies

The CGN was configured for optimal setting for the specific test being executed for NAT444 or DS-Lite. Individual vendors provided validation of the configuration used for the co-existence technology under test prior to the start of testing. Some NAT444 testing used private [RFC1918] IPv4 space between the CGN and CPE router; other tests used public (non-[RFC1918]) IPv4 space between the CGN and CPE router. With the exception of 6to4 ([RFC3056]) traffic, we observed no difference in test results whether private or public address space was used. 6to4 failed when public space was used between the CGN and

CPE router was public, but CPE routers did not initiate 6to4 when private space was used.

CPE gateways and client devices were configured with IPv4 or IPv6 addresses using DHCP or manual configuration as required by each of the devices used in the test.

All devices were brought to operational state. Connectivity of CPE devices to provider network and public Internet were verified prior to start of each test.

IP sniffers and metrics tools were configured as required before starting tests. IP capture and metrics data was collected for all failed test scenarios. Sniffing was configured behind the home routers, north and south of the CMTS, and north and south of the CGN.

The test technician executed test scenarios listed above, for single and dual ISP environments, testing multiple users on multiple home networks, using the applications described above, where applicable to the each specific test scenario. Results checklists were compiled for all tests executed and for each combination of devices tested.

3. Observed CGN Impacts

CGN testing revealed that basic services such as e-mail and web browsing worked normally and as expected. However, there were some service affecting issues noted for applications that fall into two categories; dropped service and performance impacted service. In addition, for some specific applications in which the performance was impacted, throughput, latency and jitter measurements were taken. We observed that performance often differs from vendor to vendor and from test environment to test environment, and the results are somewhat difficult to predict. So as to not become a comparison between different vendor implementations, these results are presented in summary form. When issues were identified, we worked with the vendors involved to confirm the specific issues and explore workarounds. Except where noted, impacts to NAT444 and DS-Lite were similar.

In 2010 testing, we identified that IPv6 transition technologies such as 6to4 [RFC3056] and Teredo [RFC4380]) fail outright or are subject to severe service degradation. We did not repeat transition technology testing in 2011.

Note: While e-mail and web browsing operated as expected within our environment, there have been reports that anti-spam/anti-abuse measures limiting the number of connections from a single address can

cause problems in a CGN environment by improperly interpreting address sharing as too many connections from a single device. Care should be taken when deploying CGN to mitigate the impact of address sharing when configuring anti-spam/anti-abuse measures. See Section 3.4.

3.1. Dropped Services

Several peer-to-peer applications, specifically peer-to-peer gaming using Xbox and peer-to-peer SIP calls using the PJSIP client, failed in both the NAT444 and Dual-Stack Lite environments. Many CGN devices use "full cone" NAT so that once the CGN maps a port for outbound services, it will accept incoming connections to that port. However, some applications did not first send outgoing traffic and hence did not open an incoming port through the CGN. Other applications try to open a particular fixed port through the CGN; while service will work for a single subscriber behind the CGN, it fails when multiple subscribers try to use that port.

PJSIP and other SIP software worked when clients used a registration server to initiate calls, provided that the client inside the CGN initiated the traffic first and that only one SIP user was active behind a single IPv4 address at any given time. However, in our testing, we observed that when making a direct client-to-client SIP call across two home networks on a single ISP, or when calling from a single home network across dual ISPs, calls could neither be initiated nor received.

In the case of peer-to-peer gaming between two Xbox 360 users in different home networks on the same ISP, the game could not be connected between the two users. Both users shared an outside IP address, and tried to connect to the same port, causing a connection failure. There are some interesting nuances to this problem. In the case where two users are in the same home network and the scenario is through a single ISP, when the Xbox tries to register with the Xbox server, the server sees that both Xboxes are coming through the same public IP address and directs the devices to connect using their internal IP addresses. So, the connection ultimately gets established directly between both Xboxes via the home gateway, rather than the Xbox server. In the case where there are two Xbox users on two different home networks using a single ISP, and the CGN is configured with only one public IPv4 address, this scenario will not work because the route between the two users cannot be determined. However, if the CGN is configured with two public NAT IP addresses this scenario will work because now there is a unique IP address to communicate with. This is not an ideal solution, however, because it means that there is a one-to-one relationship between IP addresses in the public NAT and the number of Xbox users on each network.

Update: in December, 2011, Microsoft released an update for Xbox. While we did not conduct thorough testing using the new release, preliminary testing indicates that Xboxes that upgraded to the latest version can play head-to-head behind a CGN, at least for some games.

Other peer-to-peer applications that were noted to fail were seeding sessions initiated on Bittorrent and uTorrent. In our test, torrent seeding was initiated on a client inside the CGN. Leeching was initiated using a client on the public Internet. It was observed that direct peer-to-peer seeding did not work. However, the torrent session typically redirected the leeching client to a proxy server, in which case the torrent session was set up successfully. Additionally, with the proxy in the network, re-seeding via additional leech clients worked as would be expected for a typical torrent session. Finally, uTorrent tries to use STUN to identify its outside address. In working with vendors, we learned that increasing the STUN timeout to 4 minutes improved uTorrent seeding performance behind a CGN, resulting in the ability for the uTorrent client to open a port and successfully seed content.

FTP sessions to servers located inside the home (e.g. behind two layers of NAT) failed. When the CGN was bypassed and traffic only needed to flow through one layer of NAT, clients were able to connect. Finally, multicast traffic was not forwarded through the CGN.

3.2. Performance Impacted Services

Large size file transfers and multiple video streaming sessions initiated on a single client on the same home network behind the CGN experienced reduced performance in our environment. We measured these variations in user experience against a baseline IPv4 environment where NAT is not deployed.

In our testing, we tried large file transfers from several FTP sites, as well as downloading sizable audio and video files (750MB - 1.4 GB) from the Internet Archive. We observed that when Dual-Stack Lite was implemented for some specific home router and client combinations, the transfer rate was markedly slower. For example, PC1 using one operating system behind the same home router as PC2 using a different operating system yielded a transfer rate of 120Kbps for PC1, versus 250Kbps for PC2. Our conclusion is that varying combinations of home routers and CE client devices may result in a user experience that is less than what the user would expect for typical applications. It is also difficult to predict which combinations of CPE routers and CE devices will produce a reduced experience for the user. We did not analyze the root cause of the divergence in performance across CE devices, as this was beyond the scope of our testing. However, as

this issue was specific to Dual-Stack Lite, we suspect that it is related to the MTU.

While video streaming sessions for a single user generally performed well, testing revealed that video streaming sessions such as Microsoft Smooth Streaming technology (i.e. Silverlight) or Netflix might also exhibit some service impacting behavior. In particular, this was observed on one older, yet popular and well-known CPE router where the first session was severely degraded when a second session was initiated in the same home network. Traffic from the first session ceased for 8 s once the second session was initiated. While we are tempted to write this off as a problematic home router, its popularity suggests that home router interactions may cause issues in NAT444 deployments (newer routers that support DS-Lite were not observed to experience this condition). Overall, longer buffering times for video sessions were noted for most client devices behind all types of home routers. However, once the initial buffering was complete, the video streams were consistently smooth. In addition, there were varying degrees as to how well multiple video sessions were displayed on various client devices across the CPE routers tested. Some video playback devices performed better than others.

3.3. Improvements since 2010

Since CableLabs completed initial CGN testing in 2010, there have been quantifiable improvements in performance over CGN since that time. These improvements may be categorized as follows:

- o Content provider updates
- o Application updates
- o Improvements on the CGNs themselves

In terms of content provider updates, we have noted improvements in the overall performance of streaming applications in the CGN environment. Whereas applications such as streaming video were very problematic a year ago with regard to jitter and latency, our most recent testing revealed that there is less of an issue with these conditions, except in some cases when multiple video streaming sessions were initiated on the same client using specific types of home routers. Applications such as MS Smooth Streaming appear to have addressed these issues to some degree.

As far as application updates, use of STUN and/or proxy servers to offset some of the limitations of NAT and tunneling in the network are more evident as workarounds to the peer-to-peer issues. Applications appear to have incorporated other mechanisms for

delivering content faster, even if buffering times are somewhat slower and the content is not rendered as quickly.

CGN vendors have also upgraded their devices to mitigate several known issues with specific applications. With regard to addressing peer-to-peer SIP call applications, port reservations appear to be a workaround to the problem. However, this approach has limitations because of there are limited numbers of users that can have port reservations at any given time. For example, one CGN implementation allowed a port reservation to be made on port 5060 (default SIP port) but this was the only port that could be configured for the SIP client. This means that only one user can be granted the port reservation.

3.4. Additional CGN Challenges

There are other challenges that arise when using shared IPv4 address space, as with NAT444. Some of these challenges include:

- o Loss of geolocation information - Often, translation zones will cross traditional geographic boundaries. Since the source addresses of packets traversing an LSN are set to the external address of the LSN, it is difficult for external entities to associate IP/Port information to specific locations/areas.
- o Lawful Intercept/Abuse Response - Due to the nature of NAT444 address sharing, it will be hard to determine the customer/endpoint responsible for initiating a specific IPv4 flow based on source IP address alone. Content providers, service providers, and law enforcement agencies will need to use new mechanisms (e.g., logging source port and timestamp in addition to source IP address) to potentially mitigate this new problem. This may impact the timely response to various identification requests. See [RFC6269].
- o Antispoofing - Multiplexing users behind a single IP address can lead to situations where traffic from that address triggers antispoofing/DDoS protection mechanisms, resulting in unintentional loss of connectivity for some users. We have received reports of such antispoofing/DDoS mechanisms affecting email and web services in some instances, but did not experience them in our environment.

4. 2011 Summary of Results

4.1. NAT444

Test Scenario (per Test Plan)	Single ISP, Single HN, Single User	Single ISP, Two HN, Single User on Each	Dual ISP, One HN with One User on Each ISP	Dual ISP, One HN+One User on ISP-A, Two HN with One User on Each on ISP-B	Notes
Video streaming over Netflix	Pass	Pass	Pass	Pass	fails behind one router
Video streaming over YouTube	Pass	Pass	Pass	Pass	
Video streaming over Joost	Pass	Pass	Pass	Pass	
Online gaming with one user	Pass	Pass	Pass	NT	
Peer to Peer gaming with two users	Pass	Fail	Pass	NT	fails when both users NAT to same address
Bit Torrent uTorrent file seeding	Fail	Fail	Fail	Fail	
Bit Torrent uTorrent file leeching	Pass	Pass	Pass	Pass	

Pandora internet radio	Pass	Pass	Pass	Pass	
FTP server	Pass	Pass	Pass	Pass	
Web conferencing GTM	Pass	Pass	Pass	Pass	
Social Networking Facebook	Pass	Pass	Pass	Pass	
Social Networking Webkinz	Pass	Pass	Pass	Pass	
X-Lite for SIP calls with proxy	Pass	Pass	Pass	Pass	
X-Lite for SIP calls no proxy	Fail	Fail	Fail	Fail	
Skype text chat	Pass	Pass	Pass	Pass	
Skype video chat	Pass	Pass	Pass	Pass	
Oovoo	Pass	Pass	Pass	Pass	
MS Smooth streaming	Pass	Pass	Pass	Pass	
Internet Archive video streaming	Pass	Pass	Pass	Pass	
Internet Archive audio streaming	Pass	Pass	Pass	Pass	

Internet Archive file download	Pass	Pass	Pass	Pass	
Iclips	Pass	Pass	Pass	Pass	

NAT-444

4.2. DS-Lite

Test Scenario (per Test Plan)	DS-Lite Test Results	Duration of Test Performed	Description of Test Execution	General Observations/Notes
Video streaming over Netflix	Pass	15		
Video streaming over YouTube	Pass	10		
Video streaming over Joost	Pass	10		
On line gaming (one user)	Pass	15		
Peer to Peer gaming (two users)	Fail	NA	user inside HN1 playing game against user inside HN2	Users inside both HN are not able to connect. The error shown on console- "The game session is no longer available"

Bit Torrent/uTorr ent file seeding	Fail	12	user on the internet is able to download file using proxy server and not peer-to-pee r	
Bit Torrent/uTorr ent file leeching	Pass	10		
Pandora internet radio	Pass	10		
FTP server	Pass	700 Mb		
Web conferencing (GTM)	Pass	10		
Social Networking - Facebook	Pass	NA		
Social Networking - Webkinz	Pass	NA		
X-Lite (for SIP calls) (proxy given)	Pass	10		
X-Lite (for SIP calls) (proxy not given)	Fail	NA		
Skype text chat	Pass	NA		

Skype video chat	Pass	20		
Oovoo	Pass	15		
MS Smooth streaming	Pass	10		
Internet Archive - video streaming	Pass	10		
Internet Archive - audio streaming	Pass	5		
Internet Archive - file download	Pass	80 Mb		
Iclips	Pass	10		

DSLite

5. 2010 Summary of Results

The tables below summarize results from 2010 NAT444 testing at CableLabs, Time Warner Cable, and Rogers Communications. They are included for comparison with 2011 results, documented above.

5.1. Case1: Single Client, Single Home Network, Single Service Provider

Test Case	Results	Notes
Web browsing	pass	
Email	pass	
FTP download	pass	performance degraded on very large downloads
Bittorrent leeching	pass	
Bittorrent seeding	fail	
Video streaming	pass	
Voice chat	pass	
Netflix streaming	pass	
Instant Messaging	pass	
Ping	pass	
Traceroute	pass	
Remote desktop	pass	
VPN	pass	
Xbox live	pass	
Xbox online	pass	Blocked by some LSNs.
Xbox network test	fail	Your NAT type is moderate. For best online experience you need an open NAT configuration. You should enable UPnP on the router.

Nintendo Wii	pass behind one LSN, fail behind another	
Playstation 3	pass	
Team Fortress 2	fail	pass behind one LSN, but performance degraded
Starcraft II	pass	
World of Warcraft	pass	
Call of Duty	pass	performance degraded behind one LSN
Slingcatcher	fail	
Netflix Party (Xbox)	fail	pass behind one LSN
Hulu	pass	performance degraded behind one LSN
AIM File Tranfer	pass	performance degraded
Webcam	fail	
6to4	fail	
Teredo	fail	

Case1

5.2. Case2: Two Clients, Single Home Network, Single Service Provider

Test Case	Results	Notes
Bittorrent leeching	pass	
Bittorrent seeding	fail	
Video streaming	fail	
Voice chat	pass	
Netflix streaming	pass	performance severely impacted, eventually failed
IM	pass	
Limewire leeching	pass	
Limewire seeding	fail	

Case2

5.3. Case3: Two Clients, Two Home Networks, Single Service Provider

Test Case	Results	Notes
Limewire leeching	pass	
Limewire seeding	fail	
Utorrent leeching	pass	
Utorrent seeding	fail	

Case3

5.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP

Test Case	Results	Notes
Skype voice call	pass	
IM	pass	
FTP	fail	
Facebook chat	pass	
Skype video	pass	

Case4

6. CGN Mitigation

Our testing did not focus on mitigating the impact of Carrier Grade NAT, as described above. As such, mitigation is not the focus of this document. However, there are several approaches that could lessen the impacts described above.

Challenge	Potential Workaround(s)
Peer-to-peer	Use a proxy server; [I-D.ietf-pcp-base]
Gaming	[I-D.ietf-pcp-base]
Negative impact to geo-location services	Deploy CGN close to the edge of the network; use regional IP and port assignments.
Logging requirements for lawful intercept	Deterministic Logging [I-D.donley-behave-deterministic-cgn]; data compression [I-D.sivakumar-behave-nat-logging]; bulk port logging

CGN mitigation

Other mitigation techniques that are currently being researched, such as [I-D.tsou-stateless-nat44], may also improve performance.

7. IANA Considerations

This document has no IANA considerations.

8. Security Considerations

Security considerations are described in [RFC6264] and [RFC6269].

In general, since a CGN device shares a single IPv4 address with multiple subscribers, CGN devices may provide an attractive target for denial of service attacks. In addition, as described in [I-D.donley-behave-deterministic-cgn], abuse attribution is more challenging with CGN, and requires content providers to log IP address, source port, and time to correlate with service provider CGN logs. Also, if a CGN public IP address is added to a blacklist (e.g. for SPAM) or if a server limits the number of connections per IP address, it could negatively impact legitimate users.

9. Informative References

[I-D.donley-behave-deterministic-cgn]

Donley, C., Grundemann, C., Sarawat, V., Sundaresan, K., and O. Vautrin, "Deterministic Address Mapping to Reduce Logging in Carrier Grade NAT Deployments", draft-donley-behave-deterministic-cgn-05 (work in progress), January 2013.

[I-D.ietf-pcp-base]

Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", draft-ietf-pcp-base-29 (work in progress), November 2012.

[I-D.shirasaki-nat444]

Yamagata, I., Shirasaki, Y., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444", draft-shirasaki-nat444-02 (work in progress), July 2010.

[I-D.sivakumar-behave-nat-logging]

Sivakumar, S. and R. Penno, "IPFIX Information Elements for logging NAT Events", draft-sivakumar-behave-nat-logging-06 (work in progress), January 2013.

[I-D.tsou-stateless-nat44]

Tsou, T., Liu, W., Perreault, S., Penno, R., and M. Chen, "Stateless IPv4 Network Address Translation",

draft-tsou-stateless-nat44-02 (work in progress),
October 2012.

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.
- [RFC4689] Poretsky, S., Perser, J., Erramilli, S., and S. Khurana, "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, October 2006.
- [RFC6264] Jiang, S., Guo, D., and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", RFC 6264, June 2011.
- [RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

Appendix A. Acknowledgements

Thanks to the following people for their testing, guidance, and feedback:

Paul Eldridge

Abishek Chandrasekaran

Vivek Ganti

Joey Padden

Lane Johnson

Authors' Addresses

Chris Donley (editor)
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
USA

Email: c.donley@cablelabs.com

Lee Howard
Time Warner Cable
13241 Woodland Park Rd
Herndon, VA 20171
USA

Email: william.howard@twcable.com

Victor Kuarsingh
Rogers Communications
8200 Dixie Road
Brampton, ON L6T 0C1
Canada

Email: victor.kuarsingh@rci.rogers.com

John Berg
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
USA

Email: j.berg@cablelabs.com

Jinesh Doshi
University of Colorado

Email: jinesh.doshi@colorado.edu

Internet Engineering Task Force
Internet-Draft
Intended status: BCP
Expires: April 21, 2011

A. Durand
Juniper Networks
I. Gashinsky
Yahoo! Inc.
D. Lee
Facebook, Inc.
S. Sheppard
ATT Labs
October 18, 2010

Logging recommendations for Internet facing servers
draft-durand-server-logging-recommendations-00

Abstract

In the wake of IPv4 exhaustion and deployment of IP address sharing techniques, this document recommends that Internet facing servers log port number and accurate timestamps in addition to the incoming IP address.

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, 2011.

Copyright Notice

Copyright (c) 2010 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. Recommendations	3
3. ISP Considerations	4
4. IANA Considerations	4
5. Security Considerations	4
6. References	5
6.1. Normative references	5
6.2. Informative references	5
Authors' Addresses	5

1. Introduction

According to the most recent predictions, the global IPv4 address free pool at IANA will exhaust sometime in 2011. After that, service providers will have a hard time finding enough IPv4 global addresses to sustain product and subscriber growth. Due to the huge global existing infrastructure, both hardware and software, vendors and service providers must continue to support IPv4 technologies for the foreseeable future. As legacy applications and hardware are retired the reliance on IPv4 will diminish but this is a years long perhaps decades long process.

To maintain legacy IPv4 address support, service providers will have little choice but to share IPv4 global addresses among multiple customers. Techniques to do so are outside of the scope of this documents. All include some form of address translation/address sharing, being NAT44, NAT64 or DS-Lite.

The effects on the Internet of the introduction of those address sharing techniques have been documented in [I-D.ietf-intarea-shared-addressing-issues].

Address sharing techniques come with their own logging infrastructure to track the relation between which original IP address and source port(s) were associated with which user and external IPv4 address at any given point in time. In the past to support abuse mitigation or public safety requests, the knowledge of the external global IP address was enough to identify a subscriber of interest. With address sharing technologies, only providing information about the external public address associated with a session to a service provider is no longer sufficient information to unambiguously identify customers.

Note: this document provides recommendations for Internet facing servers logging incoming connections. Its does not provide any recommendations about logging on carrier-grade NAT or other address sharing tools.

2. Recommendations

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

It is RECOMMENDED as best current practice that Internet facing servers logging incoming IP addresses also log:

- o The source port number.
- o A timestamp accurate to the second, with associated time zone.
- o The transport protocol (usually TCP or UDP) and destination port number, when the server application is defined to use multiple transports or multiple ports.

Discussion: Carrier-grade NATs may have different policies to recycle ports, some implementations may decide to reuse ports almost immediately, some may wait several minutes before marking the port ready for reuse. As a result, servers have no idea how fast the ports will be reused and, thus, should log timestamps using a reasonably accurate clock. At this point the RECOMMENDED accuracy for timestamps is to the second or better.

Examples of Internet facing servers include, but are not limited to, web servers and email servers.

Although the deployment of address sharing techniques is not immediately foreseen in IPv6, the above recommendations apply to both IPv4 and IPv6, if only for consistency and code simplification reasons.

Discussions about data retention policies are out of scope for this document.

3. ISP Considerations

ISP deploying IP address sharing techniques should also deploy a corresponding logging architecture to maintain records of the relation between customers identity and IP/port resources they utilize. However, recommendation on this topic are out of scope for this document.

4. IANA Considerations

None.

5. Security Considerations

In the absence of source port number and accurate timestamp, operators deploying any address sharing techniques will not be able to identify unambiguously customers when dealing with abuse or public safety queries.

6. References

6.1. Normative references

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

6.2. Informative references

[I-D.ietf-intarea-shared-addressing-issues]
Ford, M., Boucadair, M., Durand, A., Levis, P., and P.
Roberts, "Issues with IP Address Sharing",
draft-ietf-intarea-shared-addressing-issues-02 (work in
progress), October 2010.

Authors' Addresses

Alain Durand
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089-1206
USA

Email: adurand@juniper.net

Igor Gashinsky
Yahoo! Inc.
45 West 18th St.
New York, NY 10011
USA

Email: igor@yahoo-inc.com

Donn Lee
Facebook, Inc.
1601 S. California Ave.
Palo Alto, CA 94304
USA

Email: donn@facebook.com

Scott Sheppard
ATT Labs
575 Morosgo Ave, 4d57
Atlanta, GA 30324
USA

Email: Scott.Sheppard@att.com

Behave	A. Hamarsheh
Internet-Draft	ETRO/Vrije Universiteit Brussel
Obsoletes: 3338 (if approved)	M. Goossens
Intended status: Experimental	ETRO/Vrije Universiteit Brussel
Expires: March 14, 2011	September 10, 2010

Hosts with Any Network Connectivity Using "Bump-in-the-API" (BIA)
draft-hamarsheh-behave-biav2-03

Abstract

This document specifies a mechanism for hosts with any network connectivity (IPv4 only, IPv6 only, or dual IPv4/IPv6 connectivity) to run applications of any capability (IPv4 only, IPv6 only, or dual IPv4/IPv6) without any modification to those applications. It is a generalisation of a previous experimental protocol called "Bump-in-the-API" (BIA) [RFC3338]. New mechanism of BIA allows a changeover between the application layer and the IP communication layers from IPv4 to IPv6 and vice versa or IPv6 to IPv4 and vice versa, without requiring those applications to be converted in addressing capabilities, effectively shielding the application layer from IPv4 or IPv6 connectivity. This is considered by the authors to be one of the essential conditions for the transition to IPv6 in the Internet to be successful.

Status of this Memo

This Internet-Draft is submitted to IETF 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 March 14, 2011.

Copyright Notice

Copyright (c) 2010 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 BSD License.

Table of Contents:

1. Motivation and Introduction	4
1.1 Motivation	4
1.2 Introduction	5
2. Applicability and Related Techniques	6
2.1 Applicability	6
2.2 Related Techniques	7
3. Host Configurations Using BIA	8
3.1 IPv6 Only Host Using BIA	9
3.2 IPv4 Only Host Using BIA	9
3.3 Dual Connectivity Host Using BIA	10
4. BIA Modules	11
4.1 Name Resolver	11
4.1.1 IPv4 Only Application on The Local Host	11
4.1.2 IPv6 Only Application on The Local Host	12
4.1.3 Reverse DNS Lookup	13
4.1.4 Originating Without DNS Lookup	13
4.2 Address Resolver	14
4.2.1 Mapping	14
4.2.2 Embedding	15
4.3 Function Mapper	15
5. Behavior Examples	16
5.1 IPv4 Only Application, IPv6 Only Connectivity with an IPv6 Only Peer	16
5.1.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating to IPv6 Only Peer	16
5.1.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host	18
5.2 IPv4 only Application, IPv6 Only Network and Dual Connectivity Peer	19

5.2.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating with Dual Connectivity Host ..	19
5.2.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host Communicating with Dual Connectivity Host ..	20
5.3 IPv6 Only Application, IPv4 Only Connectivity with an IPv4 Only Peer	20
5.3.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with IPv4 Only Host	20
5.3.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with IPv4 Only Host	21
5.4 IPv6 Only Application, IPv4 Only Network and Dual Connectivity Peer	22
5.4.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with Dual Connectivity Host ..	22
5.4.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with Dual Connectivity Host ..	23
5.5 IPv4 Only Application on Dual Connectivity Host Communicating to IPv6 Only Peer	23
5.5.1 IPv4 Only Originator Application on Dual Connectivity Host Communicating to IPv6 Only Peer	23
5.5.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv6 Only Peer	23
5.6 IPv6 Only Application on Dual Connectivity Host Communicating to IPv4 Only Peer	23
5.6.1 IPv6 Only Originator Application on Dual Connectivity Host Communicating to IPv4 Only Peer	23
5.6.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv4 Only Peer	23
6. Considerations	23
6.1 Socket API Conversion	23
6.2 Address Mapping and Embedding	23
6.3 ICMP Message Handling	24
6.4 Implementation Issues	24
7. Limitations	25
8. IANA Considerations	25
9. Security Considerations	25
10. Normative References	27
Appendix: API list intercepted by BIA	29
Authors Addresses	31

1. Motivation And Introduction

1.1 Motivation

It is probably important to give a brief analysis first of one of the blocking factors withholding the wide-spread introduction of IPv6 in order to fully understand why the here proposed BIA is considered an essential component in the unlocking of IPv6.

At the inception of IPv6 it was - rather naively - presumed that all parties involved with the Internet would be eager to make the changeover and that the transition would happen spontaneously.

It is now quite generally acknowledged that some human and commercial factors preventing a spontaneous transition have been largely underestimated. In the transition to IPv6 there are essentially two parties involved: network providers and end-users.

The benefits of using IPv6 are almost entirely for the network providers, while the end-users have only potentially indirect benefits from better network operation. No drive to make the changeover should be expected from the majority of end-users, as they have probably little to gain. The network providers can expect benefits, but they are obviously dependent on the willingness of their end-users to make any changeover. The result is some kind of deadlock: no (commercial) network provider is going to force the customers to make the changeover against their will. So making the transition transparent to the end-user is the key in any transition to IPv6. The average end-users are not really aware about what goes on in the network layer, and even if they do, they usually could not care less. It does not matter much to them if their applications are communicating using IPv4 or IPv6. But, while there is no drive to be expected from the end-users for any transition to IPv6, the vast majority would not object to the transition on condition they can go on using their applications as before.

While the first impression is that applications are not affected by the changeover on the IP layer from IPv4 to IPv6, this is unfortunately not true. The applications are using IP addresses, and hence should be capable of dealing with the longer IPv6 addresses when having to communicate over IPv6.

Expecting all applications to be modified to be capable of dealing with the longer IPv6 addresses is rather naive. Apart from the "standard" Internet applications with rather good support such as web browsers, email programs, etc. that can be expected to be IPv6 enabled, there are thousands of other applications, some of them are written by small companies (of which some may be out of business) and others are even "home-made". For some applications, Internet communication is only a side-issue, for example for registering and/or checking for updates, and upgrading to become IPv6 compatible is probably not a high priority. It is to be expected that a large proportion of applications will only be modified to be IPv6 compatible when IPv6 usage gets into full swing. And even if the

IPv6 capable new versions of application software are made available, it is again rather naive to expect all end-users to do the required updating of all the software on their system.

The end-users MAY be willing to accept a changeover to IPv6, but will NOT accept that some of their applications will no longer work as before. From this observation it becomes obvious that it is absolutely essential that provisions are standard installed and enabled on any general purpose machine (the vast majority of systems connected to the Internet) that is provided for IPv6 communication and potentially has to run IPv4-only applications to continue communicating as before when communicating using IPv6. While the demand for mandatory provisions on every general purpose machine capable of communicating using IPv6 may seem a tall order, it should be realized that this approach is much more realistic than expecting all applications to be made IPv6 compatible: compared to thousands of applications that would need conversion requiring all application developers to follow suit, the number of communication stack implementations on general purpose machines is very small and is made by only a handful of developers.

While somewhat less of an urgent issue, the solution should be general enough to handle the reverse problem as well: an IPv6 only application should be able to communicate on a machine with IPv4 only connectivity, or dual IPv4/IPv6 connectivity when communicating using IPv4 with remote hosts that have IPv4-only connectivity. While this looks like a move in the wrong direction in the context of transition towards IPv6, this capability is also important to break the slowdown of the development of IPv6 compatible applications, as described in [RFC2460]. Little effort is being invested into making applications IPv6 capable, as almost no machines currently have IPv6 connectivity. BIA allows to using these IPv6 capable applications to run on the IPv4 infrastructure, removing the practical limitation that IPv6 applications cannot be used at this time.

Other practical issues are blocking the deployment of IPv6, such as the lack of IPv6 support in public access networks, the lack of real auto configuration between IPv4 and IPv6 connectivity, incompatibility in IPv4/IPv6 connectivity of hosts, etc. Solutions to these other practical issues are being investigated currently by the authors.

1.2 Introduction

The original BIA is an experimental function intended at allowing IPv4 only applications on dual stack (dual connectivity) hosts to communicate over IPv6 with remote IPv6 only applications. It was also only useable in the specific context described.

The proposed BIA is a generalisation of the original concept, allowing any mixture of IPv4/IPv6 type capable applications to communicate over

any IPv4/IPv6 connections with any IPv4/IPv6 type capable remote applications. BIA effectively decouples application IPv4/IPv6 capability from IPv4/IPv6 connectivity, and all allows IPv4/IPv6 incompatibility between two communicating applications. The concept is quite simple: BIA essentially does internal address translation where necessary between IPv4 and IPv6 addresses in between the application and the communication stack; functionally, it can be compared to an internal NAT [RFC3022] between the communication stack and the application layer. Conceptually BIA is an adaptation layer that needs to be inserted between the application layer and the IP communication stack as an API layer on top of the native API functions, offering the same API functions as the native ones to the application layer. In an optimized implementation, it can probably better be implemented as an internal modification to the API itself.

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] .

This document uses terms defined in [RFC2460], [RFC4213], [RFC2767], [RFC3338], and [I-D.draft-huang-behave-rfc3338bis].

2. Applicability and Related techniques

The term IPv4/IPv6 connectivity will be used, rather than stack, since it is the connectivity that specifies whether the machine can communicate using IPv4, IPv6 or both, and not only the implementation of the IP stacks inside the machine; e.g. a dual stack machine has both IPv4 and IPv6 stacks implemented, but may have only IPv4 or IPv6 network connectivity due to the type of network it is connected to, or may have to use IPv6 because the remote has only IPv6 connectivity.

2.1 Applicability

The BIA is a mechanism that should be mandatory installed and enabled on hosts potentially having to run applications with incompatible IPv4/IPv6 addressing capability regarding to their IPv4/IPv6 network connectivity. For example, IPv4 only applications that have to communicate over IPv6; or IPv6 only applications having to communicate over IPv4 only connectivity. It allows an IPv4 only application which is running on the local host to communicate over IPv6 with another IPv4/IPv6 application on another host without any modification. It is important to note that the mechanism assumes that the host knows whether it is connected via dual IPv4/IPv6 connectivity, IPv4 only connectivity, or IPv6 only connectivity (this is an implementation issue and will not be discussed here). Table 1 describes the scenarios of all IPv4/IPv6 capability types of applications running over all

possible types of host connectivities. Only the situations with incompatibility between application IPv4/IPv6 capability and IPv4/IPv6 connectivity are listed.

Source Host			Destination Host	
Appl. Version	Host Connectivity	Network	Host Connectivity	
IPv4	IPv6	<-IPv6->	IPv6	
IPv4	IPv6	<-IPv6->	IPv4/IPv6	
IPv6	IPv4	<-IPv4->	IPv4	
IPv6	IPv4	<-IPv4->	IPv4/IPv6	
IPv4	IPv4/IPv6	<-IPv6->	IPv6	
IPv6	IPv4/IPv6	<-IPv4->	IPv4	

Table 1: List all the scenarios treated by BIA mechanism

2.2 Related Techniques

The original BIA mechanism is customized for dual stack hosts. BIA is a mechanism that is inserted between the socket API module and the TCP/IP module. The main purpose of this mechanism is to make the IPv4 applications communicate with applications that can only communicate using IPv6 (IPv6 only connectivity and/or IPv6 only application) without any modification on those IPv4 applications. This would be achieved by translating the IPv4 socket API functions into IPv6 socket API functions and vice versa.

BIS mechanism [RFC2767] allows host to communicate with other IPv6 hosts using existing IPv4 applications. It is also customized for dual stack hosts. The technique uses SIIT [RFC2765] to translate the IPv4 traffic into IPv6 traffic and vice versa. However, this mechanism uses translator which is inserted between the TCP/IP module and the network card driver. The limitations of this mechanism are similar to the SIIT limitations concerning the IP header translation methods. Its implementation is also fully dependent on the network interface driver.

3. Host Configurations using BIA

BIA can be installed on three different host configurations regarding to IPv4/IPv6 connectivity:

1. IPv4 only host: only IPv4 connectivity.
2. IPv6 only host: only IPv6 connectivity.
3. IPv4/IPv6 host: both IPv4 and IPv6 connectivity.

The connectivity of the local host for communication with a remote host is actually decided by several factors:

- The implementation of stack(s) in the local (IPv4 only stack, IPv6 only stack, or dual stack).
- The network connectivity of the local host (IPv4 network connectivity only, IPv6 network connectivity only, both IPv4 and IPv6 network connectivity).
- The connectivity of the remote machine (IPv4 only connectivity, IPv6 only connectivity, both IPv4 and IPv6 connectivity).

This means that the connectivity of a host, even with dual stack implementation, is dynamic: it depends on the network connectivity, which may change (e.g. a laptop that may be regularly connected to different networks over time) and/or the connectivity of the remote host.

For example a local host may be limited to IPv6 communication with a remote host because it only has an IPv6 stack implemented, it may have a dual stack implementation but only IPv6 network connectivity, or the remote host may have only IPv6 connectivity.

The connectivity of the host will be combined with three possibilities of application addressing capability.

- IPv4 only application: only IPv4 addressing capability.
- IPv6 only application: only IPv6 addressing capability.
- IPv4/IPv6 application: both IPv4 and IPv6 addressing capability.

There will be different behavior for BIA depending on the local host IPv4/IPv6 connectivity as well as the application's IPv4/IPv6 addressing capability.

- IPv4 applications communicating over IPv4 or IPv6 applications communicating over IPv6 are the native situations and do not need consideration here; in this case BIA simply has to perform no action.
- For an IPv4 application that needs to communicate using IPv6, the IPv4 application's addressing needs to be converted to IPv6 in order to be transmitted to the remote host. The opposite conversion has to be applied when an IPv6 application needs to communicate over IPv4.

- For an IPv6 only application on an IPv4/IPv6 host communicating with an IPv4/IPv6 remote host, the mechanism provides an optional feature to make this application able to communicate over IPv4 as well as over IPv6. If this application is trying to communicate over IPv4, the application's addressing needs to be converted to IPv4 in order to be transmitted to the remote host.

3.1 IPv6 Only Host Architecture Using BIA

IPv4 applications need IPv4 connectivity for communication. BIA is a mechanism that enables hosts that have to communicate using IPv6 to run IPv4 applications. Such hosts MUST have BIA installed and enabled. Figure 1 shows the architecture of the IPv6 host in which BIA is installed.

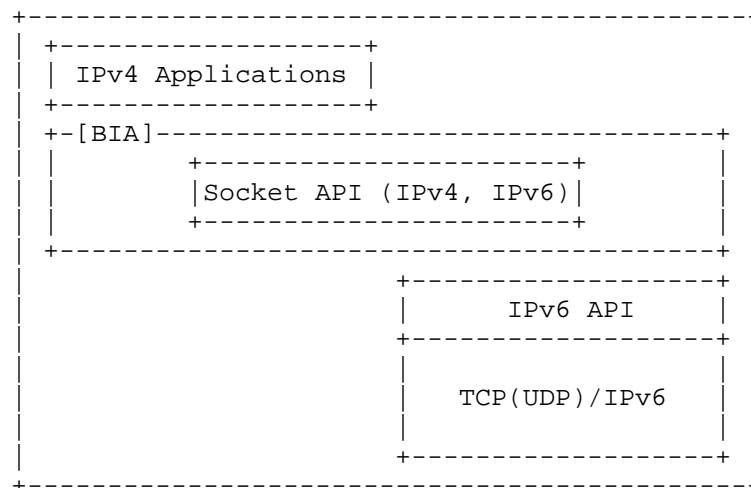


Figure 1: the architecture of IPv6 only host
In which BIA is installed.

3.2 IPv4 Only Host Architecture Using BIA

IPv4 only hosts are capable of running IPv4 applications only. BIA can be installed on such machines to allow these hosts to run IPv6 only applications as well. Figure 2 shows the host architecture of the IPv4 host in which BIA is installed.

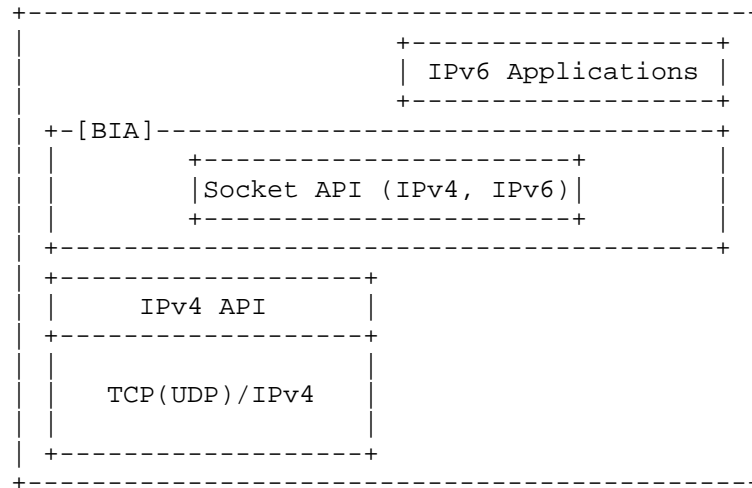


Figure 2: the architecture of IPv4 only host
In which BIA is installed.

3.3 Dual Connectivity Host Architecture Using BIA

[RFC4213] suggests that dual stack hosts need applications, dual TCP/IP modules and addresses for both IPv4 and IPv6. In such hosts, the BIA will be used only when the received DNS record(s) version is incompatible with the running of the application's IPv4/IPv6 capability. For example, if a dual connectivity host is running an IPv4 only application, and this application is going to communicate with an IPv6 only host, then the name resolver will receive the 'AAAA' record for the destination host so that the current connectivity will be IPv6, and BIA will translate the IPv4 socket API functions into IPv6 socket API functions and vice versa. BIA always will use the API functions that are compatible with the destination address. Figure 3 shows a dual connectivity host on which BIA is installed.

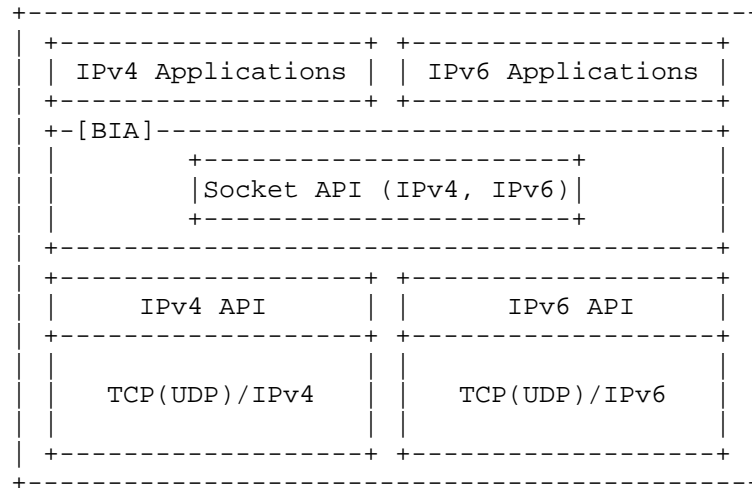


Figure 3: the architecture of dual stack host
In which BIA is installed.

4. BIA Modules

Like BIA, the API translator in BIA consists of three modules, name resolver, address resolver, and function mapper.

4.1 Name Resolver

In general the name resolver module returns a proper answer in response to the IPv4 or IPv6 application's DNS resolving request. The name resolver has different behaviors depending on the running application's IPv4/IPv6 capability and the IPv4/IPv6 connectivity.

4.1.1 IPv4 only application on the local host:

- IPv4 Only Host:

Since this is the native situation, BIA needs to perform no action.

- IPv6 Only Host

This behavior of the name resolver occurs when an IPv4 application needs to communicate using IPv6. The application will try to resolve names via the IPv4 resolver library (e.g. `gethostbyname`). BIA will call the IPv6 equivalent function (e.g. `getnameinfo`) that will resolve both 'A' and 'AAAA' records. If it got 'AAAA' record(s) only, it requests the address resolver (see below) to assign internal IPv4 address(es) corresponding to the IPv6 address(es) of the 'AAAA' record(s), then creates 'A' record(s) for the assigned IPv4 address(es), finally returns the created 'A' record(s) of the internal address(es) to the IPv4 application. Note that this behavior is similar to the name resolver behavior in the BIA mechanism, but

there are differences in the way internal addresses are assigned and managed (see address resolver further).

If both 'A' and 'AAAA' records are received, BIA will discard the 'A' record(s) as these cannot be used with IPv6, and select only the 'AAAA' record(s).

- IPv4/IPv6 Host:

The application will try to resolve names via the IPv4 resolver library (e.g. `gethostbyname`). BIA will call the IPv6 equivalent function (e.g. `getnameinfo`) that will resolve both 'A' and 'AAAA' records. If it got 'AAAA' record(s) only, it requests the address resolver (see below) to assign internal IPv4 address(es) corresponding to the IPv6 address(es) of the 'AAAA' record(s), then creates 'A' record(s) for the assigned IPv4 address(es), finally returns the created 'A' record(s) of the internal address(es) to the IPv4 application. If it got 'A' record(s) only, the communication will continue as native IPv4 communication, and BIA has to do no operation. If both 'A' and 'AAAA' records are returned, the communication can be effected natively using IPv4 using the 'A' record(s). But as an additional, optional feature, the IPv4 application can also be allowed to communicate over IPv6 with IPv6 peer(s). In that case, it requests the address resolver (see below) to assign internal IPv4 address(es) corresponding to the IPv6 address(es) of the 'AAAA' record(s), then creates 'A' record(s) for the assigned IPv4 address(es), and finally returns both the original 'A' for the remote AND the created 'A' record(s) of the internal address(es) to the IPv4 application. This extends the communication capabilities of the application to cover both IPv4 and IPv6 communication with the remote(s).

4.1.2 IPv6 Application on the local host:

- IPv6 Only Host:

Since this is the native situation, BIA needs to perform no action.

- IPv4 Only Host:

This situation occurs when an IPv6 application needs to communicate using IPv4, the application will try to resolve names via the IPv6 resolver library (e.g. `getnameinfo`). BIA will call the IPv4 equivalent function (e.g. `gethostbyname`). If it got 'A' record(s) only, it requests the address resolver to assign an internal IPv4-embedded IPv6 address(es) [I-D.draft-ietf-behave-address-format] corresponding to the IPv4 address(es), then creates 'AAAA' record(s) for the IPv4-embedded IPv6 address(es) and returns these 'AAAA' record(s) to the IPv6 application.

- IPv4/IPv6 Host:

The application will try to resolve names via the IPv6 resolver library (e.g. `gethostinfo`) that will resolve both 'A' and 'AAAA' records. If it

got 'A' record(s) only, the name resolver will request the address resolver to assign internal IPv4-embedded IPv6 address(es) corresponding to the IPv4 address(es), then creates 'AAAA' record(s) for the IPv4-embedded IPv6 address(es) and returns these 'AAAA' record(s) to the application. If it got 'AAAA' records only, the communication will continue as native IPv6 communication, and BIA has to do no operation. If both 'A' and 'AAAA' records are returned, the communication can be effected natively using IPv6 using the 'AAAA' record(s). But as an additional, optional feature, the IPv6 application can also be allowed to communicate over IPv4 with IPv4 peer(s). In that case, it requests the address resolver to assign internal IPv4-embedded IPv6 address(es), then creates 'AAAA' record(s), and finally returns both the original 'AAAA' for the remote AND the created 'AAAA' record(s) of the internal address(es) to the IPv6 application. This extends the communication capabilities of the application to cover both IPv4 and IPv6 communication with the remote(s).

4.1.3 Reverse DNS lookup

For various reasons, applications may do "pointer" lookups, i.e. the application passes the IP address and expects the host name in return. BIA should be able to handle these calls. When address translation (mapping or embedding) was performed on the host IP address, the application will call with the internal address generated by BIA. The DNS call to resolve the name should obviously be made with the external address that corresponds to the translated address, and the name returned for the external address needs to be returned to the application.

4.1.4 Originating without DNS lookup

Some applications bypass the DNS lookup, and use an IP address directly instead. While often this is bad practice, in some instances this is how the software is being operated. For an IPv4 only application making such call, if an address mapping was stored for the supplied IPv4 address, that mapping can be used. Otherwise, as no DNS call is made, a correspondence between IPv4 and IPv6 addresses cannot be made by the name and address resolvers, and communication using incompatible application IPv4 capability and IPv6 connectivity is impossible. But for both for IPv4 and IPv6 applications, as a last resort, a "dirty trick" can be attempted however. Using the IP address from the application, a "pointer" DNS lookup can be made. If this succeeds, a forward DNS lookup can be made on the returned name, which may return one or more addresses of the other type required to establish the required IPv4/IPv6 address relationship. If this trick does not succeed, communication will be impossible, unless native communication is available (IPv4 over IPv4 connectivity or IPv6 over IPv6 connectivity).

It is recommended that address relationships can be manually entered in the mapping table for such occurrences. Such address mappings are in this case external IPv4-to-external IPv6 address mappings, and not relations between an internal and an external address.

4.2 Address Resolver

The address resolver is only involved with incompatibility between application IPv4/IPv6 capability and host IPv4/IPv6 connectivity. The address resolver has different behavior depending on the name resolver and function mapper requests. Like in the original BIA address mapper, the address resolver in BIA maintains a table of the pairs of an internal IPv4 address and an external IPv6 address in an IPv6 only host. These IPv4 addresses are assigned from an IPv4 address pool for internal addresses, but the mechanism for the pool is different here, as explained further.

The key difference between BIA and the BIA mechanism is the ability for the later to address all kinds of remote host connectivity (i.e. IPv4 only, IPv6 only and dual IPv4/IPv6 connectivity). Different addressing techniques are used depending on the remote host. The sending host has to take the decision either to map the destination IPv6 address into an internal IPv4 address assigned from the IPv4 address pool, or to embed the IPv4 address into an internal IPv4-embedded IPv6 address. The Address resolver in BIA can receive two possible address types-normally after calling the name resolver and querying the DNS- regarding the remote host(s) for that domain name:

- IPv6 Address: in this case it receives 'AAAA' record(s) and the address resolver has to map the IPv6 into an internal IPv4 address(es).
- IPv4 address: in this case it receives 'A' record(s) and the address resolver has to embed the IPv4 address into an internal IPv6 address(es).

4.2.1 Mapping

This technique is used when an IPv4 application needs to communicate using IPv6. It internally maintains a table of the pairs of IPv4 address(es) and IPv6 address(es). The IPv4 addresses are assigned from an IPv4 address pool. These addresses should be reserved from an unassigned class A domain reserved by IANA to be used by BIA for mapping purposes (see further). When the name resolver or the function mapper requests it to assign an internal IPv4 address corresponding to an IPv6 address, it selects and returns an IPv4 address out of the pool, and registers a new entry into the table dynamically. As in the original BIA, the registration occurs in the following two cases:

1. When the name resolver gets only an 'AAAA' record for the target host name and there is not a mapping entry for the IPv6 address.
2. When the function mapper gets a socket API function call from the data received and there is not a mapping entry for the IPv6 source address. Address mappings are stored. When the address resolver is called to map an IPv6 external address into an IPv4 internal address, it will first look up the table to check whether there was a previous mapping for that address. If one is found, it will reuse and return that mapping. If not, it will create a new mapping, store that mapping and return the newly created mapping.

4.2.2 Embedding

Unlike the original BIA, BIA also allows IPv6 only applications to communicate over IPv4. Therefore a correspondence between external IPv4 addresses and internal IPv6 addresses need to be established. The proposed method is "IPv4-in-IPv6" address embedding [I-D.draft-ietf-behave-address-format]. The address resolver is configured to use one of the methodologies that are described in [I-D.draft-ietf-behave-address-format] to create an IPv4-embedded IPv6 address. The new address consists of: Network Specific Prefix (NSP) (32 bits), the IPv4 destination address (32 bits), and finally the suffix (64 bits). Figure 4 demonstrates the IPv4-embedded IPv6 address structure. As the real external IPv4 address is embedded into the internal IPv6 address, no registering is required in this case, as there is always a unique correspondence.

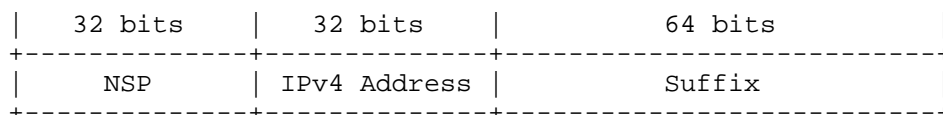


Figure 4: The structure of IPv4-embedded IPv6 address

4.3 Function Mapper

The function mapper has different behavior depending on the host connectivity. In dual connectivity hosts (IPv4 and IPv6 connectivity) the function mapper is used to decide which API functions to call in the current communication. It is important to note that the BIA modules will be invoked just if there is an incompatibility between the running application and the connectivity type. In case of IPv6 only or IPv4 only connectivity, the main goal of the function mapper is just like in original BIA mechanism. It is used when conversion is required from IPv4 to IPv6 or the other way around between application layer and communication stack. In that case it translates the API functions used by the application into the API functions needed for the communication, and vice versa.

In dual connectivity hosts, deciding which API functions to call depends on the address type of the remote. The following is the behavior of the function mapper running on dual connectivity hosts:

1. IPv4 only application communicating over IPv6: in this case it will call the equivalent IPv6 socket API functions. The application will use the IPv4 socket API to communicate with other hosts. Since the application needs to communicate over IPv6, the function mapper intercepts the IPv4 socket API functions and calls the equivalent IPv6 socket API functions instead.

2. IPv6 only application communicating over IPv4: in this case it will call the equivalent IPv4 socket API functions. The application will use the IPv6 socket API to communicate with other hosts. Since the application needs to communicate over IPv4, the function mapper intercepts the IPv6 socket API functions and calls the equivalent IPv4 socket API functions instead.

5. Behavior Examples

The following sections will describe the behaviors of the hosts and applications that are listed in table 1.

In the following sections, the meanings of arrows are as follows:

- > A DNS message for name resolving created by the Applications and the name resolver in the API translator.
- +++> An IPv4 or IPv4-embedded IPv6 address request to and reply from the address resolver for the name resolver and the function mapper.
- ==> Data flow by API functions created by the applications and the function mapper in the API translator.

5.1 IPv4 Only Application, IPv6 Only Connectivity with an IPv6 Only Peer.

5.1.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating to IPv6 Only Peer

When an IPv4 application sends a DNS query to its name server, the name resolver intercepts the query and then creates a new query to resolve both 'A' and 'AAAA' records. When only 'AAAA' record(s) is (are) resolved, the name resolver requests the address resolver to get IPv4 address(es) corresponding to the IPv6 address(es) for each IPv6 address from the 'AAAA' record. The address resolver first looks up the table of stored entries to check if the correspondence was made previously. If yes, the stored mapping is retrieved and passed to the name resolver. If not, the address resolver creates a new mapping for an internal IPv4 address corresponding to the IPv6 external address, stores the mapping, and returns the mapping to the name resolver. The name resolver, upon receiving the internal IPv4 address(es) creates 'A' record(s) for the

assigned IPv4 address(es) and returns these to the application. In order for the IPv4 application to send IPv4 packets over IPv6, it calls the IPv4 socket API function. The function mapper detects the API function call from the application. The IPv6 address is required to invoke the IPv6 socket API function, thus the function mapper requests the IPv6 address corresponding for the internal IPv4 address to the address resolver. The address resolver selects the external destination IPv6 address corresponding to the internal IPv4 address from the mapping table and returns it to the function mapper. Using this IPv6 address, the function mapper will invoke the IPv6 socket API function corresponding to the IPv4 socket API function received from the application.

When a reply is received, this will come in through the IPv6 socket API, and the function mapper requests the address resolver for the IPv4 address corresponding to the received IPv6 address. This IPv4 address will be used to translate the IPv6 socket API function call into the corresponding IPv4 socket API function call for the IPv4 application. Figure 5 illustrates the behavior described above.

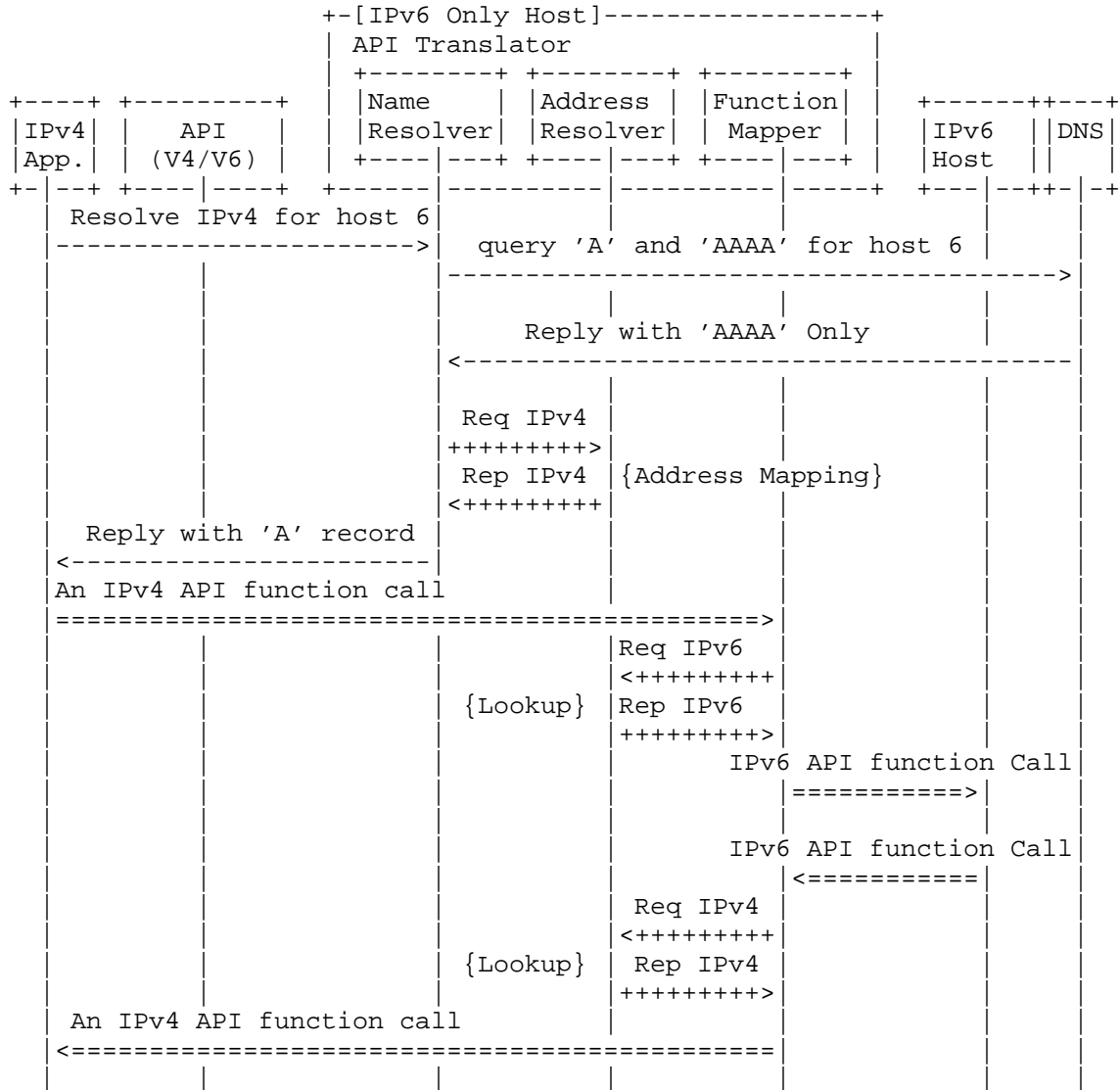


Figure 5: The behavior of the originator communicates with IPv6 Application

5.1.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host

The IPv6 originator host that started the communication to this host has resolved the address of this IPv6 host with 'AAAA' record(s) through its name server, and has sent an IPv6 packet to this IPv6 host. The function mapper requests the internal IPv4 address

corresponding to the originator's IPv6 address. The address resolver looks up the mapping table to check for an entry. If one is found, it returns the internal IPv4 address corresponding to the IPv6 address. Then the function mapper invokes the corresponding IPv4 socket API function for the IPv4 application corresponding to the IPv6 function. If not, the address resolver creates a new mapping for an internal IPv4 address corresponding to the IPv6 external address, stores the mapping, and returns the mapping to the function resolver. The remaining part of the handling is identical to what was described in 5.1.1. Figure 6 illustrates the behavior described above.

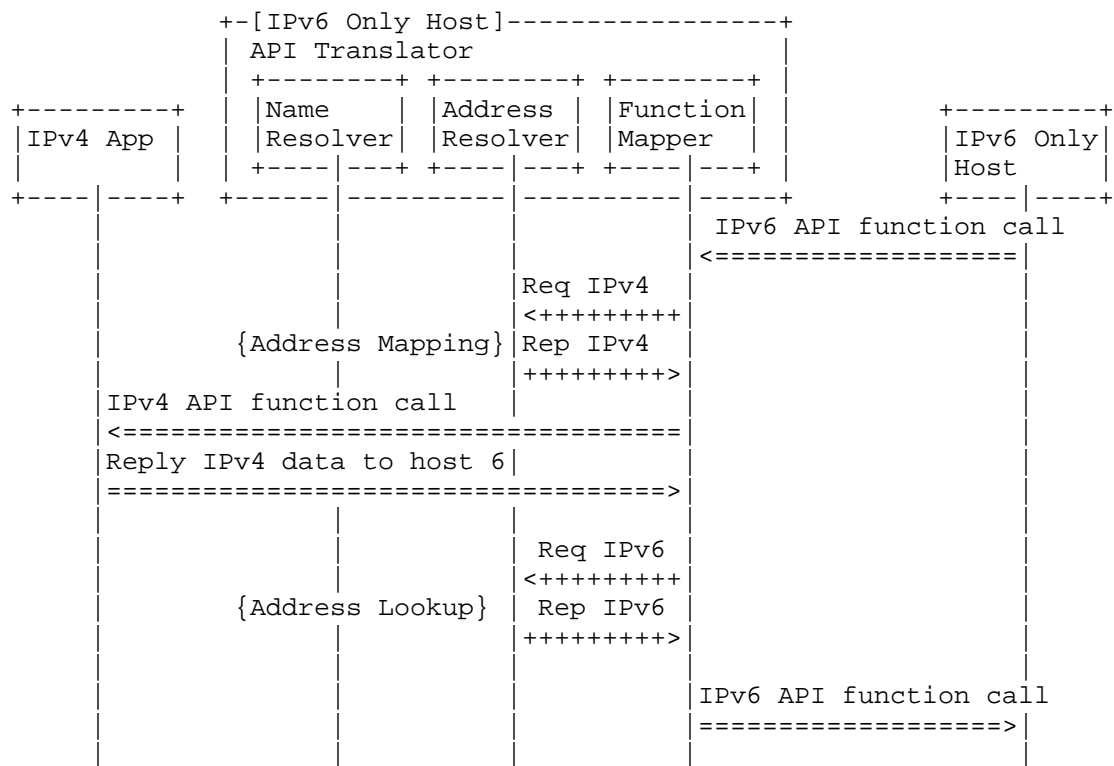


Figure 6: Behavior of receiving data from IPv6 host

5.2 IPv4 Only Application, IPv6 Only Network and Dual Connectivity Peer

5.2.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating with Dual Connectivity Host

When an IPv4 application sends a DNS query to its name server, the name resolver intercepts the query and then creates a new query to resolve

both 'A' and 'AAAA' records. If both 'A' and 'AAAA' records are resolved, the name resolver will only select the 'AAAA' records and drop the 'A' record(s) and requests the address resolver to assign internal IPv4 address(es) corresponding to the IPv6 address(es). The remaining behavior is exactly like described in 5.1.1.

5.2.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host Communicating with Dual Connectivity Host

Exactly the same as in section 5.1.2

5.3 IPv6 Only Application, IPv4 Only Connectivity with an IPv4 Only Peer

5.3.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with IPv4 Only Host

When an IPv6 application sends a DNS query to its name server to resolve both 'A' and 'AAAA' records, the name resolver intercepts the query and then creates a new query to resolve only 'A' record(s), since it is a IPv4 only host. With only 'A' record(s) resolved, the name resolver requests the address resolver to embed the IPv4 address(es) into IPv6 address(es) using the format described in section 4.2.2. The name resolver creates 'AAAA' record(s) for the IPv4 embedded IPv6 address(es) and returns it to the application. In order for the IPv6 application to send IPv6 packets to IPv4 only host, it calls the IPv6 socket API function. The function mapper detects the API function call from the application. The function mapper requires an IPv4 address to invoke the IPv4 socket API function, so it requests the corresponding IPv4 address to the address resolver. The address resolver extracts the destination IPv4 address from the IPv4-embedded IPv6 address and returns it to the function mapper. Using this IPv4 address, the function mapper will invoke the IPv4 socket API function corresponding to the IPv6 socket API function. We notice here the address resolver is not going to save any new records to the mapping table.

When a reply is received, this will come in through the IPv4 socket API, and the function mapper requests the address resolver for the IPv6 address corresponding to the received IPv4 address. This IPv6 address will be used to translate the IPv4 socket API function call into the corresponding IPv6 socket API function call for the IPv6 application. Figure 7 illustrates the behavior described above.

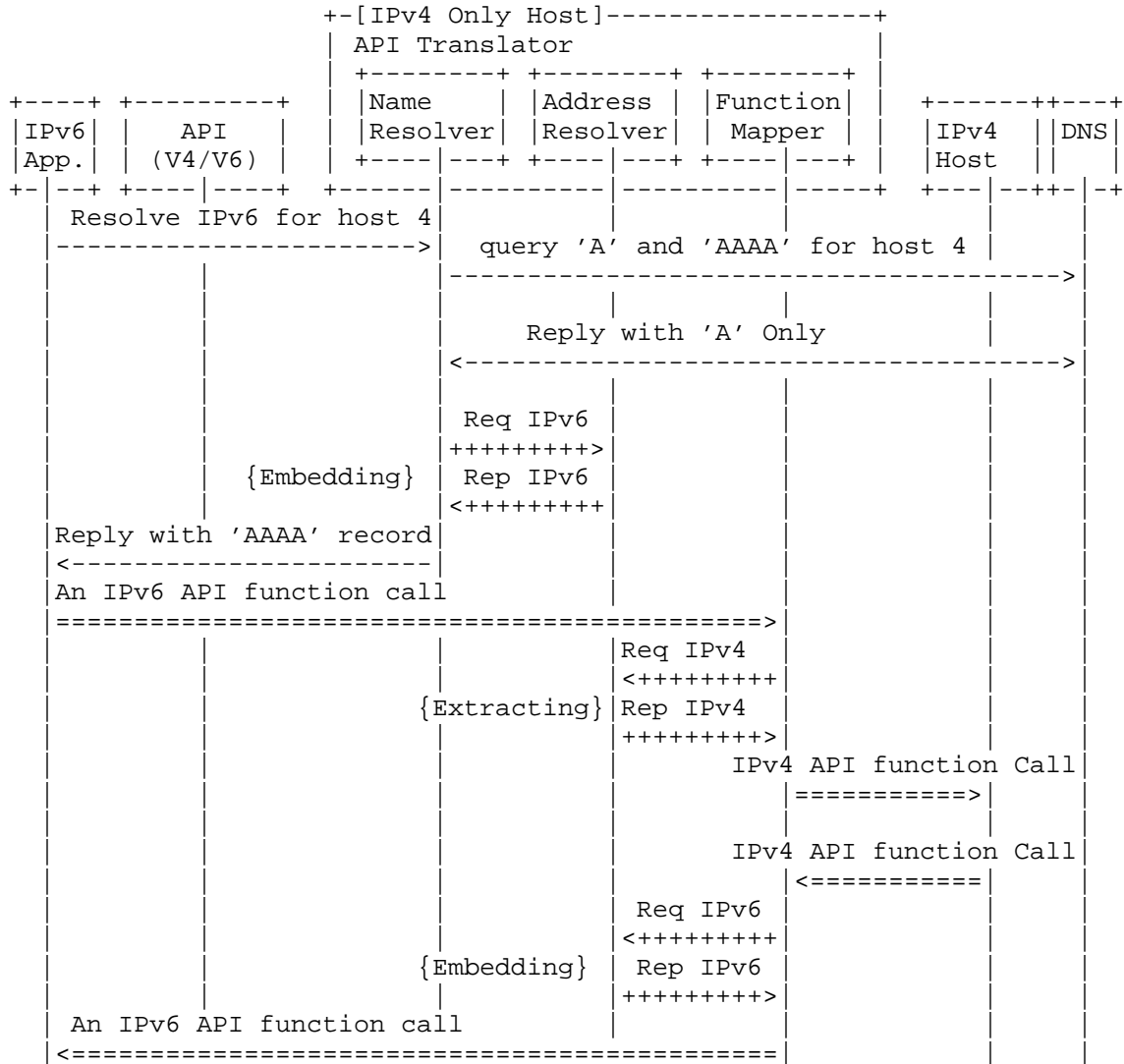


Figure 7: The behavior of the originator communicates with IPv4 Application

5.3.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with IPv4 Only Host

The IPv4 originator host has resolved the address of this IPv4 host with 'A' records through its name server, and has sent an IPv4 packet to this IPv4 host. The function mapper requests the IPv6 address to the address resolver in order to invoke the IPv6 socket API function to communicate with the IPv6 application. The address resolver embeds

the IPv4 address(es) into IPv6 address(es) using the format described in section 4.2.2, and returns this address. Then the function mapper invokes the corresponding IPv6 socket API function for the IPv6 application corresponding to the IPv4 function.

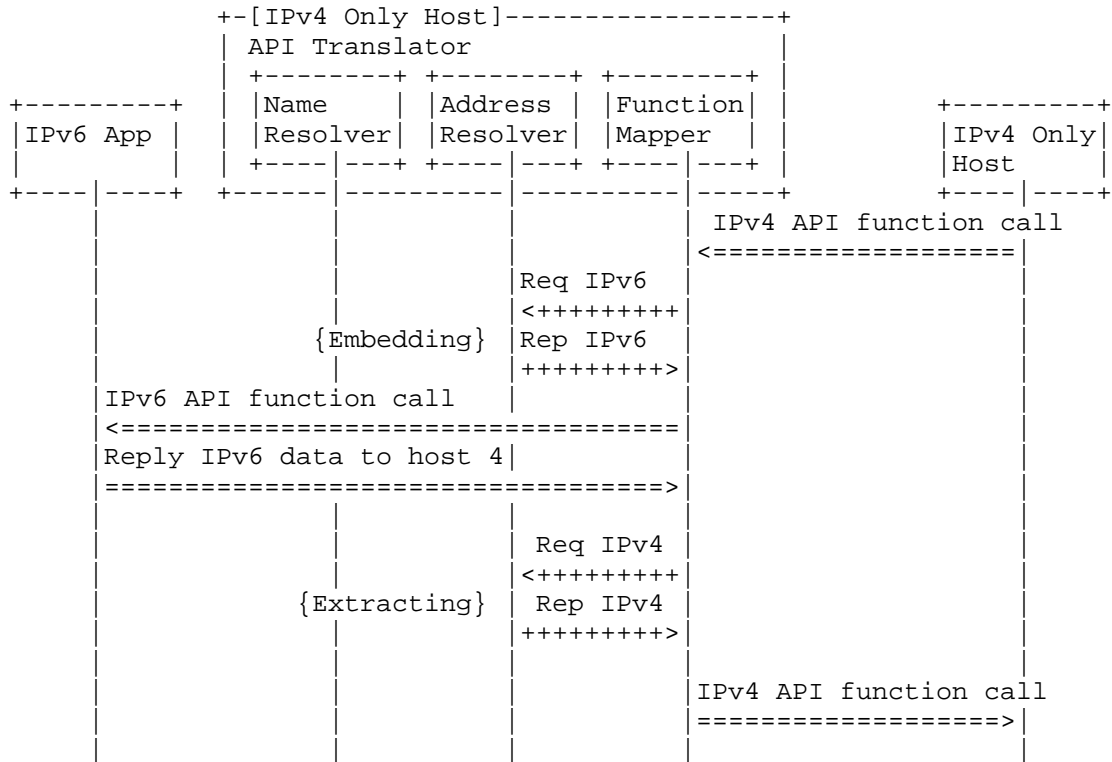


Figure 8: Behavior of receiving data from IPv4 host

5.4 IPv6 Only Application, IPv4 Only Network and Dual Connectivity Peer

5.4.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with Dual Connectivity Host

When an IPv6 application sends a DNS query to its name server, the name resolver intercepts the query and then creates a new query to resolve 'A' record(s); no 'AAAA' record(s) are returned, as it is an IPv4 only host. When 'A' record(s) is/are resolved, the name resolver will request the address resolver to embed the IPv4 address(es) into IPv6 address(es) using the format that is describes in section 4.2.2. The remaining processing is as in 5.3.1.

5.4.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with Dual Connectivity Host

Exactly the same as in section 5.3.2

5.5 IPv4 Only Application on Dual Connectivity Host Communicating to IPv6 Only Peer

5.5.1 IPv4 Only Originator Application on Dual Connectivity Host Communicating to IPv6 Only Peer

Exactly the same as in section 5.1.1

5.5.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv6 Only Peer

Exactly the same as in section 5.1.2

5.6 IPv6 Only Application on Dual Connectivity Host Communicating to IPv4 Only Peer

5.6.1 IPv6 Only Originator Application on Dual Connectivity Host Communicating to IPv4 Only Peer

Exactly the same as in section 5.3.1

5.6.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv4 Only Peer

Exactly the same as in section 5.3.2

6. Considerations

6.1 Socket API Conversion

IPv4 socket API functions are translated into semantically the same IPv6 socket API functions and vice versa. See Appendix A for the API list intercepted by BIA. IP addresses embedded in application layer protocols (e.g., FTP) can be translated in API functions. Its implementation depends on operating systems.

NOTE: Basically, IPv4 socket API functions are not fully compatible with IPv6 since the IPv6 has new advanced features.

6.2 Address Mapping and Embedding

There are some considerations as to the choice and management of the internal addresses:

- For a diversity of reasons, several applications store the addresses of machines they have been communicating with, and check these

addresses on next contact. It is hence important that each mapping of external IPv6 to internal IPv4 addresses is being stored by BIA, so as to always use the same internal address for a particular external address at the next communication. This implies a very wide IPv4 address range available for mapping. The authors propose a Class A range, making approximately 16Mega addresses available. Even that can get exhausted in some cases, so this range should be supplemented by a round-robin scheme where, in case of exhaustion, the mappings that remain unused for the longest time can be reused for new mappings (not the creation time, but the last time that mapping was actively used is important). It is considered that this would be sufficient in about all operational situations. As this mapping list potentially can become very large, the store/retrieval mechanism implementation should be optimized for speed or it may introduce unacceptable long delays. This is an implementation issue however, and will not be dealt with here.

- It is obvious that an IPv4/IPv6 correspondence will be frequently required in the course of a communication; short time caching seems essential to avoid having to look up the correspondence again during the course of a communication.
- A machine having both IPv4 and IPv6 connectivity will be using both IPv4 and IPv6 addresses. To avoid conflicts, it is essential that the internal addresses used will never be used as an external address. Original BIA proposed the use of a limited (256 addresses) range as a "pool" in an "unassigned" IPv4 address range. The limited size (256) is much too small for operational purposes, even not considering the requirement for storing the mappings as described in the previous paragraph, as a machine may have more than this number of mappings active concurrently. Taking into account the requirement for storing the mappings, a very large range of unassigned addresses is required. Please refer to section 8 of this document.

6.3 ICMP Message Handling

When an application needs ICMP messages values (e.g., Type, Code, etc.) sent from a network layer, ICMPv4 message values MAY be translated into ICMPv6 message values based on SIIT [RFC2765], and vice versa. It can be implemented using raw socket.

6.4 Implementation Issues

Some operating systems support the preload library functions, so it is easy to implement the API translator by using it. For example, the user can replace all existing socket API functions with user-defined socket API functions which translate the socket API function. In this case, every IPv4 application has its own translation library using a preloaded library which will be bound into the application before executing it dynamically.

Some other operating systems support the user-defined layered protocol allowing a user to develop some additional protocols and put them in the existing protocol stack. In this case, the API translator can be implemented as a layered protocol module.

In the above two approaches, it is assumed that there exists both TCP(UDP)/IPv4 and TCP(UDP)/IPv6 stacks and there is no need to modify or to add a new TCP-UDP/IPv6 stack.

7. Limitations

This mechanism supports unicast communications only. In order to support multicast functions, some other additional functionalities must be considered in the function mapper module.

Since the IPv6 socket API has new advanced features, it is difficult to translate such kinds of IPv6 socket APIs into IPv4 socket APIs. Thus, IPv6 inbound communication with advanced features may be discarded.

It should be noted that the original BIA assumes the hosts have compatible network connectivity. The new version of the BIA is developed to support the heterogeneity between connectivity and applications only, NOT incompatible network connectivity. Communication between hosts with incompatible connectivity (IPv4 only connectivity to IPv6 only connectivity, or the other way around) cannot be handled by BIA, and other solutions need to be applied, e.g. protocol translation mechanisms PNAT [I-D.draft-huang-behave-pnat], NAT64 [I-D.ietf-behave-v6v4-xlate-stateful], NAT-PT-HIST[RFC4966], or [I-D.draft-ietf-behave-v6v4-framework].

8. IANA Considerations

The authors propose that IANA reserves one of the few remaining reserved IPv4 Class A ranges specifically to be used in the internal mapping, and making sure this range will never be used for external addressing. While giving up one of the precious last remaining IPv4 Class A ranges for this purpose seems a big demand, the authors feel that unblocking one of the main obstacles in IPv6 deployment warrants this.

Similarly, a NSP for the embedding of IPv4 in internal IPv6 addresses should be reserved by IANA for BIA use, in order to avoid conflicts with other types of embedded IPv6 addresses being used as external addresses. This assignment should not be a big problem however.

9. Security Considerations

The security consideration of BIA mostly relies on that of NAT-PT-HIST [RFC4966]. The differences are due to the

address translation occurring at the API and not in the network layer. That is, since the mechanism uses the API translator at the socket API level, hosts can utilize the security of the network layer (e.g., IPsec) when they communicate with IPv6 hosts using IPv4 applications via the mechanism. As well, there isn't a DNS ALG as in NAT-PT-HIST, so there is no interference with DNSSEC.

The use of address pooling may open a denial of service attack vulnerability. So BIA should employ the same sort of protection techniques as NAT-PT-HIST [RFC4966] does.

10. Normative References

- [RFC3338] Lee, S., Shin, M-K., Kim, Y-J., Nordmark, E., and A. Durand, "Dual Stack Hosts Using "Bump-in-the-API" (BIA)", RFC 3338, October 2002.
- [RFC2460] Deering, S., and R., Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.
- [RFC2767] Tsuchiya, K., HIGUCHI, H., and Y. Atarashi, "Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)", RFC 2767, February 2000.
- [I-D.draft-huang-behave-rfc3338bis]
Huang, B., Deng, H., and T. Savolainen, "Dual Stack Hosts Using "Bump-in-the-API" (BIA)",
draft-huang-behave-rfc3338bis-02 (work in progress),
March 2010.
- [RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.
- [I-D.draft-ietf-behave-address-format]
Huitema, C., "IPv6 Addressing of IPv4/IPv6 Translators",
draft-ietf-behave-address-format-10 (work in progress),
August 2010.
- [I-D.draft-huang-behave-pnat]
Huang, B., and H., Deng, "Prefix NAT: Host based IPv6 translation", draft-huang-behave-pnat-01 (work in progress), February 2010.
- [I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 2010.

- [RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.
- [I-D.draft-ietf-behave-v6v4-framework]
Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation",
draft-ietf-behave-v6v4-framework-10(work in progress),
August 17, 2010.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

Appendix A : API list intercepted by BIA

The following functions are the API list which SHOULD be intercepted by BIA module.

The functions that the application uses to pass addresses into the system are:

```
bind()  
connect()  
sendmsg()  
sendto()
```

The functions that return an address from the system to an application are:

```
accept()  
recvfrom()  
recvmsg()  
getpeername()  
getsockname()
```

The functions that are related to socket options are:

```
getsockopt()  
setsockopt()
```

The functions that are used for conversion of IP addresses embedded in application layer protocol (e.g., FTP, DNS, etc.) are:

```
recv()  
send()  
read()  
write()
```

As well, raw sockets for IPv4 and IPv6 MAY be intercepted.

Most of the socket functions require a pointer to the socket address structure as an argument. Each IPv4 argument is mapped into corresponding an IPv6 argument, and vice versa.

According to [RFC3493], the following new IPv6 basic APIs and structures are required.

IPv4	new IPv6

AF_INET	AF_INET6
sockaddr_in	sockaddr_in6
gethostbyname()	getaddrinfo()
gethostbyaddr()	getnameinfo()
inet_ntoa()/inet_addr()	inet_pton()/inet_ntop()
INADDR_ANY	in6addr_any

BIA MAY intercept `inet_ntoa()` and `inet_addr()` and use the address mapper for those. Doing that enables BIA to support literal IP addresses.

The `gethostbyname()` call return a list of addresses. When the name resolver function invokes `getaddrinfo()` and `getaddrinfo()` returns multiple IP addresses, whether IPv4 or IPv6, they SHOULD all be represented in the addresses returned by `gethostbyname()`. Thus if `getaddrinfo()` returns multiple IPv6 addresses, this implies that multiple address mappings will be created; one for each IPv6 address.

Authors' Addresses

Ala Hamarsheh

Electronics and Informatics Department ETRO/Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Brussels, Belgium

Tel: +32 2 629 2930

Fax: +32 2 629 2883

Email: ala.hamarsheh@vub.ac.be

Prof. Marnix Goossens

Electronics and Informatics Department ETRO/Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Brussels, Belgium

Tel: +32 2 629 2987

Fax: +32 2 629 2883

Email: marnix.goossens@vub.ac.be

Internet-Draft

BIAv2

September 2010

Network working group
Internet-Draft
Intended status: Informational
Expires: July 23, 2011

A. Hamarsheh
ETRO/Vrije Universiteit Brussel
M. Goossens
ETRO/Vrije Universiteit Brussel
January 19, 2011

Hosts with Any Network Connectivity Using "Bump-in-the-API" (BIA)
draft-hamarsheh-behave-biav2-05

Abstract

This document specifies a mechanism for hosts with any network connectivity (IPv4 only, IPv6 only, or dual IPv4/IPv6 connectivity) to run applications of any capability (IPv4 only, IPv6 only, or dual IPv4/IPv6) without any modification to those applications. It is a generalisation of a previous experimental protocol called "Bump-in-the-API" (BIA) [RFC3338]. New mechanism of BIA allows a changeover between the application layer and the IP communication layers from IPv4 to IPv6 and vice versa or IPv6 to IPv4 and vice versa, without requiring those applications to be converted in addressing capabilities, effectively shielding the application layer from IPv4 or IPv6 connectivity. This is considered by the authors to be one of the essential conditions for the transition to IPv6 in the Internet to be successful.

Status of this Memo

This Internet-Draft is submitted to IETF 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 July 23, 2011.

Copyright Notice

Copyright (c) 2010 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 BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents:

1. Motivation and Introduction	5
1.1 Motivation	5
1.2 Introduction	6
2. Applicability and Related Techniques	7
2.1 Applicability	7
2.2 Related Techniques	8
3. Host Configurations Using BIA	9
3.1 IPv6 Only Host Using BIA	10
3.2 IPv4 Only Host Using BIA	10
3.3 Dual Connectivity Host Using BIA	11
4. BIA Modules	12
4.1 Name Resolver	12
4.1.1 IPv4 Only Application on The Local Host	12
4.1.2 IPv6 Only Application on The Local Host	13
4.1.3 Reverse DNS Lookup	14
4.1.4 Originating Without DNS Lookup	14
4.2 Address Resolver	15
4.2.1 Mapping	15
4.2.2 Embedding	16
4.3 Function Mapper	16

5. Behavior Examples	17
5.1 IPv4 Only Application, IPv6 Only Connectivity with an IPv6 Only Peer	18
5.1.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating to IPv6 Only Peer	18
5.1.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host	19
5.2 IPv4 only Application, IPv6 Only Network and Dual Connectivity Peer	21
5.2.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating with Dual Connectivity Host ..	21
5.2.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host Communicating with Dual Connectivity Host ..	21
5.3 IPv6 Only Application, IPv4 Only Connectivity with an IPv4 Only Peer	21
5.3.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with IPv4 Only Host	21
5.3.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with IPv4 Only Host	23
5.4 IPv6 Only Application, IPv4 Only Network and Dual Connectivity Peer	24
5.4.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with Dual Connectivity Host ..	24
5.4.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with Dual Connectivity Host ..	24
5.5 IPv4 Only Application on Dual Connectivity Host Communicating to IPv6 Only Peer	24
5.5.1 IPv4 Only Originator Application on Dual Connectivity Host Communicating to IPv6 Only Peer	24
5.5.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv6 Only Peer	24
5.6 IPv6 Only Application on Dual Connectivity Host Communicating to IPv4 Only Peer	24
5.6.1 IPv6 Only Originator Application on Dual Connectivity Host Communicating to IPv4 Only Peer	24
5.6.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv4 Only Peer	24
6. Considerations	25
6.1 Socket API Conversion	25
6.2 Address Mapping and Embedding	25
6.3 ICMP Message Handling	26
6.4 Implementation Issues	26
7. Limitations	26
8. IANA Considerations	26
9. Security Considerations	27

10. References	27
10.1 Normative References	27
10.2 Informative References	28
Authors Addresses	29

1. Motivation And Introduction

1.1 Motivation

It is probably important to give a brief analysis first of one of the blocking factors withholding the wide-spread introduction of IPv6 in order to fully understand why the here proposed BIA is considered an essential component in the unlocking of IPv6.

At the inception of IPv6 it was - rather naively - presumed that all parties involved with the Internet would be eager to make the changeover and that the transition would happen spontaneously.

It is now quite generally acknowledged that some human and commercial factors preventing a spontaneous transition have been largely underestimated. In the transition to IPv6 there are essentially two parties involved: network providers and end-users.

The benefits of using IPv6 are almost entirely for the network providers, while the end-users have only potentially indirect benefits from better network operation. No drive to make the changeover should be expected from the majority of end-users, as they have probably little to gain. The network providers can expect benefits, but they are obviously dependent on the willingness of their end-users to make any changeover. The result is some kind of deadlock: no (commercial) network provider is going to force the customers to make the changeover against their will. So making the transition transparent to the end-user is the key in any transition to IPv6. The average end-users are not really aware about what goes on in the network layer, and even if they do, they usually could not care less. It does not matter much to them if their applications are communicating using IPv4 or IPv6. But, while there is no drive to be expected from the end-users for any transition to IPv6, the vast majority would not object to the transition on condition they can go on using their applications as before.

While the first impression is that applications are not affected by the changeover on the IP layer from IPv4 to IPv6, this is unfortunately not true. The applications are using IP addresses, and hence should be capable of dealing with the longer IPv6 addresses when having to communicate over IPv6.

Expecting all applications to be modified to be capable of dealing with the longer IPv6 addresses is rather naive. Apart from the "standard" Internet applications with rather good support such as web browsers, email programs, etc. that can be expected to be IPv6 enabled, there are thousands of other applications, some of them are written by small companies (of which some may be out of business) and others are even "home-made". For some applications, Internet communication is only a side-issue, for example for registering and/or checking for updates, and upgrading to become IPv6 compatible is probably not a high priority. It is to be expected that a large

proportion of applications will only be modified to be IPv6 compatible when IPv6 usage gets into full swing. And even if the IPv6 capable new versions of application software are made available, it is again rather naive to expect all end-users to do the required updating of all the software on their system.

The end-users MAY be willing to accept a changeover to IPv6, but will NOT accept that some of their applications will no longer work as before. From this observation it becomes obvious that it is absolutely essential that provisions are standard installed and enabled on any general purpose machine (the vast majority of systems connected to the Internet) that is provided for IPv6 communication and potentially has to run IPv4-only applications to continue communicating as before when communicating using IPv6. While the demand for mandatory provisions on every general purpose machine capable of communicating using IPv6 may seem a tall order, it should be realized that this approach is much more realistic than expecting all applications to be made IPv6 compatible: compared to thousands of applications that would need conversion requiring all application developers to follow suit, the number of communication stack implementations on general purpose machines is very small and is made by only a handful of developers.

While somewhat less of an urgent issue, the solution should be general enough to handle the reverse problem as well: an IPv6 only application should be able to communicate on a machine with IPv4 only connectivity, or dual IPv4/IPv6 connectivity when communicating using IPv4 with remote hosts that have IPv4-only connectivity. While this looks like a move in the wrong direction in the context of transition towards IPv6, this capability is also important to break the slowdown of the development of IPv6 compatible applications, as described in [RFC2460]. Little effort is being invested into making applications IPv6 capable, as almost no machines currently have IPv6 connectivity. BIA allows to using these IPv6 capable applications to run on the IPv4 infrastructure, removing the practical limitation that IPv6 applications cannot be used at this time.

Other practical issues are blocking the deployment of IPv6, such as the lack of IPv6 support in public access networks, the lack of real auto configuration between IPv4 and IPv6 connectivity, incompatibility in IPv4/IPv6 connectivity of hosts, etc. Solutions to these other practical issues are being investigated currently by the authors.

1.2 Introduction

The original BIA is an experimental function intended at allowing IPv4 only applications on dual stack (dual connectivity) hosts to

communicate over IPv6 with remote IPv6 only applications. It was also only usable in the specific context described. The proposed BIA is a generalisation of the original concept, allowing any mixture of IPv4/IPv6 type capable applications to communicate over any IPv4/IPv6 connections with any IPv4/IPv6 type capable remote applications. BIA effectively decouples application IPv4/IPv6 capability from IPv4/IPv6 connectivity, and all allows IPv4/IPv6 incompatibility between two communicating applications. The concept is quite simple: BIA essentially does internal address translation where necessary between IPv4 and IPv6 addresses in between the application and the communication stack; functionally, it can be compared to an internal NAT [RFC3022] between the communication stack and the application layer. Conceptually BIA is an adaptation layer that needs to be inserted between the application layer and the IP communication stack as an API layer on top of the native API functions, offering the same API functions as the native ones to the application layer. In an optimized implementation, it can probably better be implemented as an internal modification to the API itself.

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] .

This document uses terms defined in [RFC2460], [RFC4213], [RFC2767], [RFC3338], and [I-D.draft-huang-behave-rfc3338bis].

2. Applicability and Related techniques

The term IPv4/IPv6 connectivity will be used, rather than stack, since it is the connectivity that specifies whether the machine can communicate using IPv4, IPv6 or both, and not only the implementation of the IP stacks inside the machine; e.g. a dual stack machine has both IPv4 and IPv6 stacks implemented, but may have only IPv4 or IPv6 network connectivity due to the type of network it is connected to, or may have to use IPv6 because the remote has only IPv6 connectivity.

2.1 Applicability

The BIA is a mechanism that should be mandatory installed and enabled on hosts potentially having to run applications with incompatible IPv4/IPv6 addressing capability regarding to their IPv4/IPv6 network connectivity. For example, IPv4 only applications that have to communicate over IPv6; or IPv6 only applications having to communicate over IPv4 only connectivity. It allows an IPv4 only application which is running on the local host to communicate over IPv6 with another IPv4/IPv6 application on another host without any modification.

It is important to note that the mechanism assumes that the host knows whether it is connected via dual IPv4/IPv6 connectivity, IPv4 only connectivity, or IPv6 only connectivity (this is an implementation issue and will not be discussed here). Table 1 describes the scenarios of all IPv4/IPv6 capability types of applications running over all possible types of host connectivities. Only the situations with incompatibility between application IPv4/IPv6 capability and IPv4/IPv6 connectivity are listed.

Source Host			Destination Host	
Appl. Version	Host Connectivity	Network	Host Connectivity	
IPv4	IPv6	<-IPv6->	IPv6	
IPv4	IPv6	<-IPv6->	IPv4/IPv6	
IPv6	IPv4	<-IPv4->	IPv4	
IPv6	IPv4	<-IPv4->	IPv4/IPv6	
IPv4	IPv4/IPv6	<-IPv6->	IPv6	
IPv6	IPv4/IPv6	<-IPv4->	IPv4	

Table 1: List all the scenarios treated by BIA mechanism

2.2 Related Techniques

The original BIA mechanism is customized for dual stack hosts. BIA is a mechanism that is inserted between the socket API module and the TCP/IP module. The main purpose of this mechanism is to make the IPv4 applications communicate with applications that can only communicate using IPv6 (IPv6 only connectivity and/or IPv6 only application) without any modification on those IPv4 applications. This would be achieved by translating the IPv4 socket API functions into IPv6 socket API functions and vice versa.

BIS mechanism [RFC2767] allows host to communicate with other IPv6 hosts using existing IPv4 applications. It is also customized for dual stack hosts. The technique uses SIIT [RFC2765] to translate the IPv4 traffic into IPv6 traffic and vice versa. However, this mechanism uses translator which is inserted between the TCP/IP module and the network card driver. The limitations of this mechanism are similar to the SIIT limitations concerning the IP header translation methods. Its implementation is also fully dependent on the network interface driver.

3. Host Configurations using BIA

BIA can be installed on three different host configurations regarding to IPv4/IPv6 connectivity:

1. IPv4 only host: only IPv4 connectivity.
2. IPv6 only host: only IPv6 connectivity.
3. IPv4/IPv6 host: both IPv4 and IPv6 connectivity.

The connectivity of the local host for communication with a remote host is actually decided by several factors:

- The implementation of stack(s) in the local (IPv4 only stack, IPv6 only stack, or dual stack).
- The network connectivity of the local host (IPv4 network connectivity only, IPv6 network connectivity only, both IPv4 and IPv6 network connectivity).
- The connectivity of the remote machine (IPv4 only connectivity, IPv6 only connectivity, both IPv4 and IPv6 connectivity).

This means that the connectivity of a host, even with dual stack implementation, is dynamic: it depends on the network connectivity, which may change (e.g. a laptop that may be regularly connected to different networks over time) and/or the connectivity of the remote host.

For example a local host may be limited to IPv6 communication with a remote host because it only has an IPv6 stack implemented, it may have a dual stack implementation but only IPv6 network connectivity, or the remote host may have only IPv6 connectivity.

The connectivity of the host will be combined with three possibilities of application addressing capability.

- IPv4 only application: only IPv4 addressing capability.
- IPv6 only application: only IPv6 addressing capability.
- IPv4/IPv6 application: both IPv4 and IPv6 addressing capability.

There will be different behavior for BIA depending on the local host IPv4/IPv6 connectivity as well as the application's IPv4/IPv6 addressing capability.

- IPv4 applications communicating over IPv4 or IPv6 applications communicating over IPv6 are the native situations and do not need consideration here; in this case BIA simply has to perform no action.

- For an IPv4 application that needs to communicate using IPv6, the IPv4 application's addressing needs to be converted to IPv6 in order to be transmitted to the remote host. The opposite conversion has to be applied when an IPv6 application needs to communicate over IPv4.
- For an IPv6 only application on an IPv4/IPv6 host communicating with an IPv4/IPv6 remote host, the mechanism provides an optional feature to make this application able to communicate over IPv4 as well as over IPv6. If this application is trying to communicate over IPv4, the application's addressing needs to be converted to IPv4 in order to be transmitted to the remote host.

3.1 IPv6 Only Host Architecture Using BIA

IPv4 applications need IPv4 connectivity for communication. BIA is a mechanism that enables hosts that have to communicate using IPv6 to run IPv4 applications. Such hosts **MUST** have BIA installed and enabled. Figure 1 shows the architecture of the IPv6 host in which BIA is installed.

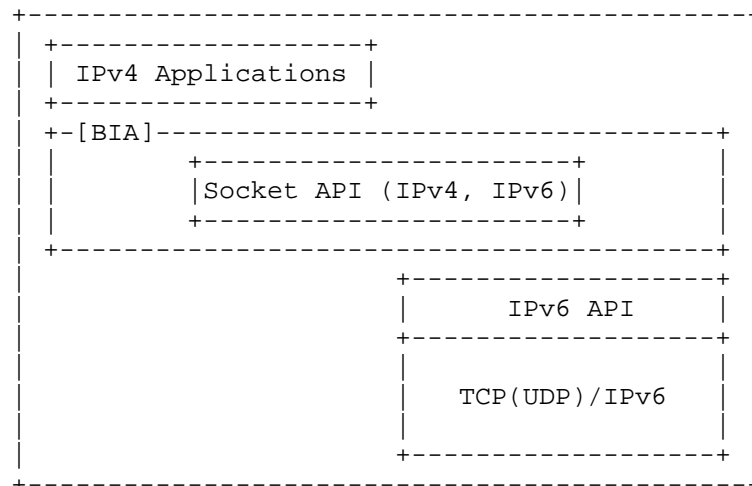


Figure 1: the architecture of IPv6 only host
In which BIA is installed.

3.2 IPv4 Only Host Architecture Using BIA

IPv4 only hosts are capable of running IPv4 applications only. BIA can be installed on such machines to allow these hosts to run IPv6 only applications as well. Figure 2 shows the host architecture of the IPv4 host in which BIA is installed.

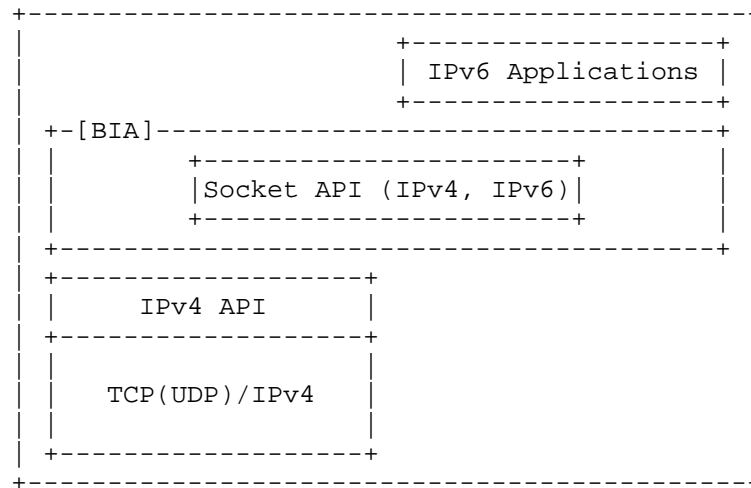


Figure 2: the architecture of IPv4 only host
In which BIA is installed.

3.3 Dual Connectivity Host Architecture Using BIA

[RFC4213] suggests that dual stack hosts need applications, dual TCP/IP modules and addresses for both IPv4 and IPv6. In such hosts, the BIA will be used only when the received DNS record(s) version is incompatible with the running of the application's IPv4/IPv6 capability. For example, if a dual connectivity host is running an IPv4 only application, and this application is going to communicate with an IPv6 only host, then the name resolver will receive the 'AAAA' record for the destination host so that the current connectivity will be IPv6, and BIA will translate the IPv4 socket API functions into IPv6 socket API functions and vice versa. BIA always will use the API functions that are compatible with the destination address. Figure 3 shows a dual connectivity host on which BIA is installed.

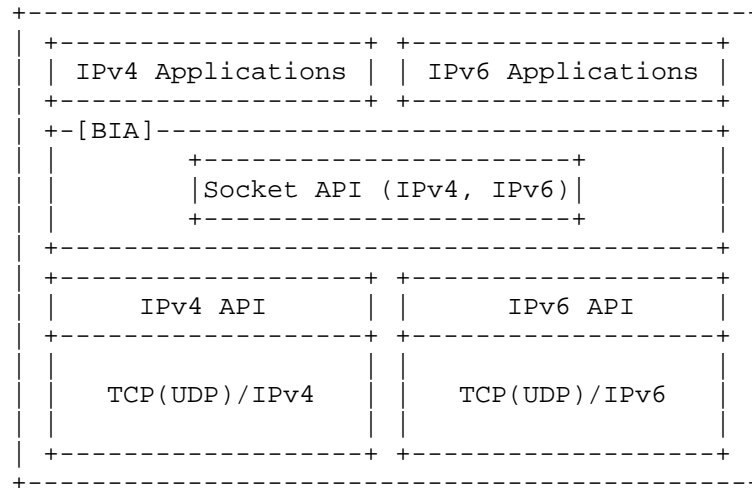


Figure 3: the architecture of dual stack host
In which BIA is installed.

4. BIA Modules

Like BIA, the API translator in BIA consists of three modules, name resolver, address resolver, and function mapper.

4.1 Name Resolver

In general the name resolver module returns a proper answer in response to the IPv4 or IPv6 application's DNS resolving request. The name resolver has different behaviors depending on the running application's IPv4/IPv6 capability and the IPv4/IPv6 connectivity.

4.1.1 IPv4 only application on the local host:

- IPv4 Only Host:

Since this is the native situation, BIA needs to perform no action.

- IPv6 Only Host

This behavior of the name resolver occurs when an IPv4 application needs to communicate using IPv6. The application will try to resolve names via the IPv4 resolver library (e.g. `gethostbyname`). BIA will call the IPv6 equivalent function (e.g. `getnameinfo`) that will resolve both 'A' and 'AAAA' records. If it got 'AAAA' record(s) only, it requests the address resolver (see below) to assign internal IPv4 address(es) corresponding to the IPv6 address(es) of the 'AAAA' record(s), then creates 'A' record(s) for the assigned IPv4 address(es), finally returns the created 'A' record(s) of the

internal address(es) to the IPv4 application. Note that this behavior is similar to the name resolver behavior in the BIA mechanism, but there are differences in the way internal addresses are assigned and managed (see address resolver further).

If both 'A' and 'AAAA' records are received, BIA will discard the 'A' record(s) as these cannot be used with IPv6, and select only the 'AAAA' record(s).

- IPv4/IPv6 Host:

The application will try to resolve names via the IPv4 resolver library (e.g. `gethostbyname`). BIA will call the IPv6 equivalent function (e.g. `getnameinfo`) that will resolve both 'A' and 'AAAA' records. If it got 'AAAA' record(s) only, it requests the address resolver (see below) to assign internal IPv4 address(es) corresponding to the IPv6 address(es) of the 'AAAA' record(s), then creates 'A' record(s) for the assigned IPv4 address(es), finally returns the created 'A' record(s) of the internal address(es) to the IPv4 application. If it got 'A' record(s) only, the communication will continue as native IPv4 communication, and BIA has to do no operation. If both 'A' and 'AAAA' records are returned, the communication can be effected natively using IPv4 using the 'A' record(s). But as an additional, optional feature, the IPv4 application can also be allowed to communicate over IPv6 with IPv6 peer(s). In that case, it requests the address resolver (see below) to assign internal IPv4 address(es) corresponding to the IPv6 address(es) of the 'AAAA' record(s), then creates 'A' record(s) for the assigned IPv4 address(es), and finally returns both the original 'A' for the remote AND the created 'A' record(s) of the internal address(es) to the IPv4 application. This extends the communication capabilities of the application to cover both IPv4 and IPv6 communication with the remote(s).

4.1.2 IPv6 Application on the local host:

- IPv6 Only Host:

Since this is the native situation, BIA needs to perform no action.

- IPv4 Only Host:

This situation occurs when an IPv6 application needs to communicate using IPv4, the application will try to resolve names via the IPv6 resolver library (e.g. `getnameinfo`). BIA will call the IPv4 equivalent function (e.g. `gethostbyname`). If it got 'A' record(s) only, it requests the address resolver to assign an internal IPv4-embedded IPv6 address(es) [I-D.draft-ietf-behave-address-format] corresponding to the IPv4 address(es), then creates 'AAAA' record(s) for the IPv4-embedded IPv6 address(es) and returns these 'AAAA' record(s) to the IPv6 application.

- IPv4/IPv6 Host:

The application will try to resolve names via the IPv6 resolver library (e.g. `gethostinfo`) that will resolve both 'A' and 'AAAA' records. If it got 'A' record(s) only, the name resolver will request the address resolver to assign internal IPv4-embedded IPv6 address(es) corresponding to the IPv4 address(es), then creates 'AAAA' record(s) for the IPv4-embedded IPv6 address(es) and returns these 'AAAA' record(s) to the application. If it got 'AAAA' records only, the communication will continue as native IPv6 communication, and BIA has to do no operation. If both 'A' and 'AAAA' records are returned, the communication can be effected natively using IPv6 using the 'AAAA' record(s). But as an additional, optional feature, the IPv6 application can also be allowed to communicate over IPv4 with IPv4 peer(s). In that case, it requests the address resolver to assign internal IPv4-embedded IPv6 address(es), then creates 'AAAA' record(s), and finally returns both the original 'AAAA' for the remote AND the created 'AAAA' record(s) of the internal address(es) to the IPv6 application. This extends the communication capabilities of the application to cover both IPv4 and IPv6 communication with the remote(s).

4.1.3 Reverse DNS lookup

For various reasons, applications may do "pointer" lookups, i.e. the application passes the IP address and expects the host name in return. BIA should be able to handle these calls. When address translation (mapping or embedding) was performed on the host IP address, the application will call with the internal address generated by BIA. The DNS call to resolve the name should obviously be made with the external address that corresponds to the translated address, and the name returned for the external address needs to be returned to the application.

4.1.4 Originating without DNS lookup

Some applications bypass the DNS lookup, and use an IP address directly instead. While often this is bad practice, in some instances this how the software is being operated. For an IPv4 only application making such call, if an address mapping was stored for the supplied IPv4 address, that mapping can be used. Otherwise, as no DNS call is made, a correspondence between IPv4 and IPv6 addresses cannot be made by the name and address resolvers, and communication using incompatible application IPv4 capability and IPv6 connectivity is impossible. But for both for IPv4 and IPv6 applications, as a last resort, a "dirty trick" can be attempted however. Using the IP address from the application, a "pointer" DNS lookup can be made. If this succeeds, a forward DNS lookup can be made on the returned name, which may return

one or more addresses of the other type required to establish the required IPv4/IPv6 address relationship. If this trick does not succeed, communication will be impossible, unless native communication is available (IPv4 over IPv4 connectivity or IPv6 over IPv6 connectivity).

It is recommended that address relationships can be manually entered in the mapping table for such occurrences. Such address mappings are in this case external IPv4-to-external IPv6 address mappings, and not relations between an internal and an external address.

4.2 Address Resolver

The address resolver is only involved with incompatibility between application IPv4/IPv6 capability and host IPv4/IPv6 connectivity. The address resolver has different behavior depending on the name resolver and function mapper requests. Like in the original BIA address mapper, the address resolver in BIA maintains a table of the pairs of an internal IPv4 address and an external IPv6 address in an IPv6 only host. These IPv4 addresses are assigned from an IPv4 address pool for internal addresses, but the mechanism for the pool is different here, as explained further.

The key difference between BIA and the BIA mechanism is the ability for the later to address all kinds of remote host connectivity (i.e. IPv4 only, IPv6 only and dual IPv4/IPv6 connectivity). Different addressing techniques are used depending on the remote host. The sending host has to take the decision either to map the destination IPv6 address into an internal IPv4 address assigned from the IPv4 address pool, or to embed the IPv4 address into an internal IPv4-embedded IPv6 address. The Address resolver in BIA can receive two possible address types-normally after calling the name resolver and querying the DNS- regarding the remote host(s) for that domain name:

- IPv6 Address: in this case it receives 'AAAA' record(s) and the address resolver has to map the IPv6 into an internal IPv4 address(es).
- IPv4 address: in this case it receives 'A' record(s) and the address resolver has to embed the IPv4 address into an internal IPv6 address(es).

4.2.1 Mapping

This technique is used when an IPv4 application needs to communicate using IPv6. It internally maintains a table of the pairs of IPv4 address(es) and IPv6 address(es). The IPv4 addresses are assigned from an IPv4 address pool. These addresses should be selected from a private IPv4 addresses as per [RFC1918] to be used by BIA for

mapping purposes (see further). When the name resolver or the function mapper requests it to assign an internal IPv4 address corresponding to an IPv6 address, it selects and returns an IPv4 address out of the pool, and registers a new entry into the table dynamically. As in the original BIA, the registration occurs in the following two cases:

1. When the name resolver gets only an 'AAAA' record for the target host name and there is not a mapping entry for the IPv6 address.
2. When the function mapper gets a socket API function call from the data received and there is not a mapping entry for the IPv6 source address. Address mappings are stored. When the address resolver is called to map an IPv6 external address into an IPv4 internal address, it will first look up the table to check whether there was a previous mapping for that address. If one is found, it will reuse and return that mapping. If not, it will create a new mapping, store that mapping and return the newly created mapping.

4.2.2 Embedding

Unlike the original BIA, BIA also allows IPv6 only applications to communicate over IPv4. Therefore a correspondence between external IPv4 addresses and internal IPv6 addresses need to be established. The proposed method is "IPv4-in-IPv6" address embedding [I-D.draft-ietf-behave-address-format]. The address resolver is configured to use one of the methodologies that are described in [I-D.draft-ietf-behave-address-format] to create an IPv4-embedded IPv6 address. The new address consists of: Network Specific Prefix (NSP) (32 bits), the IPv4 destination address (32 bits), and finally the suffix (64 bits). Figure 4 demonstrates the IPv4-embedded IPv6 address structure. As the real external IPv4 address is embedded into the internal IPv6 address, no registering is required in this case, as there is always a unique correspondence.

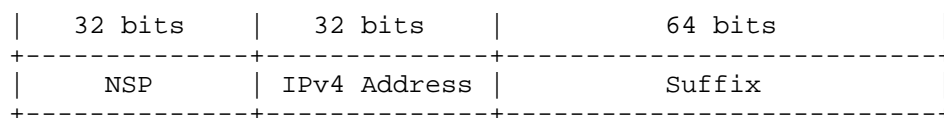


Figure 4: The structure of IPv4-embedded IPv6 address

4.3 Function Mapper

The function mapper has different behavior depending on the host connectivity. In dual connectivity hosts (IPv4 and IPv6 connectivity) the function mapper is used to decide which API functions to call in

the current communication. It is important to note that the BIA modules will be invoked just if there is an incompatibility between the running application and the connectivity type. In case of IPv6 only or IPv4 only connectivity, the main goal of the function mapper is just like in original BIA mechanism. It is used when conversion is required from IPv4 to IPv6 or the other way around between application layer and communication stack. In that case it translates the API functions used by the application into the API functions needed for the communication, and vice versa.

In dual connectivity hosts, deciding which API functions to call depends on the address type of the remote. The following is the behavior of the function mapper running on dual connectivity hosts:

1. IPv4 only application communicating over IPv6: in this case it will call the equivalent IPv6 socket API functions. The application will use the IPv4 socket API to communicate with other hosts. Since the application needs to communicate over IPv6, the function mapper intercepts the IPv4 socket API functions and calls the equivalent IPv6 socket API functions instead.

2. IPv6 only application communicating over IPv4: in this case it will call the equivalent IPv4 socket API functions. The application will use the IPv6 socket API to communicate with other hosts. Since the application needs to communicate over IPv4, the function mapper intercepts the IPv6 socket API functions and calls the equivalent IPv4 socket API functions instead.

5. Behavior Examples

The following sections will describe the behaviors of the hosts and applications that are listed in table 1.

In the following sections, the meanings of arrows are as follows:

- > A DNS message for name resolving created by the Applications and the name resolver in the API translator.
- +++> An IPv4 or IPv4-embedded IPv6 address request to and reply from the address resolver for the name resolver and the function mapper.
- ===> Data flow by API functions created by the applications and the function mapper in the API translator.

5.1 IPv4 Only Application, IPv6 Only Connectivity with an IPv6 Only Peer.

5.1.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating to IPv6 Only Peer

When an IPv4 application sends a DNS query to its name server, the name resolver intercepts the query and then creates a new query to resolve both 'A' and 'AAAA' records. When only 'AAAA' record(s) is (are) resolved, the name resolver requests the address resolver to get IPv4 address(es) corresponding to the IPv6 address(es) for each IPv6 address from the 'AAAA' record. The address resolver first looks up the table of stored entries to check if the correspondence was made previously. If yes, the stored mapping is retrieved and passed to the name resolver. If not, the address resolver creates a new mapping for an internal IPv4 address corresponding to the IPv6 external address, stores the mapping, and returns the mapping to the name resolver. The name resolver, upon receiving the internal IPv4 address(es) creates 'A' record(s) for the assigned IPv4 address(es) and returns these to the application. In order for the IPv4 application to send IPv4 packets over IPv6, it calls the IPv4 socket API function. The function mapper detects the API function call from the application. The IPv6 address is required to invoke the IPv6 socket API function, thus the function mapper requests the IPv6 address corresponding for the internal IPv4 address to the address resolver. The address resolver selects the external destination IPv6 address corresponding to the internal IPv4 address from the mapping table and returns it to the function mapper. Using this IPv6 address, the function mapper will invoke the IPv6 socket API function corresponding to the IPv4 socket API function received from the application.

When a reply is received, this will come in through the IPv6 socket API, and the function mapper requests the address resolver for the IPv4 address corresponding to the received IPv6 address. This IPv4 address will be used to translate the IPv6 socket API function call into the corresponding IPv4 socket API function call for the IPv4 application. Figure 5 illustrates the behavior described above.

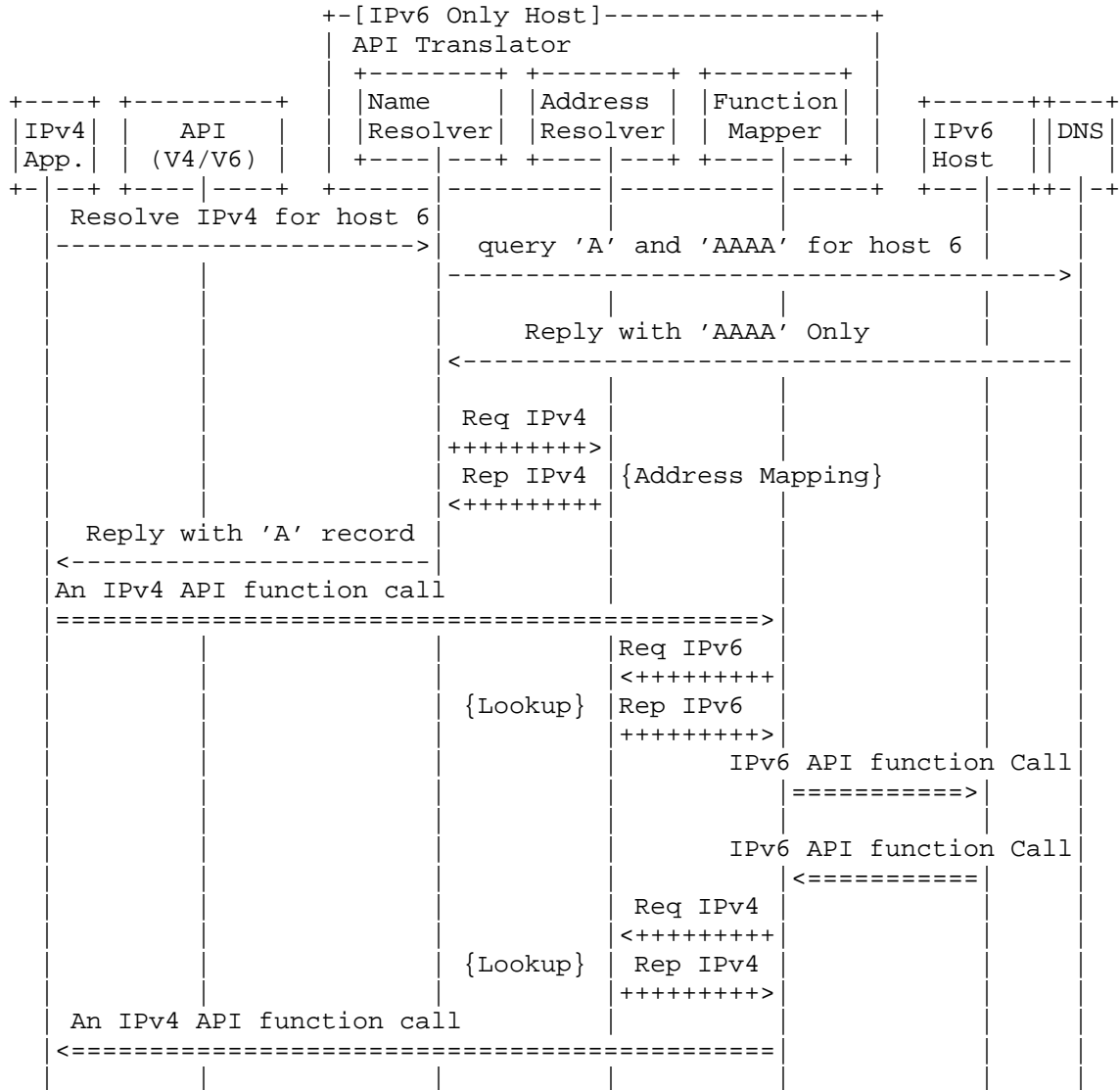


Figure 5: The behavior of the originator communicates with IPv6 Application

5.1.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host

The IPv6 originator host that started the communication to this host has resolved the address of this IPv6 host with 'AAAA' record(s)

through its name server, and has sent an IPv6 packet to this IPv6 host. The function mapper requests the internal IPv4 address corresponding to the originator's IPv6 address. The address resolver looks up the mapping table to check for an entry. If one is found, it returns the internal IPv4 address corresponding to the IPv6 address. Then the function mapper invokes the corresponding IPv4 socket API function for the IPv4 application corresponding to the IPv6 function. If not, the address resolver creates a new mapping for an internal IPv4 address corresponding to the IPv6 external address, stores the mapping, and returns the mapping to the function resolver. The remaining part of the handling is identical to what was described in 5.1.1. Figure 6 illustrates the behavior described above.

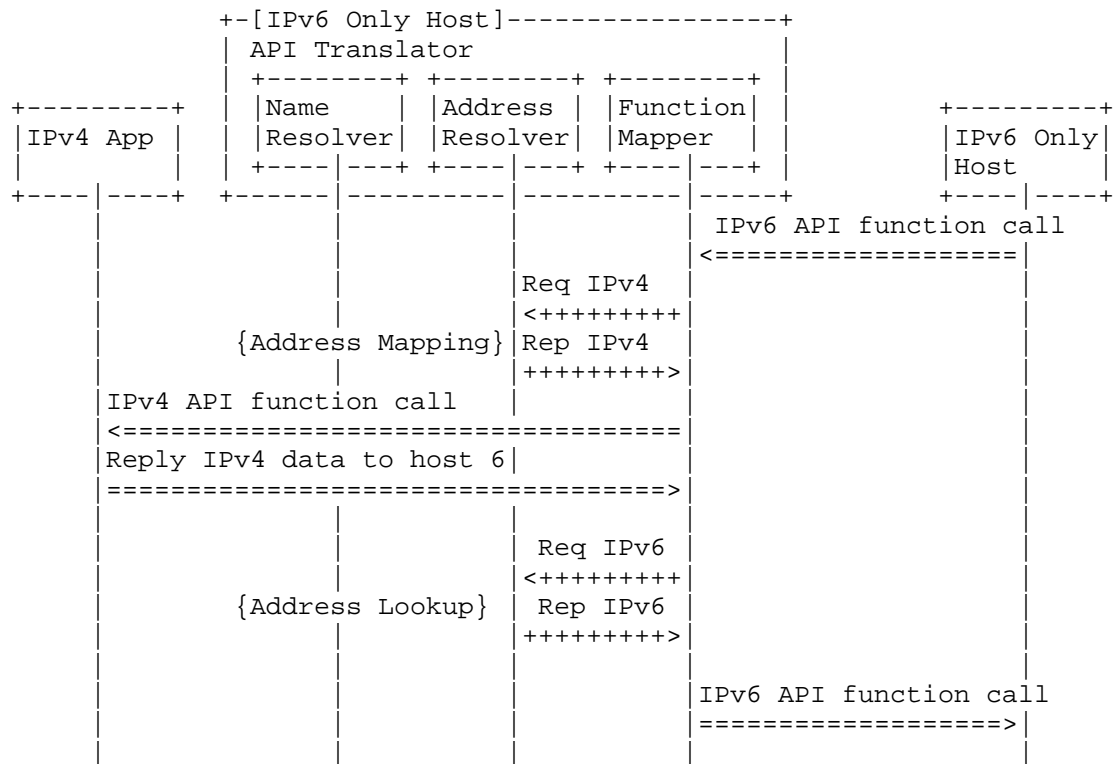


Figure 6: Behavior of receiving data from IPv6 host

5.2 IPv4 Only Application, IPv6 Only Network and Dual Connectivity Peer

5.2.1 Behavior for IPv4 Only Originator Application on IPv6 Only Host Communicating with Dual Connectivity Host

When an IPv4 application sends a DNS query to its name server, the name resolver intercepts the query and then creates a new query to resolve both 'A' and 'AAAA' records. If both 'A' and 'AAAA' records are resolved, the name resolver will only select the 'AAAA' records and drop the 'A' record(s) and requests the address resolver to assign internal IPv4 address(es) corresponding to the IPv6 address(es). The remaining behavior is exactly like described in 5.1.1.

5.2.2 Behavior for IPv4 Only Recipient Application on IPv6 Only Host Communicating with Dual Connectivity Host

Exactly the same as in section 5.1.2

5.3 IPv6 Only Application, IPv4 Only Connectivity with an IPv4 Only Peer

5.3.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with IPv4 Only Host

When an IPv6 application sends a DNS query to its name server to resolve both 'A' and 'AAAA' records, the name resolver intercepts the query and then creates a new query to resolve only 'A' record(s), since it is a IPv4 only host. With only 'A' record(s) resolved, the name resolver requests the address resolver to embed the IPv4 address(es) into IPv6 address(es) using the format described in section 4.2.2. The name resolver creates 'AAAA' record(s) for the IPv4 embedded IPv6 address(es) and returns it to the application. In order for the IPv6 application to send IPv6 packets to IPv4 only host, it calls the IPv6 socket API function. The function mapper detects the API function call from the application. The function mapper requires an IPv4 address to invoke the IPv4 socket API function, so it requests the corresponding IPv4 address to the address resolver. The address resolver extracts the destination IPv4 address from the IPv4-embedded IPv6 address and returns it to the function mapper. Using this IPv4 address, the function mapper will invoke the IPv4 socket API function corresponding to the IPv6 socket API function. We notice here the address resolver is not going to save any new records to the mapping table. When a reply is received, this will come in through the IPv4 socket API, and the function mapper requests the address resolver for the IPv6 address corresponding to the received IPv4 address. This IPv6 address will be used to translate the IPv4 socket API function call

into the corresponding IPv6 socket API function call for the IPv6 application. Figure 7 illustrates the behavior described above.

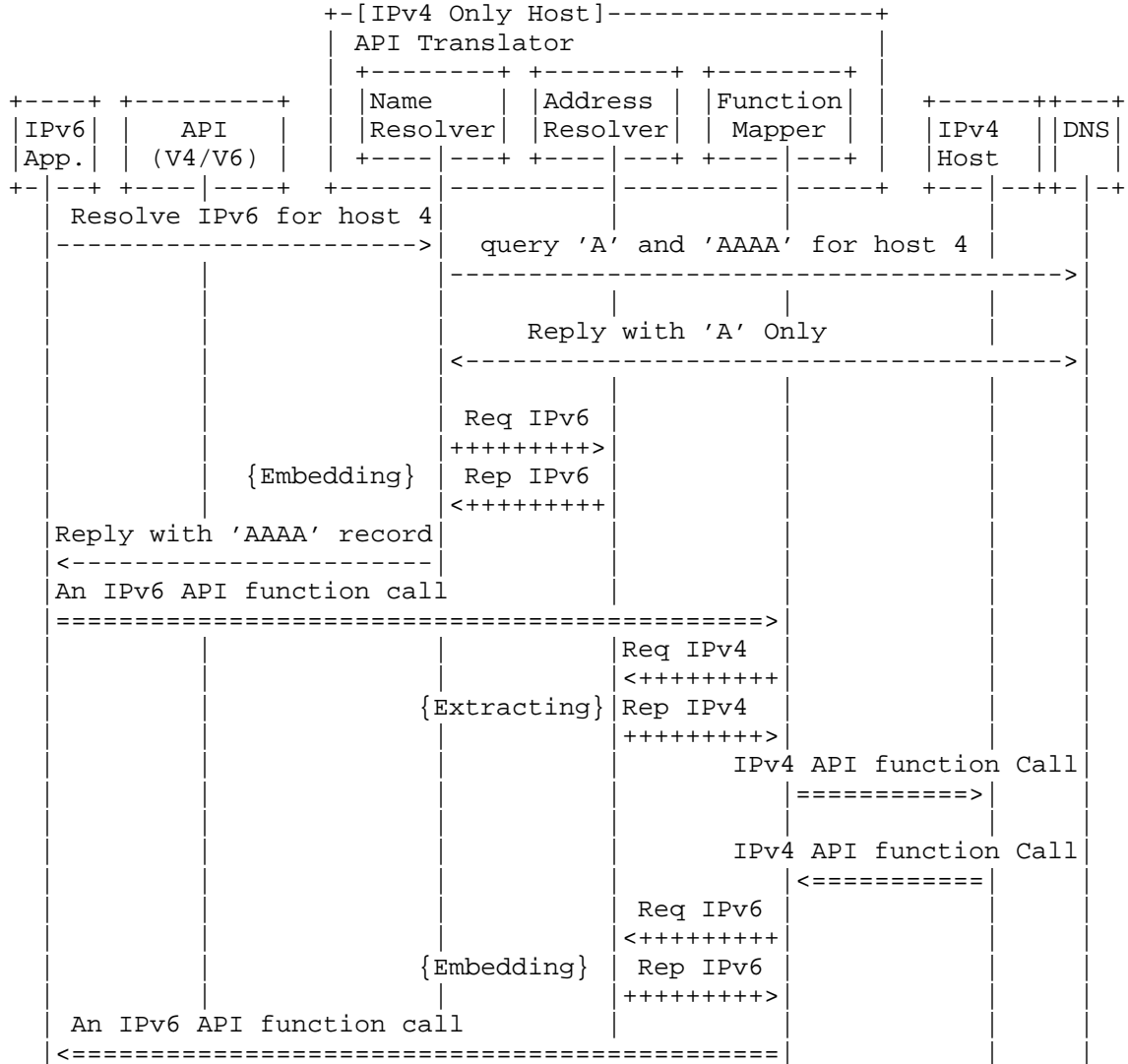


Figure 7: The behavior of the originator communicates with IPv4 Application

5.3.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with IPv4 Only Host

The IPv4 originator host has resolved the address of this IPv4 host with 'A' records through its name server, and has sent an IPv4 packet to this IPv4 host. The function mapper requests the IPv6 address to the address resolver in order to invoke the IPv6 socket API function to communicate with the IPv6 application. The address resolver embeds the IPv4 address(es) into IPv6 address(es) using the format described in section 4.2.2, and returns this address. Then the function mapper invokes the corresponding IPv6 socket API function for the IPv6 application corresponding to the IPv4 function.

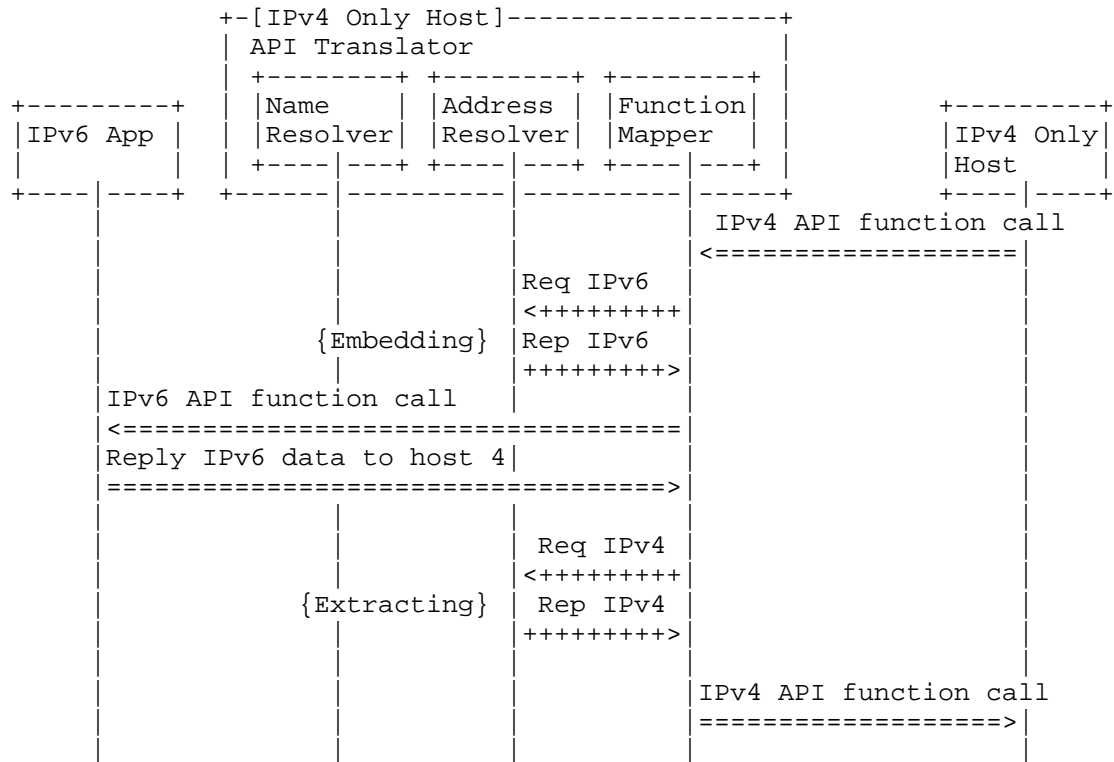


Figure 8: Behavior of receiving data from IPv4 host

5.4 IPv6 Only Application, IPv4 Only Network and Dual Connectivity Peer

5.4.1 Behavior for IPv6 Only Originator Application on IPv4 Only Host Communicating with Dual Connectivity Host

When an IPv6 application sends a DNS query to its name server, the name resolver intercepts the query and then creates a new query to resolve 'A' record(s); no 'AAAA' record(s) are returned, as it is an IPv4 only host. When 'A' record(s) is/are resolved, the name resolver will request the address resolver to embed the IPv4 address(es) into IPv6 address(es) using the format that is describes in section 4.2.2. The remaining processing is as in 5.3.1.

5.4.2 Behavior for IPv6 Only Recipient Application on IPv4 Only Host Communicating with Dual Connectivity Host

Exactly the same as in section 5.3.2

5.5 IPv4 Only Application on Dual Connectivity Host Communicating to IPv6 Only Peer

5.5.1 IPv4 Only Originator Application on Dual Connectivity Host Communicating to IPv6 Only Peer

Exactly the same as in section 5.1.1

5.5.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv6 Only Peer

Exactly the same as in section 5.1.2

5.6 IPv6 Only Application on Dual Connectivity Host Communicating to IPv4 Only Peer

5.6.1 IPv6 Only Originator Application on Dual Connectivity Host Communicating to IPv4 Only Peer

Exactly the same as in section 5.3.1

5.6.2 IPv4 Only Recipient Application on Dual Connectivity Host Communicating to IPv4 Only Peer

Exactly the same as in section 5.3.2

6. Considerations

6.1 Socket API Conversion

This document uses the socket API conversion specified in [RFC3338].

6.2 Address Mapping and Embedding

There are some considerations as to the choice and management of the internal addresses:

- For a diversity of reasons, several applications store the addresses of machines they have been communicating with, and check these addresses on next contact. It is hence important that each mapping of external IPv6 to internal IPv4 addresses is being stored by BIA, so as to always use the same internal address for a particular external address at the next communication. This implies a very wide IPv4 address range available for mapping. The authors propose a Class A range, making approximately 16Mega addresses available. Even that can get exhausted in some cases, so this range should be supplemented by a round-robin scheme where, in case of exhaustion, the mappings that remain unused for the longest time can be reused for new mappings (not the creation time, but the last time that mapping was actively used is important). It is considered that this would be sufficient in about all operational situations. As this mapping list potentially can become very large, the store/retrieval mechanism implementation should be optimized for speed or it may introduce unacceptable long delays. This is an implementation issue however, and will not be dealt with here.
- It is obvious that an IPv4/IPv6 correspondence will be frequently required in the course of a communication; short time caching seems essential to avoid having to look up the correspondence again during the course of a communication.
- A machine having both IPv4 and IPv6 connectivity will be using both IPv4 and IPv6 addresses. To avoid conflicts, it is essential that the internal addresses used will never be used as an external address. Original BIA proposed the use of a limited (256 addresses) range as a "pool" in an "unassigned" IPv4 address range. The limited size (256) is much too small for operational purposes, even not considering the requirement for storing the mappings as described in the previous paragraph, as a machine may have more than this number of mappings active concurrently. Taking into account the requirement for storing the mappings, a very large range of unassigned addresses is required.

6.3 ICMP Message Handling

When an application needs ICMP messages values (e.g., Type, Code, etc.) sent from a network layer, ICMPv4 message values MAY be translated into ICMPv6 message values based on SIIT [I-D.ietf-behave-v6v4-xlate], and vice versa. It can be implemented using raw socket.

6.4 Implementation Issues

This document uses the implementation issues specified in [RFC3338].

7. Limitations

This mechanism supports unicast communications only. In order to support multicast functions, some other additional functionalities must be considered in the function mapper module.

Since the IPv6 socket API has new advanced features, it is difficult to translate such kinds of IPv6 socket APIs into IPv4 socket APIs. Thus, IPv6 inbound communication with advanced features may be discarded.

It should be noted that the original BIA assumes the hosts have compatible network connectivity. The new version of the BIA is developed to support the heterogeneity between connectivity and applications only, NOT incompatible network connectivity. Communication between hosts with incompatible connectivity (IPv4 only connectivity to IPv6 only connectivity, or the other way around) cannot be handled by BIA, and other solutions need to be applied, e.g. protocol translation mechanisms PNAT [I-D.draft-huang-behave-pnat], NAT64 [I-D.ietf-behave-v6v4-xlate-stateful], or SIIT [I-D.ietf-behave-v6v4-xlate].

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

This document uses the security considerations specified in [RFC3338].

10. References

10.1 Normative References

- [RFC3338] Lee, S., Shin, M-K., Kim, Y-J., Nordmark, E., and A. Durand, "Dual Stack Hosts Using "Bump-in-the-API" (BIA)", RFC 3338, October 2002.
- [RFC2460] Deering, S., and R., Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.
- [RFC2767] Tsuchiya, K., HIGUCHI, H., and Y. Atarashi, "Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)", RFC 2767, February 2000.
- [I-D.draft-huang-behave-rfc3338bis] Huang, B., Deng, H., and T. Savolainen, "Dual Stack Hosts Using "Bump-in-the-API" (BIA)", draft-huang-behave-rfc3338bis-02 (work in progress), March 2010.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.
- [I-D.draft-ietf-behave-address-format] Huitema, C., "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-10 (work in progress), August 2010.

[I-D.draft-huang-behave-pnat]

Huang, B., and H., Deng, "Prefix NAT: Host based IPv6 translation", draft-huang-behave-pnat-01 (work in progress), February 2010.

[I-D.ietf-behave-v6v4-xlate-stateful]

Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 2010.

[I-D.ietf-behave-v6v4-xlate]

Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", draft-ietf-behave-v6v4-xlate-23 (work in progress), September 2010.

10.2 Informative References

- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

Authors' Addresses

Ala Hamarsheh

Electronics and Informatics Department ETRO/Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Brussels, Belgium

Tel: +32 2 629 2930

Fax: +32 2 629 2883

Email: ala.hamarsheh@vub.ac.be

Prof. Marnix Goossens

Electronics and Informatics Department ETRO/Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Brussels, Belgium

Tel: +32 2 629 2987

Fax: +32 2 629 2883

Email: marnix.goossens@vub.ac.be

Internet Engineering Task Force
Internet-Draft
Intended status: BCP
Expires: April 21, 2011

I. Yamagata
S. Miyakawa
NTT Communications
A. Nakagawa
Japan Internet Exchange (JPIX)
H. Ashida
iTSCOM
October 18, 2010

Common requirements for IP address sharing schemes
draft-ietf-behave-lsn-requirements-00

Abstract

This document defines common requirements of multiple types of Large Scale Network Address Translation (NAT) that handles Unicast UDP, TCP and ICMP.

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, 2011.

Copyright Notice

Copyright (c) 2010 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. Terminology	3
3. The policy of assignment of LSN external IP address, port and identifier	5
4. Requirements for UDP	7
5. Requirements for TCP	8
6. Requirements for ICMP	9
7. Identifying particular users (BOTs, spammers, etc)	9
7.1. Store Translation Log	9
7.2. Fixed port assignment	10
8. Considerations about limiting the number of LSN external ports	10
9. IANA Considerations	11
10. Security Considerations	11
11. Acknowledgements	11
12. References	11
12.1. Normative References	11
12.2. Informative Reference	12
Authors' Addresses	12

1. Introduction

Now there are several IPv4 address sharing schemes such as Large Scale NAT (as known as NAT444[I-D.shirasaki-nat444-isp-shared-addr]), DS-Lite[I-D.ietf-softwire-dual-stack-lite], A+P[I-D.ymbk-aplusp] and so on under the discussion.

Those IPv4 address sharing schemes are intended to be used in the middle of the ISP access network against IPv4 address shortage problem by sharing one global IPv4 address by multiple users. Authors believe that there are common requirements among all IPv4 address sharing schemes to make them "transparent" as much as possible. At the BEHAVE working group of IETF, following RFCs have already defined to achieve maximum transparency at the residential CPE which has NAT function;

- RFC4787 : NAT Behavioral Requirements for Unicast UDP
- RFC5382 : NAT Behavioral Requirements for TCP
- RFC5508 : NAT Behavioral Requirements for ICMP

However so, because those RFCs are mainly aimed at residential CPE and any IPv4 address sharing schemes are a bit different from it, we believe that requirements for LSN and other schemes should be defined alternatively to those RFCs.

2. Terminology

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

Readers are expected to be familiar with [RFC4787] and the terms defined there. The following term are used in this document:

Large-Scale NAT(LSN): NAT devices placed between CPE and public Internet by an operator. LSN converts CPE IP Address, CPE Port, and CPE Identifier into LSN external IP Address, LSN external Port and LSN external Identifier in communication between CPE and GGN external.

LSN external realm: The realm where IPv4 global addresses are assigned

LSN internal realm: The realm placed between LSN and CPEs

LSN external IP address: The IP address on LSN in LSN external realm mapping to CPE IP address

LSN external port: The port on LSN in LSN external realm mapping to CPE port

LSN external identifier: The identifier of ICMP on LSN in LSN external realm mapping to CPE identifier

Customer Premises Equipment(CPE): The terminal which is placed in LSN internal realm and may establish TCP sessions to LSN external realm (e.g. a single PC or NATBox)

CPE IP address: The IP address on CPE in LSN internal realm

CPE port: The port on CPE in LSN internal realm

CPE identifier: CPE's identifier of ICMP in LSN internal realm

CPE 3-tuple: The tuple of TCP/UDP, CPE IP address, and CPE Port
Service Server (SS) The server an operator supplies various services for CPE

Service Server (SS): The server placed in external realm

Service Provide Server (SPS): The server placed in external realm and controlled by LSN administrators

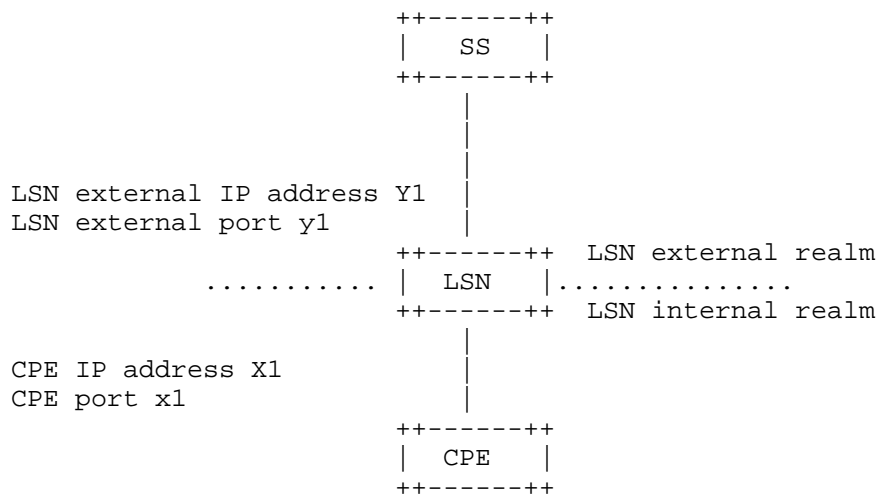


Figure 1. LSN network

3. The policy of assignment of LSN external IP address, port and identifier

A LSN has a pool of LSN external IP addresses, ports and identifiers. CPEs share LSN external IP addresses. Each LSN occupies combination of LSN external IP address, LSN external port and LSN external identifier exclusively. For a fair use of limited resources, LSN has a limitation for the number of the LSN external ports per CPE. LSNs need to keep high transparency to continue existing services after LSN is introduced. Requirement of high transparency for LSN leads to high scalability of LSN. High transparency means LSN basically keeps communications among CPEs except effect of limitations of the number of LSN external ports and TCP sessions.

A CPE MAY apply UDP hole punching or TCP hole punching for interactive services among CPEs like Voice over IP and P2P. LSN SHOULD NOT interfere in services using UDP hole punching or TCP hole punching.

REQ-1: A LSN MUST allocate one external IP address to each CPE.

- a) LSN external IP address allocated to the CPE MUST be same for the UDP, TCP and ICMP.

Justification: If a LSN allocates multiple LSN external IP addresses to each CPE, some applications might not work.

REQ-2: A LSN MUST allocate LSN external ports which is mapped for CPE ports of UDP.

- a) A LSN MUST NOT overload LSN external port while a NAT UDP mapping timer does not expire.
- b) A LSN MAY reuse LSN external port after a NAT UDP mapping timer expires.
- c) A LSN SHOULD limit the number of the LSN external ports of UDP per CPE.
- d) The number of the LSN external ports of UDP per CPE which LSN can allocate SHOULD be configurable for the administrator of LSN.

Justification: CPEs can communicate to CPE external realm fairly by limiting the number of LSN external ports per CPE.

REQ-3: A LSN MUST allocate LSN external ports which is mapped for CPE ports of TCP.

- a) A LSN MUST NOT overload LSN external port while the port is allocated for one or more TCP sessions originated by another CPE.
- b) A LSN MAY reuse LSN external port while the port is allocated for no session originated by any CPE.
- c) A LSN SHOULD limit the number of the LSN external ports of TCP per CPE.
- d) The number of the LSN external ports of TCP per CPE SHOULD be an administratively configurable option.
- e) A LSN SHOULD limit the number of the new sessions of TCP per time unit and per CPE.

Justification: CPEs can communicate to CPE external realm fairly by limiting the number of LSN external ports per CPE. In addition, TCP LSN external port MAY have TCP sessions, and therefore the TCP session timer is necessary for every 5-Tuple. LSN can have not only the limitations of the number of LSN external ports but also TCP sessions per CPE. Thus a LSN can prevent denial of service attacks with the tons of TCP open and close by malicious CPEs.

REQ-4: A LSN MUST allocate LSN external identifiers which is mapped for CPE identifiers of ICMP.

- a) A LSN MUST NOT overload LSN external identifier before an ICMP Query session timer expires.
- b) A LSN MAY reuse LSN external identifier after an ICMP Query session timer expires.
- c) A LSN SHOULD limit the number of the LSN external identifier allocated per CPE.
- d) The number of the LSN external identifiers per CPE which LSN can allocate SHOULD be an administratively configurable option.

Justification: CPEs can communicate to CPE external realm fairly by limiting the number of LSN external identifiers every CPE.

If a CPE has already consumed many LSN external ports, the CPE might not use new ports because LSNs limit the number of ports.

REQ-5: A LSN MAY have implementations that some specific applications

can work well even if each CPE's usable number of LSN external ports have already consumed.

Justification: Some specific applications don't work well due to limitation of number of number of ports by LSN, therefore other applications might be affected in the same CPE.

In Section 8 we discuss in detail.

4. Requirements for UDP

[RFC4787] describes requirements of the Unicast UDP of a NAT, and the behavior of "Endpoint-Independent Filtering "is RECOMMENDED, and a NAT MUST have an "Endpoint-Independent Mapping" behavior to ensure transparency of LSN.

To have "Endpoint-Independent Filtering" and "Endpoint-Independent Mapping" behaviors for LSNs, LSNs help to establish UDP Hole Punching among CPEs. In other words, the possibility of the establishment of UDP Hole Punching among CPEs which have LSN is equal to the possibility among CPEs which don't have LSN. If LSNs have an "Address-Dependent Mapping" or "Address and Port-Dependent Mapping" behavior, the possibility that establishment of UDP Hole Punching is less than when LSNs have an "Endpoint-Independent Mapping" behavior. If LSNs have an "Address and Port-Dependent Filtering" behavior, the possibility that establishment of UDP Hole Punching is less than when LSNs have an "Endpoint-Independent Filtering" or "Address Dependent Filtering" behavior.

If a LSN supports NAT Hairpinning, a CPE can communicate other CPEs in LSN internal realm of the same LSN.

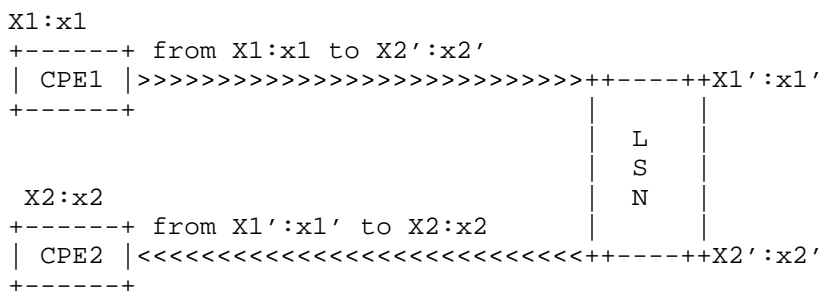


Figure 2. Hairpinning

REQ-6: A LSN SHOULD comply with [RFC4787] for unicast UDP, unless a) applies.

a) It is RECOMMENDED that a NAT have "Endpoint-Independent Filtering" behavior.

Status: "If application transparency is most important, it is RECOMMENDED that a NAT have Endpoint-Independent Filtering behavior. If a more stringent filtering behavior is most important, it is RECOMMENDED that a NAT have Address-Dependent Filtering behavior." is written at REQ-8 in RFC4787.

Justification: LSN which is placed at ISP/Carrier makes much of transparency. In particular, for applications that receive media simultaneously from multiple locations (e.g., gaming), or applications that use rendezvous techniques. But to be more precise, in the LSN case, it may not be needed for some specific protocol such as DNS query and response.

5. Requirements for TCP

[RFC5382] describes requirements of TCP of a NAT, and the behavior of "Endpoint-Independent Filtering" is RECOMMENDED, and a NAT MUST have an "Endpoint-Independent Mapping" behavior to ensure transparency of LSN

To have "Endpoint-Independent Filtering" and "Endpoint-Independent Mapping" behaviors for LSNs, LSNs help to establish TCP Hole Punching among CPEs. In other words, the possibility of the establishment of TCP Hole Punching among CPEs which have LSN is equal to the possibility among CPEs which don't have LSN. If LSNs have an "Address-Dependent Mapping" or "Address and Port-Dependent Mapping" behavior, the possibility that establishment of TCP Hole Punching is less than when LSNs have an "Endpoint-Independent Mapping" behavior. If LSNs have an "Address and Port-Dependent Filtering" behavior, the possibility that establishment of TCP Hole Punching is less than when LSNs have an "Endpoint-Independent Filtering" or "Address Dependent Filtering" behavior.

If a LSN supports NAT Hairpinning, a CPE can communicate other CPEs in LSN internal realm of the same LSN.

REQ-7: A LSN SHOULD comply with [RFC5382] for TCP, unless a) applies.

a) It is RECOMMENDED that a NAT have an "Endpoint independent filtering" behavior for TCP.

Status: "If application transparency is most important, it is RECOMMENDED that a NAT have an "Endpoint independent filtering" behavior for TCP. If a more stringent filtering behavior is most important, it is RECOMMENDED that a NAT have an "Address dependent filtering" behavior." is REQ-3 in RFC5382.

Justification: LSN which is placed at ISP/Carrier makes much of transparency. But to be more precise, in the LSN case, it may not be needed for some specific protocols.

6. Requirements for ICMP

[RFC5508] describes requirements of ICMP of a NAT. And there MAY be a case that CPE cannot establish communication from CPEs to LSN external realm because LSN limits the number of LSN external ports, identifiers and TCP sessions per CPE. It is useful if CPE can distinguish an error to occur by the limitation of the LSN external ports, identifiers and TCP sessions from other errors.

REQ-8: A LSN SHOULD comply with [RFC5508] for ICMP.

Justification: LSN SHOULD have to keep high transparency for ICMP. And CPE MAY use P2P and interactive services between CPEs after a LSN is introduced.

Therefore, written in [RFC5508], when a LSN can't establish new session of TCP/UDP by limiting of TCP/UDP ports per user, the LSN sends an ICMP destination unreachable message, with code of 13 (Communication administratively prohibited) to the sender.

7. Identifying particular users (BOTs, spammers, etc)

It is necessary for network administrators to identify a user from an IP address and a timestamp in order to deal with abuse and lawful intercept. When multiple users share one external address at LSN, the source address and the source port that are visible at the destination host are translated ones. The following mechanisms can be used to identify the user that transmitted a certain packet.

7.1. Store Translation Log

One mechanism stores the following information at LSN.

- destination address

- destination port
- translated source address
- translated source port
- untranslated source address
- untranslated source port
- timestamp

In such environment that one LSN accommodates a lot of users or processes large amount of traffic, the amount of log will be so large and the operator has to prepare large volume of storage.

7.2. Fixed port assignment

To save costs for storage, one can adopt this port assignment mechanism at LSN. By fixing the range of external port per user/CPE, and having the mapping of internal IP address to external IP address and port, there will be no need to store per session log. Note that this mechanism is possible only if the source port is known as well as the source address, the destination address and the destination port.

8. Considerations about limiting the number of LSN external ports

As discussed in section 3,4 and 5, LSN limits the number of LSN external ports and identifier per CPE. Therefore some important applications like DNS might not work well due to applications consuming many LSN external ports.

There are two ways to solve this issue. The one is that particular applications are out of the targets for the number of port limitation for LSN. In the case, the applications should be configurable for the administrator of LSN.

The other is that LSN doesn't translate address or port for some specific applications, moreover it doesn't limit the number of LSN external ports.(we call "LSN pass-through") Therefore, LSN behave as a router. In this case, some specific applications are out of limitation for the number of LSN external ports. Some applications, which don't work well due to address translation like FTP, is effective. Reducing costs of translation is also effective. As a condition, administrators of LSN can control SPS which become a target of LSN pass-through.

Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, April 2009.
- [I-D.shirasaki-nat444-isp-shared-addr]
Shirasaki, Y., Miyakawa, S., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444 addressing models", draft-shirasaki-nat444-isp-shared-addr-04 (work in progress), July 2010.

12.2. Informative Reference

- [I-D.ietf-softwire-dual-stack-lite]
Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", draft-ietf-softwire-dual-stack-lite-06 (work in progress), August 2010.
- [I-D.ymbk-aplusp]
Bush, R., "The A+P Approach to the IPv4 Address Shortage", draft-ymbk-aplusp-05 (work in progress), October 2009.

Authors' Addresses

Ikuhei Yamagata
NTT Communications Corporation
Gran Park Tower 17F, 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Phone: +81 50 3812 4704
Email: ikuhei@nttv6.jp

Shin Miyakawa
NTT Communications Corporation
Gran Park Tower 17F, 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Phone: +81 50 3812 4695
Email: miyakawa@nttv6.jp

Akira Nakagawa
Japan Internet Exchange Co., Ltd. (JPIX)
Otemachi Building 21F, 1-8-1 Otemachi, Chiyoda-ku
Tokyo 100-0004
Japan

Phone: +81 90 9242 2717
Email: a-nakagawa@jpix.ad.jp

Hiroyuki Ashida
its communications Inc.
541-1 Ichigao-cho Aoba-ku
Yokohama 225-0024
Japan

Email: ashida@itscom.ad.jp

Internet Engineering Task Force
Internet-Draft
Updates: 4787 (if approved)
Intended status: BCP
Expires: June 9, 2013

S. Perreault, Ed.
Viagenie
I. Yamagata
S. Miyakawa
NTT Communications
A. Nakagawa
Japan Internet Exchange (JPIX)
H. Ashida
IS Consulting G.K.
December 6, 2012

Common requirements for Carrier Grade NATs (CGNs)
draft-ietf-behave-lsn-requirements-10

Abstract

This document defines common requirements for Carrier-Grade NAT (CGN). It updates RFC 4787.

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 June 9, 2013.

Copyright Notice

Copyright (c) 2012 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. Terminology	3
3. Requirements for CGNs	5
4. Logging	10
5. Port Allocation Scheme	11
6. Deployment Considerations	12
7. IANA Considerations	12
8. Security Considerations	12
9. Acknowledgements	13
10. References	13
10.1. Normative References	13
10.2. Informative Reference	14
Authors' Addresses	15

1. Introduction

With the shortage of IPv4 addresses, it is expected that more Internet Service Providers (ISPs) may want to provide a service where a public IPv4 address would be shared by many subscribers. Each subscriber is assigned a private address, and a Network Address Translator (NAT) [RFC2663] situated in the ISP's network translates between private and public addresses. When a second IPv4 NAT is located at the customer edge, this results in two layers of NAT.

This service can conceivably be offered alongside others, such as IPv6 services or regular IPv4 service assigning public addresses to subscribers. Some ISPs started offering such a service long before there was a shortage of IPv4 addresses, showing that there are driving forces other than the shortage of IPv4 addresses. One approach to CGN deployment is described in [RFC6264].

This document describes behavior that is required of those multi-subscriber NATs for interoperability. It is not an IETF endorsement of CGN or a real specification for CGN, but rather just a minimal set of requirements that will increase the likelihood of applications working across CGNs.

Because subscribers do not receive unique IPv4 addresses, Carrier Grade NATs introduce substantial limitations in communications between subscribers and with the rest of the Internet. In particular, it is considerably more involved to establish proxy functionality at the border between internal and external realms. Some applications may require substantial enhancements, while some others may not function at all in such an environment. Please see "Issues with IP Address Sharing" [RFC6269] for details.

This document builds upon previous works describing requirements for generic NATs [RFC4787][RFC5382][RFC5508]. These documents, and their updates if any, still apply in this context. What follows are additional requirements, to be satisfied on top of previous ones.

2. Terminology

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

Readers are expected to be familiar with "NAT Behavioral Requirements for Unicast UDP" [RFC4787] and the terms defined there. The following additional term is used in this document:

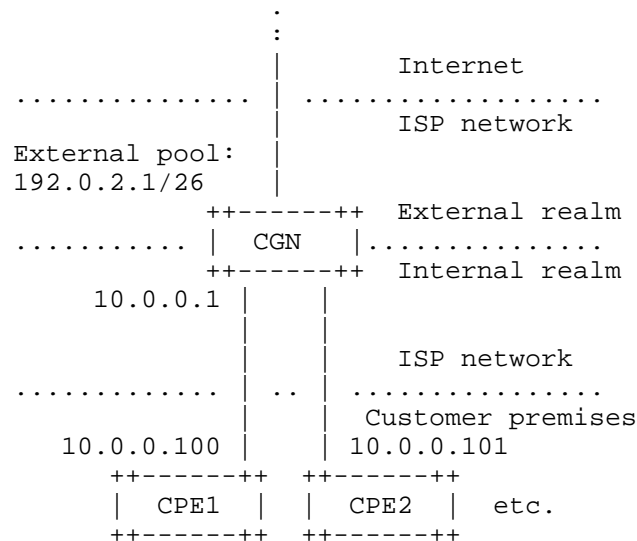
Carrier-Grade NAT (CGN): A NAT-based [RFC2663] logical function used to share the same IPv4 address among several subscribers. A CGN is not managed by the subscribers.

Note that the term "carrier-grade" has nothing to do with the quality of the NAT; that is left to discretion of implementers. Rather, it is to be understood as a topological qualifier: the NAT is placed in an ISP's network and translates the traffic of potentially many subscribers. Subscribers have limited or no control over the CGN, whereas they typically have full control over a NAT placed on their premises.

Note also that the CGN described in this document is IPv4-only. IPv6 address translation is not considered.

However, the scenario in which the IPv4-only CGN logical function is used may include IPv6 elements. For example, DS-Lite [RFC6333] uses an IPv4-only CGN logical function in a scenario making use of IPv6 encapsulation. Therefore, this document would also apply to the CGN part of DS-Lite.

Figure 1 summarizes a common network topology in which a CGN operates.



(IP addresses are only for example purposes)

Figure 1: CGN network topology

Another possible topology is one for hotspots, where there is no customer premise or customer-premises equipment (CPE), but where a CGN serves a bunch of customers who don't trust each other and hence fairness is an issue. One important difference with the previous topology is the absence of a second layer of NAT. This, however, has no impact on CGN requirements since they are driven by fairness and robustness in the service provided to customers, which applies in both cases.

3. Requirements for CGNs

What follows is a list of requirements for CGNs. They are in addition to those found in other documents such as [RFC4787], [RFC5382], and [RFC5508].

REQ-1: If a CGN forwards packets containing a given transport protocol, then it MUST fulfill that transport protocol's behavioral requirements. Current applicable documents are as follows:

- A. "NAT Behavioral Requirements for Unicast UDP" [RFC4787]
- B. "NAT Behavioral Requirements for TCP" [RFC5382]
- C. "NAT Behavioral Requirements for ICMP" [RFC5508]
- D. "NAT Behavioral Requirements for DCCP" [RFC5597]

Any future NAT behavioral requirements documents for IPv4 transport protocols will impose additional requirements for CGNs on top of those stated here.

Justification: It is crucial for CGNs to maximize the set of applications that can function properly across them. The IETF has documented the best current practices for UDP, TCP, ICMP, and DCCP.

REQ-2: A CGN MUST have a default "IP address pooling" behavior of "Paired" (as defined in [RFC4787] section 4.1). A CGN MAY provide a mechanism for administrators to change this behavior on an application protocol basis.

- * When multiple overlapping internal IP address ranges share the same external IP address pool (e.g., DS-Lite [RFC6333]), the "IP address pooling" behavior applies to mappings between external IP addresses and internal subscribers rather than between external and internal IP

addresses.

Justification: This stronger form of REQ-2 from [RFC4787] is justified by the stronger need for not breaking applications that depend on the external address remaining constant.

Note that this requirement applies regardless of the transport protocol. In other words, a CGN must use the same external IP address mapping for all sessions associated with the same internal IP address, be they TCP, UDP, ICMP, something else, or a mix of different protocols.

The justification for allowing other behaviors is to allow the administrator to save external addresses and ports for application protocols that are known to work fine with other behaviors in practice. However, the default behavior MUST be "Paired".

REQ-3: The CGN function SHOULD NOT have any limitations on the size nor the contiguity of the external address pool. In particular, the CGN function MUST be configurable with contiguous or non-contiguous external IPv4 address ranges.

Justification: Given the increasing rarity of IPv4 addresses, it is becoming harder for an operator to provide large contiguous address pools to CGNs. Additionally, operational flexibility may require non-contiguous address pools for reasons such as differentiated services, routing management, etc.

The reason for having SHOULD instead of MUST is to account for limitations imposed by available resources as well as constraints imposed for security reasons.

REQ-4: A CGN MUST support limiting the number of external ports (or, equivalently, "identifiers" for ICMP) that are assigned per subscriber.

- A. Per-subscriber limits MUST be configurable by the CGN administrator.
- B. Per-subscriber limits MAY be configurable independently per transport protocol.
- C. Additionally, it is RECOMMENDED that the CGN include administrator-adjustable thresholds to prevent a single subscriber from consuming excessive CPU resources from the CGN (e.g., rate limit the subscriber's creation of new mappings).

Justification: A CGN can be considered a network resource that is shared by competing subscribers. Limiting the number of external ports assigned to each subscriber mitigates the DoS attack that a subscriber could launch against other subscribers through the CGN in order to get a larger share of the resource. It ensures fairness among subscribers. Limiting the rate of allocation mitigates a similar attack where the CPU is the resource being targeted instead of port numbers, however this requirement is not a MUST because it is very hard to explicitly call out all CPU-consuming events.

REQ-5: A CGN SHOULD support limiting the amount of state memory allocated per mapping and per subscriber. This may include limiting the number of sessions, the number of filters, etc., depending on the NAT implementation.

A. Limits SHOULD be configurable by the CGN administrator.

B. Additionally, it SHOULD be possible to limit the rate at which memory-consuming state elements are allocated.

Justification: A NAT needs to keep track of TCP sessions associated to each mapping. This state consumes resources for which, in the case of a CGN, subscribers may compete. It is necessary to ensure that each subscriber has access to a fair share of the CGN's resources. Limiting the rate of allocation is intended to prevent CPU resource exhaustion. Item "B" is at the SHOULD level to account for the fact that means other than rate limiting may be used to attain the same goal.

REQ-6: It MUST be possible to administratively turn off translation for specific destination addresses and/or ports.

Justification: It is common for a CGN administrator to provide access for subscribers to servers installed in the ISP's network in the external realm. When such a server is able to reach the internal realm via normal routing (which is entirely controlled by the ISP), translation is unneeded. In that case, the CGN may forward packets without modification, thus acting like a plain router. This may represent an important efficiency gain.

Figure 2 illustrates this use-case.

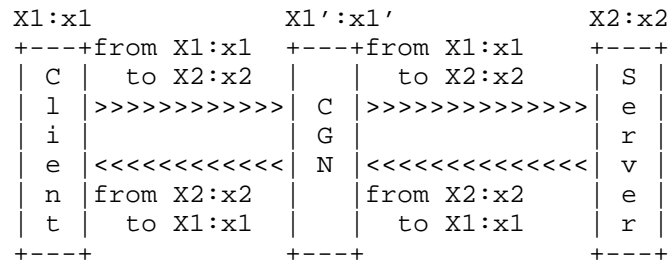


Figure 2: CGN pass-through

REQ-7: It is RECOMMENDED that a CGN use an "Endpoint-Independent Filtering" behavior (as defined in [RFC4787] section 5). If it is known that "Address-Dependent Filtering" does not cause the application-layer protocol to break (how to determine this is out of scope for this document), then it MAY be used instead.

Justification: This is a stronger form of REQ-8 from [RFC4787]. This is based on the observation that some games and peer-to-peer applications require EIF for the NAT traversal to work. In the context of a CGN it is important to minimize application breakage.

REQ-8: Once an external port is deallocated, it SHOULD NOT be reallocated to a new mapping until at least 120 seconds have passed, with the exceptions being:

- A. If the CGN tracks TCP sessions (e.g., with a state machine, as in [RFC6146] section 3.5.2.2), TCP ports MAY be reused immediately.
- B. If external ports are statically assigned to internal addresses (e.g., address X with port range 1000-1999 is assigned to subscriber A, 2000-2999 to subscriber B, etc.), and the assignment remains constant across state loss, then ports MAY be reused immediately.
- C. If the allocated external ports used address-dependent or address-and-port-dependent filtering before state loss, they MAY be reused immediately.

The length of time and the maximum number of ports in this state MUST be configurable by the CGN administrator.

Justification: This is necessary in order to prevent collisions between old and new mappings and sessions. It ensures that all established sessions are broken instead of redirected to a different peer.

The exceptions are for cases where reusing a port immediately does not create a possibility that packets would be redirected to the wrong peer. One can imagine other exceptions where mapping collisions are avoided, thus justifying the SHOULD level for this requirement.

The 120 seconds value corresponds to the Maximum Segment Lifetime (MSL) from [RFC0793].

Note that this requirement also applies to the case when a CGN loses state (due to a crash, reboot, failover to a cold standby, etc.). In that case, ports that were in use at the time of state loss SHOULD NOT be reallocated until at least 120 seconds have passed.

REQ-9: A CGN MUST implement a protocol giving subscribers explicit control over NAT mappings. That protocol SHOULD be the Port Control Protocol [I-D.ietf-pcp-base].

Justification: Allowing subscribers to manipulate the NAT state table with PCP greatly increases the likelihood that applications will function properly.

A study of PCP-less CGN impacts can be found in [I-D.donley-nat444-impacts]. Another study considering the effects of PCP on a peer-to-peer file sharing protocol can be found in [I-D.boucadair-pcp-bittorrent].

REQ-10: CGN implementers SHOULD make their equipment manageable. Standards-based management using standards such as "Definitions of Managed Objects for NAT" [RFC4008] is RECOMMENDED.

Justification: It is anticipated that CGNs will be primarily deployed in ISP networks where the need for management is critical. This requirement is at the SHOULD level to account for the fact that some CGN operators may not need management functionality.

Note also that there are efforts within the IETF toward creating a MIB tailored for CGNs (e.g., [I-D.ietf-behave-nat-mib]).

- REQ-11: When a CGN is unable to create a dynamic mapping due to resource constraints or administrative restrictions (i.e., quotas):
- A. it MUST drop the original packet;
 - B. it SHOULD send an ICMP Destination Unreachable message with code 1 (Host Unreachable) to the sender;
 - C. it SHOULD send a notification (e.g., SNMP trap) towards a management system (if configured to do so);
 - D. and it MUST NOT delete existing mappings in order to "make room" for the new one. (This only applies to normal CGN behavior, not to manual operator intervention.)

Justification: This is a slightly different form of REQ-8 from [RFC5508]. Code 1 is preferred to code 13 because it is listed as a "soft error" in [RFC1122], which is important because we don't want TCP stacks to abort the connection attempt in this case. See [RFC5461] for details on TCP's reaction to soft errors.

Sending ICMP errors and SNMP traps may be rate-limited for security reasons, which is why requirements B and C are SHOULDs, not a MUSTs.

Applications generally handle connection establishment failure better than established connection failure. This is why dropping the packet initiating the new connection is preferred over deleting existing mappings. See also the rationale in [RFC5508] section 6.

4. Logging

It may be necessary for CGN administrators to be able to identify a subscriber based on external IPv4 address, port, and timestamp in order to deal with abuse. When multiple subscribers share a single external address, the source address and port that are visible at the destination host have been translated from the ones originated by the subscriber.

In order to be able to do this, the CGN would need to log the following information for each mapping created (this list is for informational purposes only and does not constitute a requirement):

- o transport protocol
- o subscriber identifier (e.g., internal source address or tunnel endpoint identifier)
- o external source address
- o external source port
- o timestamp

By "subscriber identifier" we mean information that uniquely identifies a subscriber. For example, in a traditional NAT scenario, the internal source address would be sufficient. In the case of DS-Lite, many subscribers share the same internal address and the subscriber identifier is the tunnel endpoint identifier (i.e., the B4's IPv6 address).

A disadvantage of logging mappings is that CGNs under heavy usage may produce large amounts of logs, which may require large storage volume.

REQ-12: A CGN SHOULD NOT log destination addresses or ports unless required to do so for administrative reasons.

Justification: Destination logging at the CGN creates privacy issues. Furthermore, readers should be aware of logging recommendations for Internet-facing servers [RFC6302]. With compliant servers, the destination address and port do not need to be logged by the CGN. This can help reduce the amount of logging.

This requirement is at the SHOULD level to account for the fact that there may be other reasons for logging destination addresses or ports. One such reason might be that the remote server is not following [RFC6302].

5. Port Allocation Scheme

A CGN's port allocation scheme is subject to three competing requirements:

REQ-13: A CGN's port allocation scheme SHOULD maximize port utilization.

Justification: External ports is one of the resources being shared by a CGN. Efficient management of that resource directly impacts the quality of a subscriber's Internet connection.

Some schemes are very efficient in their port utilization. In that sense, they have good scaling properties (nothing is wasted). Others will systematically waste ports.

REQ-14: A CGN's port allocation scheme SHOULD minimize log volume.

Justification: Huge log volumes can be problematic to CGN operators.

Some schemes create one log entry per mapping. Others allow multiple mappings to generate a single log entry, which sometimes can be expressed very compactly. With some schemes the logging frequency can approach that of DHCP servers.

REQ-15: A CGN's port allocation scheme SHOULD make it hard for attackers to guess port numbers.

Justification: Easily guessed port numbers put subscribers at risk of the attacks described in [RFC6056].

Some schemes provide very good security in that ports numbers are not easily guessed. Others provide poor security to subscribers

A CGN implementation's choice of port allocation scheme optimizes to satisfy one requirement at the expense of another. Therefore, these are soft requirements (SHOULD as opposed to MUST).

6. Deployment Considerations

Several issues are encountered when CGNs are used [RFC6269]. There is current work in the IETF toward alleviating some of these issues. For example, see [I-D.ietf-intarea-nat-reveal-analysis].

7. IANA Considerations

There are no IANA considerations.

8. Security Considerations

If a malicious subscriber can spoof another subscriber's CPE, it may cause a DoS to that subscriber by creating mappings up to the allowed limit. An ISP can prevent this with ingress filtering, as described

in [RFC2827].

This document recommends Endpoint-Independent Filtering (EIF) as the default filtering behavior for CGNs. EIF has security considerations which are discussed in [RFC4787].

NATs sometimes perform fragment reassembly. CGNs would do so at presumably high data rates. Therefore, the reader should be familiar with the potential security issues described in [RFC4963].

9. Acknowledgements

Thanks for the input and review by Alexey Melnikov, Arifumi Matsumoto, Barry Leiba, Benson Schliesser, Dai Kuwabara, Dan Wing, Dave Thaler, David Harrington, Francis Dupont, Jean-Francois Tremblay, Joe Touch, Lars Eggert, Kousuke Shishikura, Mohamed Boucadair, Martin Stiernerling, Meng Wei, Nejc Skoberne, Pete Resnick, Reinaldo Penno, Ron Bonica, Sam Hartman, Sean Turner, Senthil Sivakumar, Stephen Farrell, Stewart Bryant, Takanori Mizuguchi, Takeshi Tomochika, Tina Tsou, Tomohiro Fujisaki, Tomohiro Nishitani, Tomoya Yoshida, Wes George, Wesley Eddy, and Yasuhiro Shirasaki.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4008] Rohit, R., Srisuresh, P., Raghunarayan, R., Pai, N., and C. Wang, "Definitions of Managed Objects for Network Address Translators (NAT)", RFC 4008, March 2005.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, April 2009.
- [RFC5597] Denis-Courmont, R., "Network Address Translation (NAT)

Behavioral Requirements for the Datagram Congestion Control Protocol", BCP 150, RFC 5597, September 2009.

[I-D.ietf-pcp-base]

Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", draft-ietf-pcp-base-26 (work in progress), June 2012.

10.2. Informative Reference

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5461] Gont, F., "TCP's Reaction to Soft Errors", RFC 5461, February 2009.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.
- [RFC6264] Jiang, S., Guo, D., and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", RFC 6264, June 2011.
- [RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.
- [RFC6302] Durand, A., Gashinsky, I., Lee, D., and S. Sheppard, "Logging Recommendations for Internet-Facing Servers",

BCP 162, RFC 6302, June 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[I-D.ietf-behave-nat-mib]
Perreault, S., Tsou, T., and S. Sivakumar, "Additional Managed Objects for Network Address Translators (NAT)", draft-ietf-behave-nat-mib-01 (work in progress), June 2012.

[I-D.ietf-intarea-nat-reveal-analysis]
Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier (HOST_ID) in Shared Address Deployments", draft-ietf-intarea-nat-reveal-analysis-02 (work in progress), April 2012.

[I-D.donley-nat444-impacts]
Donley, C., Howard, L., Kuarsingh, V., Berg, J., and U. Colorado, "Assessing the Impact of Carrier-Grade NAT on Network Applications", draft-donley-nat444-impacts-04 (work in progress), May 2012.

[I-D.boucadair-pcp-bittorrent]
Boucadair, M., Zheng, T., Deng, X., and J. Queiroz, "Behavior of BitTorrent service in PCP-enabled networks with Address Sharing", draft-boucadair-pcp-bittorrent-00 (work in progress), May 2012.

Authors' Addresses

Simon Perreault (editor)
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

Phone: +1 418 656 9254
Email: simon.perreault@viagenie.ca
URI: <http://www.viagenie.ca>

Ikuhei Yamagata
NTT Communications Corporation
Gran Park Tower 17F, 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Phone: +81 50 3812 4704
Email: ikuhei@nttv6.jp

Shin Miyakawa
NTT Communications Corporation
Gran Park Tower 17F, 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Phone: +81 50 3812 4695
Email: miyakawa@nttv6.jp

Akira Nakagawa
Japan Internet Exchange Co., Ltd. (JPiX)
Otemachi Building 21F, 1-8-1 Otemachi, Chiyoda-ku
Tokyo 100-0004
Japan

Phone: +81 90 9242 2717
Email: a-nakagawa@jpix.ad.jp

Hiroyuki Ashida
IS Consulting G.K.
12-17 Odenma-cho Nihonbashi Chuo-ku
Tokyo 103-0011
Japan

Email: assie@hir.jp

Behave WG
Internet-Draft
Obsoletes: 3338, 2767
(if approved)
Intended status: Standards Track
Expires: July 19, 2012

B. Huang
H. Deng
China Mobile
T. Savolainen
Nokia
January 16, 2012

Dual Stack Hosts Using "Bump-in-the-Host" (BIH)
draft-ietf-behave-v4v6-bih-09

Abstract

Bump-In-the-Host (BIH) is a host-based IPv4 to IPv6 protocol translation mechanism that allows a class of IPv4-only applications that work through NATs to communicate with IPv6-only peers. The host on which applications are running may be connected to IPv6-only or dual-stack access networks. BIH hides IPv6 and makes the IPv4-only applications think they are talking with IPv4 peers by local synthesis of IPv4 addresses. This document obsoletes RFC 2767 and RFC 3338.

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 July 19, 2012.

Copyright Notice

Copyright (c) 2012 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
1.1. Terminology	5
1.2. Acknowledgement of previous work	5
2. Components of the Bump-in-the-Host	6
2.1. Function Mapper	8
2.2. Protocol translator	8
2.3. Extension Name Resolver	8
2.3.1. Special exclusion sets for A and AAAA records	9
2.3.2. DNSSEC support	10
2.3.3. Reverse DNS lookup	10
2.3.4. DNS caches and synthetic IPv4 addresses	10
2.4. Address Mapper	11
3. Behavior and Network Examples	12
4. Considerations	16
4.1. Socket API Conversion	16
4.2. Socket bindings	16
4.3. ICMP Message Handling	16
4.4. IPv4 Address Pool and Mapping Table	16
4.5. Multi-interface	17
4.6. Multicast	18
5. Application-Level Gateway requirements considerations	19
6. IANA Considerations	20
7. Security Considerations	21
7.1. Implications on End-to-End Security	21
7.2. Filtering	21
7.3. Attacks on BIH	21
7.4. DNS considerations	22
8. Changes since RFC2767 and RFC3338	23
9. Acknowledgments	24
10. References	25
10.1. Normative References	25
10.2. Informative References	25
Appendix A. API list intercepted by BIH	27
Authors' Addresses	29

1. Introduction

This document describes Bump-in-the-Host (BIH), a successor and combination of the Bump-in-the-Stack (BIS) [RFC2767] and Bump-in-the-API (BIA) [RFC3338] technologies, which enable IPv4-only legacy applications to communicate with IPv6-only servers by synthesizing IPv4 addresses from AAAA records. Section 8 describes the reasons for making RFC2767 and RFC3338 obsolete.

The supported class of applications includes those that use DNS for IP address resolution and that do not embed IP address literals in application-protocol payloads. This includes legacy client-server applications using the DNS that are agnostic to the IP address family used by the destination and that are able to do NAT traversal. The synthetic IPv4 addresses shown to applications are taken from the RFC1918 private address pool in order to ensure that possible NAT traversal techniques will be initiated.

IETF recommends using dual-stack or tunneling based solutions for IPv6 transition and specifically recommends against deployments utilizing double protocol translation. Use of BIH together with a NAT64 is NOT RECOMMENDED [RFC6180].

BIH includes two major implementation alternatives: a protocol translator between the IPv4 and the IPv6 stacks of a host, or an API translator between the IPv4 socket API module and the TCP/IP module. Essentially, IPv4 is translated into IPv6 at the socket API layer or at the IP layer, former of which is the recommended implementation alternative.

When BIH is implemented at the socket API layer, the translator intercepts IPv4 socket API function calls and invokes corresponding IPv6 socket API function calls to communicate with IPv6 hosts.

When BIH is implemented at the network layer the IPv4 packets are intercepted and converted to IPv6 using the IP conversion mechanism defined in Stateless IP/ICMP Translation Algorithm (SIIT) [RFC6145]. The protocol translation has the same benefits and drawbacks as SIIT.

The location of the BIH refers to the location of the protocol translation function. The location of the IPv4 address and DNS A record synthesis function is orthogonal to the location of the protocol translation, and may or may not happen at the same location.

BIH can be used whenever an IPv4-only application needs to communicate with an IPv6-only server, independently of the address families supported by the access network. Hence the access network can be IPv6-only or dual-stack capable.

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 [RFC2119] .

This document uses terms defined in [RFC2460] and [RFC4213].

1.1. Terminology

DNS synthesis

DNS, A record, synthesis is a process where A type of DNS record is created by Extension Name Resolver to contain synthetic IPv4 address.

Real IPv4 address

An IPv4 address of a remote node a host has learned, for example, from DNS response to an A query.

Real IPv6 address

An IPv6 address of a remote node a host has learned, for example, from DNS response to an AAAA query.

Synthetic IPv4 address

An IPv4 address that has meaning only inside a host and that is used to provide IPv4 representation of remote node's real IPv6 address.

1.2. Acknowledgement of previous work

This document is a direct derivative from Kazuaki TSHUCHIYA, Hidemitsu HIGUCHI, and Yoshifumi ATARASHI's Bump-in-the-Stack [RFC2767] and from Seungyun Lee, Myung-Ki Shin, Yong-Jin Kim, Alain Durand, and Erik Nordmark's Bump-in-the-API [RFC3338], which similarly provides IPv4-only applications on dual-stack hosts the means to operate over IPv6. Section 8 covers the changes since those documents.

2. Components of the Bump-in-the-Host

Figure 1 shows the architecture of a host in which BIH is implemented as a socket API layer translator, i.e., as a "Bump-in-the-API".

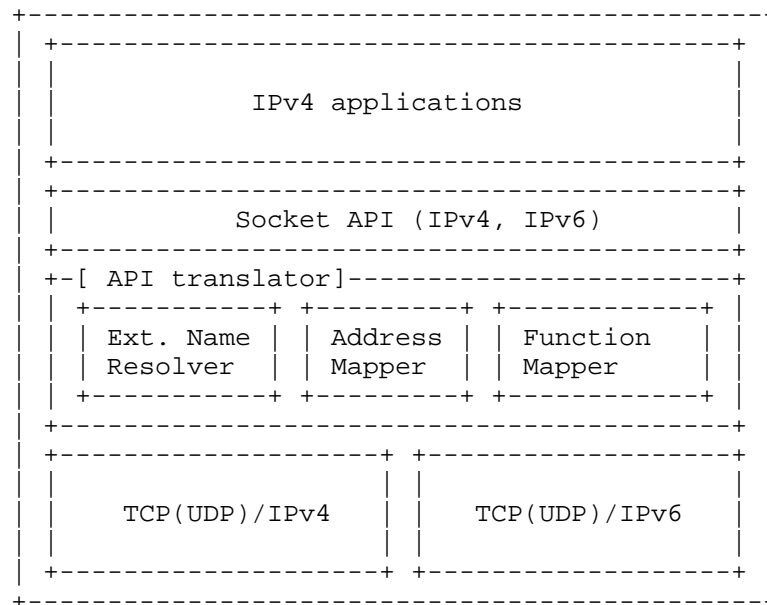


Figure 1: Architecture of a dual stack host using protocol translation at socket layer

Figure 2 shows the architecture of a host in which BIH is implemented as a network layer translator, i.e., a "Bump-in-the-Stack".

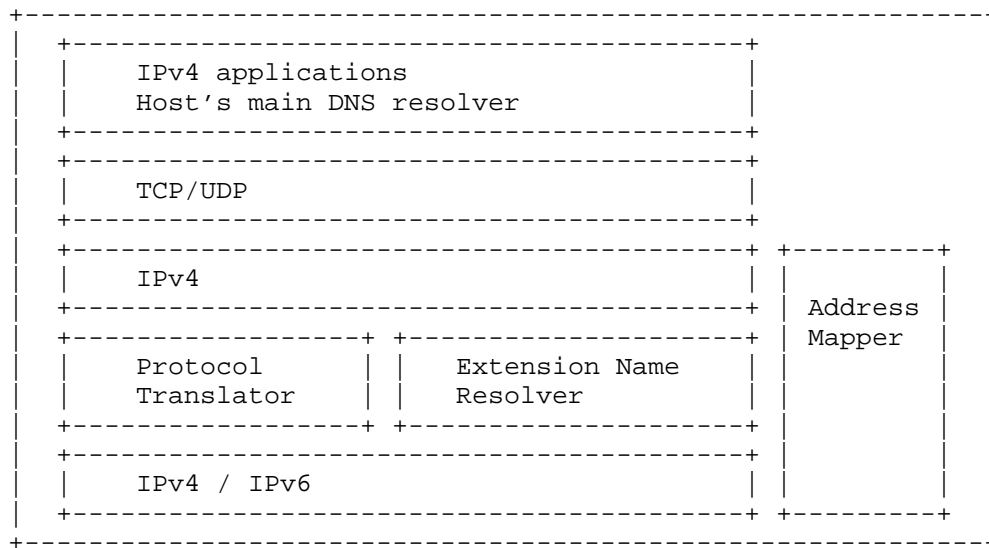


Figure 2: Architecture of a dual-stack host using protocol translation at the network layer

Dual stack hosts defined in RFC 4213 [RFC4213] need applications, TCP/IP modules and addresses for both IPv4 and IPv6. The proposed hosts in this document have an API or network-layer translator to allow legacy IPv4 applications to communicate with IPv6-only peers. The BIH architecture consists of an Extension Name Resolver, an Address Mapper, and depending on implementation either a Function Mapper or a Protocol Translator. It is worth noting that the Extension Name Resolver's placement is orthogonal to the placement of protocol translation. For example, the Extension Name Resolver may reside in the socket API while protocol translation takes place at the network layer.

The choice between the socket API and the network layer architectures varies case by case. While the socket API architecture alternative is the recommended one, it may not always be possible to choose. This may be the case, for example, when the used operating system does not allow modifications to be done for API implementations, but does allow addition of virtual network interfaces and related software modules. On the other hand, sometimes it may not be possible to introduce protocol translators inside the operating system, but it may be easy to modify implementations behind the API provided for applications. The choice of architecture also depends on who is creating implementation of BIH. For example, an application framework provider, an operating system provider, and a device vendor may all choose different approaches due their different

positions.

2.1. Function Mapper

The function mapper translates an IPv4 socket API function into an IPv6 socket API function.

When detecting IPv4 socket API function calls from IPv4 applications, the function mapper **MUST** intercept the function calls and invoke IPv6 socket API functions that correspond to the IPv4 socket API functions.

The function mapper **MUST NOT** perform function mapping when the application is initiating communications to the address range used by local synthesis and the address mapping table does not have an entry matching the address.

See Appendix A for an informational list of functions that would be appropriate to intercept by the function mapper.

2.2. Protocol translator

The protocol translator translates IPv4 into IPv6 and vice versa using the IP conversion mechanism defined in SIIT [RFC6145]. To avoid unnecessary fragmentation, the host's IPv4 module **SHOULD** be configured with a small enough MTU (MTU of the IPv6 enabled link - 20 bytes).

Protocol translation cannot be performed for IPv4 packets sent to the IPv4 address range used by local synthesis and for which a mapping table entry does not exist. The implementation **SHOULD** attempt to route such packets via IPv4 interfaces instead.

2.3. Extension Name Resolver

The Extension Name Resolver (ENR) returns an answer in response to the IPv4 application's name resolution request.

In the case of the socket API layer implementation alternative, when an IPv4 application tries to do a forward lookup to resolve names via the resolver library (e.g., `gethostbyname()`), BIH intercepts the function call and instead calls the IPv6 equivalent functions (e.g., `getaddrinfo()`) that will resolve both A and AAAA records. This implementation alternative is name resolution protocol agnostic, and hence supports techniques such as "hosts-file", NetBIOS, mDNS, and anything else the underlying operating system uses.

In the case of the network layer implementation alternative, the ENR

intercepts the A query and creates an additional AAAA query with similar content. The ENR will then collect replies to both A and AAAA queries and, depending on results, either return an A reply unmodified or synthesize a new A reply. If no reply for A query is received after ENR implementation specific timeout, after reception of positive AAAA response, the ENR MAY choose to proceed as if there were only AAAA record available for the destination.

The network layer implementation alternative will only be able to catch applications' name resolution requests that result in actual DNS queries, hence is more limited when compared to the socket API layer implementation alternative. Hence the socket API layer alternative is RECOMMENDED.

In either implementation alternative, if DNS A record reply contains non-excluded real IPv4 addresses the ENR MUST NOT synthesize IPv4 addresses.

The ENR asks the address mapper to assign a synthetic IPv4 address corresponding to each received IPv6 address if the A record query resulted in negative response, all received real IPv4 addresses were excluded, or the A query timed out. The timeout value is implementation specific and may be short in order to provide good user experience.

In the case of the API layer implementation alternative, the ENR will simply make the API (e.g. `gethostbyname`) return the synthetic IPv4 address. In the case of the network-layer implementation alternative, the ENR synthesizes an A record for the assigned synthetic IPv4 address, and delivers it up the stack. If the response contains a CNAME or a DNAME record, then the CNAME or DNAME chain is followed until the first terminating A or AAAA record is reached.

Application query	Network response	ENR behavior
-----+-----+-----		
IPv4 address(es)	IPv4 address(es)	return real IPv4 address(es)
IPv4 address(es)	IPv6 address(es)	synthesize IPv4 address(es)
IPv4 address(es)	IPv4/IPv6 address(es)	return real IPv4 address(es)

Figure 3: ENR behavior illustration

2.3.1. Special exclusion sets for A and AAAA records

An ENR implementation SHOULD by default exclude certain real IPv4 and IPv6 addresses seen on received A and AAAA records. The addresses to be excluded by default MAY include addresses such as those that

should not appear in the DNS or on the wire (see [RFC6147] section 5.1.4 and [RFC5735]). Additional addresses MAY be excluded based on possibly configurable local policies.

2.3.2. DNSSEC support

When the ENR is implemented at the network layer, the A record synthesis can cause similar issues as are described in [RFC6147] section 3. While running BIH, the main resolver of the host SHOULD NOT perform validation of A records as synthetic A records created by ENR would fail in validation. While not running BIH, host's resolver can use DNSSEC in the same way that any other resolver can. The ENR MAY support DNSSEC, in which case the (stub) resolver on a host can be configured to trust validations done by the ENR located at the network layer. In some cases the host's validating stub resolver can implement the ENR by itself.

When the ENR is implemented at the socket API level, there are no issues with DNSSEC use, as the ENR itself uses socket APIs for DNS resolution. This approach is RECOMMENDED.

2.3.3. Reverse DNS lookup

When an application requests a reverse lookup (PTR query) for an IPv4 address, the ENR MUST check whether the queried IPv4 address can be found in the Address Mapper's mapping table and is a synthetic IPv4 address. If an entry is found and the queried IPv4 address is synthetic, the ENR MUST initiate a corresponding reverse lookup for the real IPv6 address. In the case where the application requested a reverse lookup for an address not part of the synthetic IPv4 address pool, e.g., a global address, the request MUST be passed on unmodified.

For example, when an application requests a reverse lookup for a synthetic IPv4 address, the ENR needs to intercept that query. The ENR asks the address mapper for the real IPv6 address that corresponds to the synthetic IPv4 address. The ENR shall perform a reverse lookup procedure for the destination's IPv6 address and return the name received as a response to the application that initiated the IPv4 query.

2.3.4. DNS caches and synthetic IPv4 addresses

When BIH shuts down or address mapping table entries are cleared for any reason, DNS cache entries for synthetic IPv4 addresses MUST be flushed. There may be a DNS cache in the network-layer ENR itself, but also at the host's stub resolver.

2.4. Address Mapper

The address mapper maintains an IPv4 address pool that can be used for IPv4 address synthesis. The pool consists of [RFC1918] IPv4 addresses as per section 4.4. Also, the address mapper maintains a table consisting of pairs of synthetic IPv4 addresses and destinations' real IPv6 addresses.

When the extension name resolver, translator, or the function mapper requests the address mapper to assign a synthetic IPv4 address corresponding to an IPv6 address, the address mapper selects and returns an IPv4 address out of the local pool, and registers a new entry into the table. The registration occurs in the following three cases:

(1) When the extension name resolver gets only IPv6 addresses for the target host name and there is no existing mapping entry for the IPv6 addresses. One or more synthetic IPv4 addresses will be returned to the application and mappings for synthetic IPv4 addresses to real IPv6 addresses are created.

(2) When the extension name resolver gets both real IPv4 and IPv6 addresses, but the real IPv4 addresses contain only excluded IPv4 addresses (e.g., 127.0.0.1). The behavior will follow case (1).

(3) When the function mapper is triggered by a received IPv6 packet and there is no existing mapping entry for the IPv6 source address (for example, the client sent a UDP request to an anycast address but a response was received from a unicast address).

Other possible combinations are outside of BIH and BIH is not involved in those.

3. Behavior and Network Examples

Figure 4 illustrates a very basic network scenario. An IPv4-only application is running on a host attached to the IPv6-only Internet and is talking to an IPv6-only server. Communication is made possible by Bump-In-the-Host.

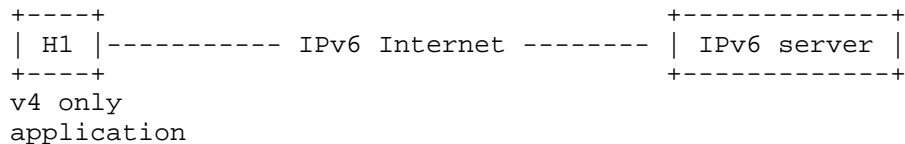


Figure 4: Network Scenario #1

Figure 5 illustrates a possible network scenario where an IPv4-only application is running on a host attached to a dual-stack network, but the destination server is running on a private site that is numbered with public IPv6 addresses and not globally reachable IPv4 addresses, such as [RFC1918] addresses, without port forwarding set up on the NAT44. The only means to contact the server is to use IPv6.

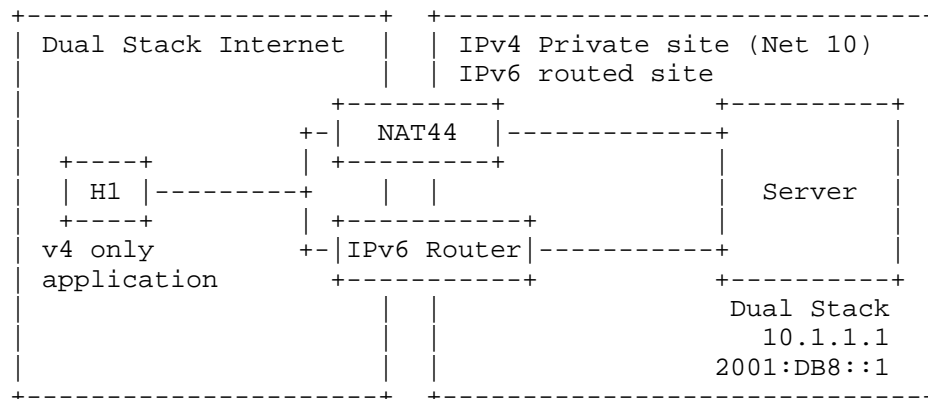
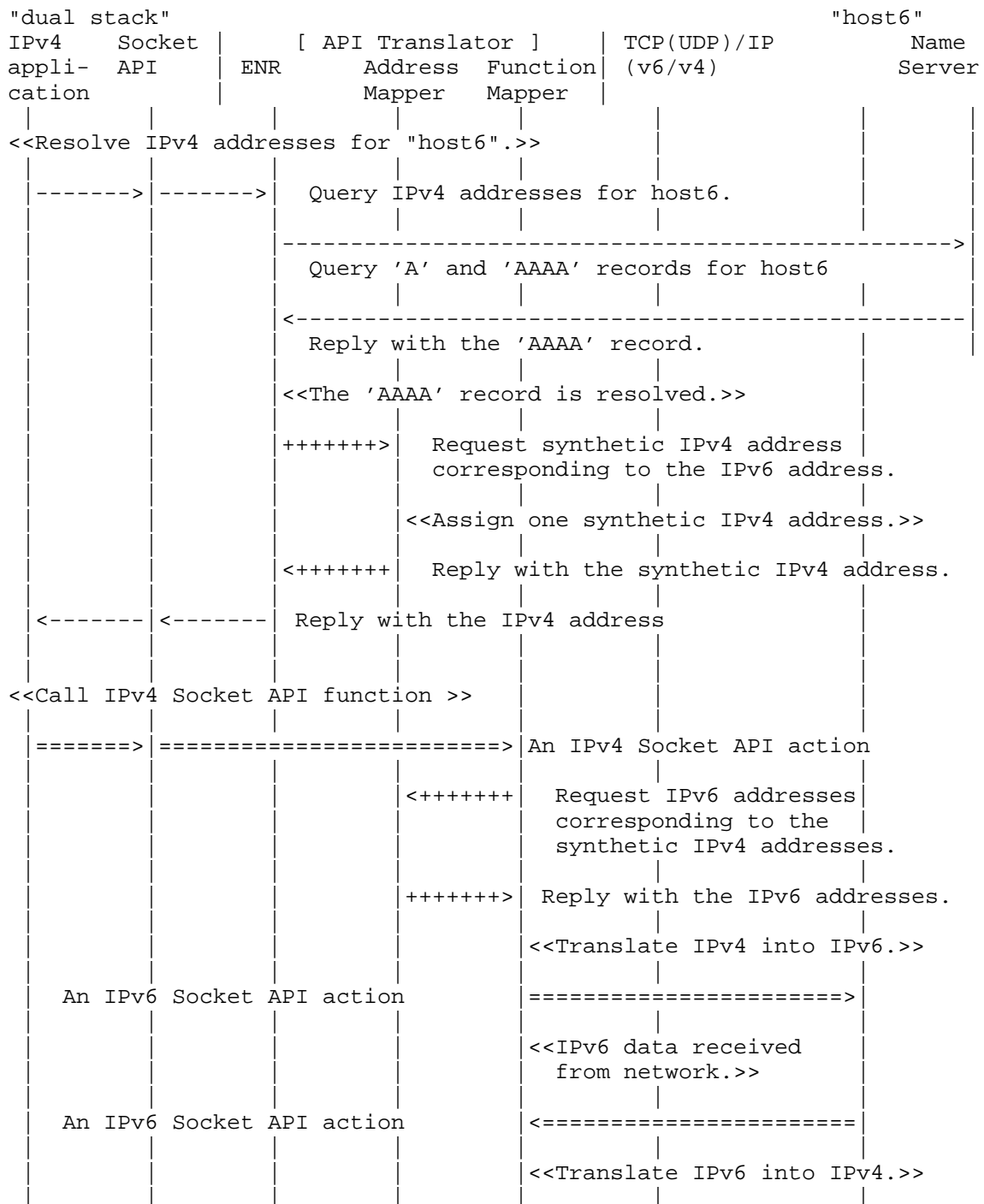


Figure 5: Network Scenario #2

Illustrations of host behavior in both implementation alternatives are given here. Figure 6 illustrates a setup where BIH (including the ENR) is implemented at the sockets API layer, and Figure 7 illustrates a setup where BIH (including the ENR) is implemented at the network layer.



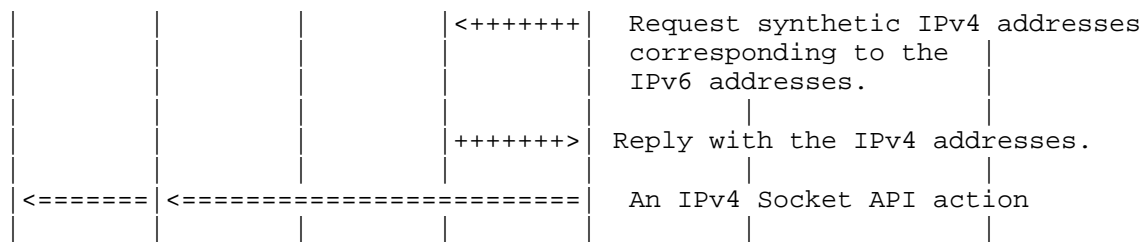
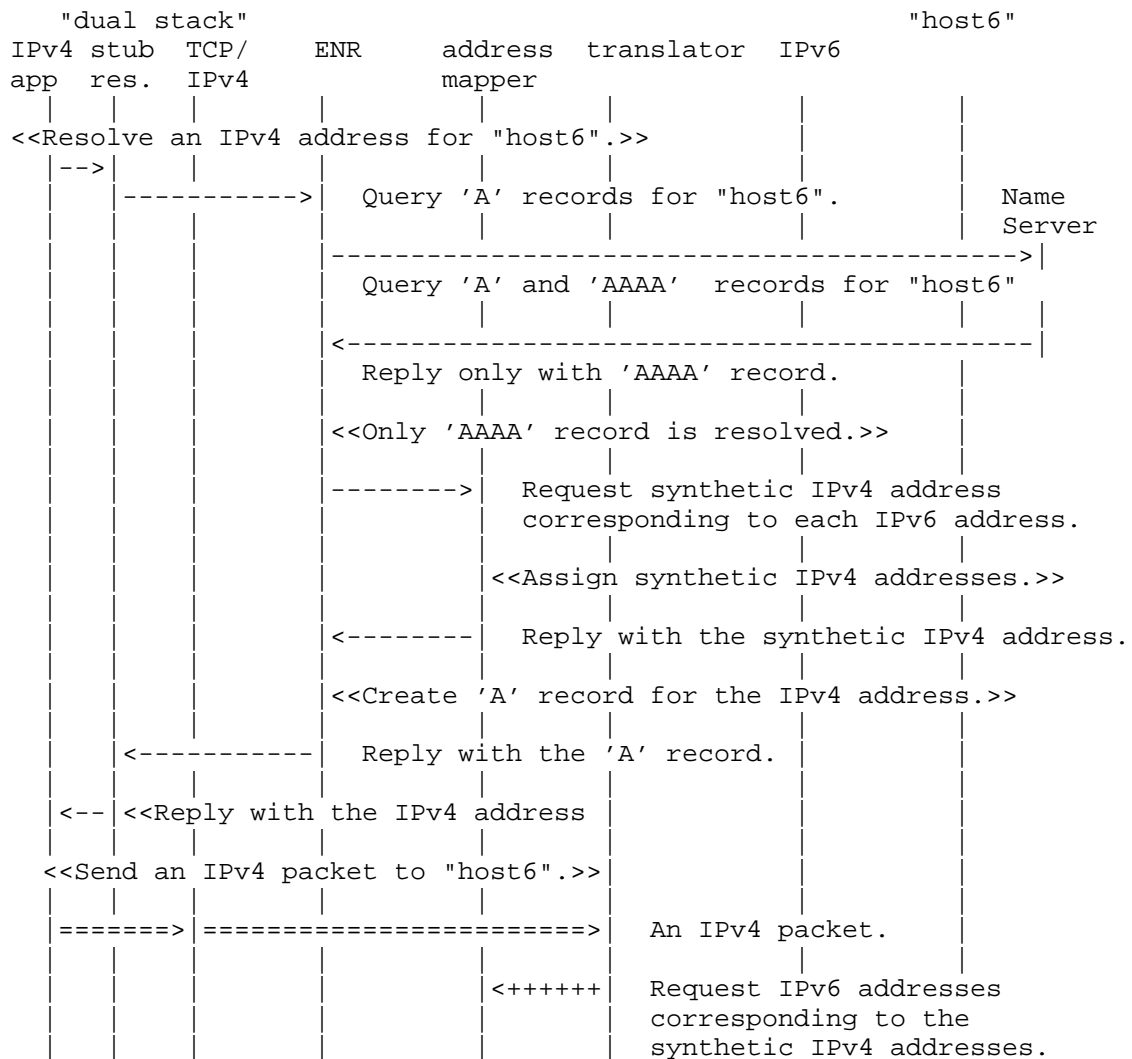


Figure 6: Example of BIH as API addition



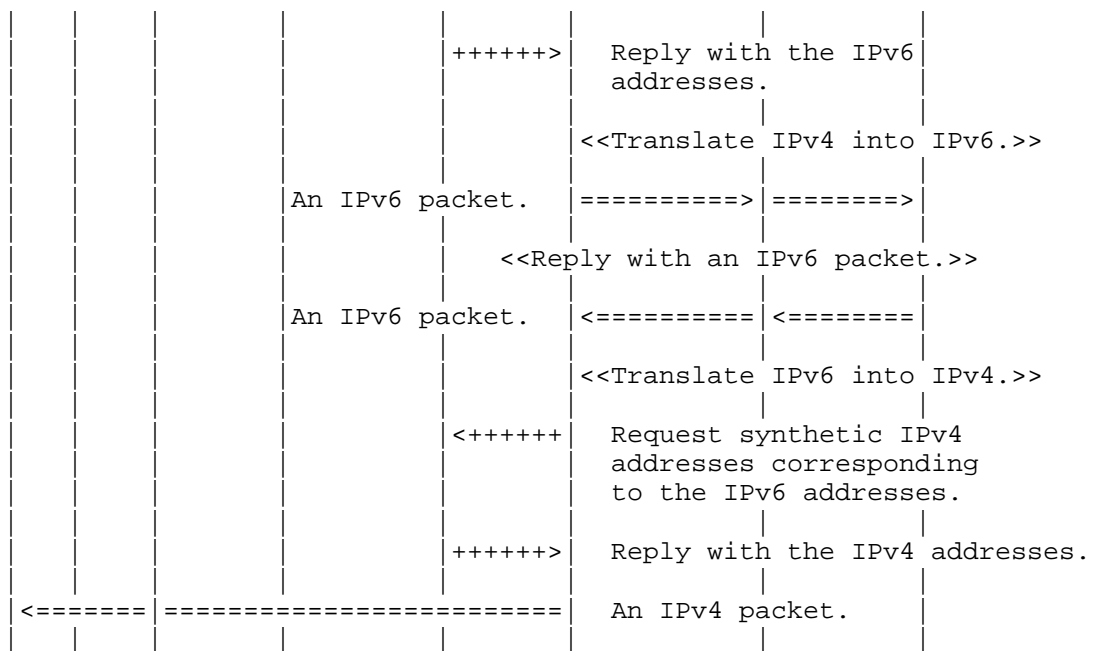


Figure 7: Example of BIH at the network layer

4. Considerations

4.1. Socket API Conversion

IPv4 socket API functions are translated into IPv6 socket API functions that are semantically as identical as possible and vice versa. See Appendix B for the API list intercepted by BIH. However, some IPv4 socket API functions are not fully compatible with IPv6 since IPv4 supports features that are not present in IPv6, such as `SO_BROADCAST`.

4.2. Socket bindings

BIH SHOULD select a source address for a socket from the recommended source address pool if a socket used for communications has not been explicitly bound to any IPv4 address.

The binding of an explicitly bound socket MUST NOT be changed by the BIH.

4.3. ICMP Message Handling

ICMPv4 and ICMPv6 messages MUST be translated as defined by SIIT [RFC6145]. In the network layer implementation alternative, protocol translator MUST translate ICMPv6 packets to ICMPv4 and vice versa, and in the socket API implementation alternative, the socket API MUST handle conversions in similar fashion.

4.4. IPv4 Address Pool and Mapping Table

The address pool consists of the [RFC1918] private IPv4 addresses. This pool can be implemented at different granularities in the node, e.g., a single pool per node, or at some finer granularity such as per-user or per-process. In the case of a large number of IPv4 applications communicating with a large number of IPv6 servers, the available address space may be exhausted if the granularity is not fine enough. This should be a rare event and chances will decrease as IPv6 support increases. The applications may use IPv4 addresses they learn for a much longer period than DNS time-to-live indicates. Therefore, the mapping table entries should be kept active for a long period of time. For example, a web browser may initiate one DNS query and then create multiple TCP sessions over time to the address it learns. When address mapping table clean-up is required, the BIH may utilize techniques used by network address translators, such as described in [RFC2663], [RFC5382], and [RFC5508].

The RFC1918 address space was chosen because generally legacy applications understand it as a private address space. A new

dedicated address space would run a risk of not being understood by applications as private. 127/8 and 169.254/16 are rejected due to possible assumptions applications may make when seeing those.

The RFC1918 addresses used by the BIH have a risk of conflicting with addresses used in the host's possible IPv4 interfaces and corresponding local networks. The conflicts can be mitigated, but not fully avoided, by using less commonly used portions of the RFC1918 address space. Addresses from 172.16/12 are thought to be less likely to be in conflict than addresses from 10/8 or 192.168/16 spaces. A source address can usually be selected in a non-conflicting manner, but a small possibility exists for synthesized destination addresses being in conflict with real addresses used in attached IPv4 networks.

The RECOMMENDED IPv4 addresses are following:

Primary source addresses: 172.21.112.0/20. Source addresses have to be allocated because applications use `getsockname()` calls and in the network layer mode an IP address of the IPv4 interface has to be shown (e.g., by `'ifconfig'`). More than one address is allocated to allow implementation flexibility, e.g., for cases where a host has multiple IPv6 interfaces. The source addresses are from different subnets than destination addresses to ensure applications would not make on-link assumptions and would instead enable NAT traversal functions.

Secondary source addresses: 10.170.224.0/20. These addresses are recommended if a host has a conflict with primary source addresses.

Primary destination addresses: 10.170.160.0/20. The address mapper will select destination addresses primarily out of this pool.

Secondary destination addresses: 172.21.80.0/20. The address mapper will select destination addresses out of this pool if the node has a dual-stack connection conflicting with primary destination addresses.

4.5. Multi-interface

In the case of dual-stack destinations BIH MUST NOT do protocol translation from IPv4 to IPv6 when the host has any IPv4 interfaces, native or tunneled, available for use.

It is possible that an IPv4 interface is activated during BIH operation, for example if a node moves to a coverage area of an IPv4-enabled network. In such an event, BIH MUST stop initiating protocol translation sessions for new connections and BIH MAY disconnect active sessions. The choice of disconnection is left for

implementations and it may depend on whether IPv4 address conflict occurs between addresses used by BIH and addresses used by the new IPv4 interface.

4.6. Multicast

Protocol translation for multicast is not supported.

5. Application-Level Gateway requirements considerations

No Application-Level Gateway (ALG) functionality is specified herein as ALG design is generally not encouraged for host-based translation and as BIH is intended for applications that do not include IP addresses in protocol payloads.

6. IANA Considerations

There are no actions for IANA.

7. Security Considerations

The security considerations of BIH follows closely, but not completely, those of NAT64 [RFC6146] and DNS64 [RFC6147]. The following sections are copied from RFC6146 and RFC6147 and modified for BIH scenario.

7.1. Implications on End-to-End Security

Any protocols that protect IP header information are essentially incompatible with BIH. This implies that end-to-end IPsec verification will fail when the Authentication Header (AH) is used (both transport and tunnel mode) and when ESP is used in transport mode. This is inherent in any network-layer translation mechanism. End-to-end IPsec protection can be restored, using UDP encapsulation as described in [RFC3948]. The actual extensions to support IPsec are out of the scope of this document.

7.2. Filtering

BIH creates binding state using packets flowing from the IPv4 side to the IPv6 side. In accordance with the procedures defined in this document following the guidelines defined in [RFC4787], a BIH implementation MUST offer "Endpoint-Independent Mapping".

Implementations MAY also provide support for "Address-Dependent Mapping" following the guidelines defined in [RFC4787].

The security properties, however, are determined by which packets the BIH allows in and which it does not. The security properties are determined by the filtering behavior and by the possible filtering configuration in the filtering portions of the BIH, not by the address mapping behavior.

7.3. Attacks on BIH

The BIH implementation itself is a potential victim of different types of attacks. In particular, the BIH can be a victim of DoS attacks. The BIH implementation has a limited number of resources that can be consumed by attackers creating a DoS attack. The BIH has a limited number of IPv4 addresses that it uses to create the bindings. Even though the BIH performs address translation, it is possible for an attacker to consume the synthetic IPv4 address pool by triggering a host to issue DNS queries for names that cause ENR to synthesise A records. DoS attacks can also affect other limited resources available in the host running BIH such as memory or link capacity. For instance, it is possible for an attacker to launch a DoS attack on the memory of the BIH running device by sending

fragments that the BIH will store for a given period. If the number of fragments is large enough, the memory of the host could be exhausted. BIH implementations **MUST** implement proper protection against such attacks, for instance, allocating a limited amount of memory for fragmented packet storage.

Another consideration related to BIH resource depletion refers to the preservation of binding state. Attackers may try to keep a binding state alive forever by sending periodic packets that refresh the state. In order to allow the BIH to defend against such attacks, the BIH implementation **MAY** choose not to extend the session entry lifetime for a specific entry upon the reception of packets for that entry through the external interface. However, such an action would not allow one-way communication sessions to stay alive.

7.4. DNS considerations

BIH operates in combination with the DNS, and is therefore subject to whatever security considerations are appropriate to the DNS mode in which the BIH is operating (i.e. recursive or stub-resolver mode).

BIH has the potential to interfere with the functioning of DNSSEC, because BIH modifies DNS answers, and DNSSEC is designed to detect such modifications and to treat modified answers as bogus.

8. Changes since RFC2767 and RFC3338

This document combines and obsoletes both [RFC2767] and [RFC3338].

The changes in this document mainly reflect the following:

1. RFC1918 addresses used used for synthesis

The RFC3338 used unassigned IPv4 addresses (e.g., 0.0.0.1 - 0.0.0.255) for synthetic IPv4 addresses. Those addresses should not have been used and that may cause problems with applications. It is preferable to use RFC1918 defined addresses instead, as described in Section 4.4.

2. Support for reverse (PTR) DNS queries

Neither RFC2767 or RFC3338 included support for reverse (PTR) DNS queries. This document adds the support at Section 2.3.3.

3. DNSSEC support

RFC2767 did not include DNSSEC considerations, which are now included in Section 2.3.2

4. Architectural recommendation

This document recommends socket API layer implementation option over network layer translation, i.e. recommends approach introduced in RFC2767 over the approach of RFC3338.

5. Standards track document

RFC2767 is classified as Informational RFC and RFC3338 as Experimental RFC. It was discussed and decided in the IETF that this technology should be on the standards track.

6. Set of other extensions and improvements

Set of lesser extensions, improvements, and clarifications have been introduced. These include but are not limited to: IPv4 and IPv6 address exclusion sets at Section 2.3.1, host's DNS cache considerations, ENR behaviour updates, updated security considerations, example updates, and deployment scenario updates.

9. Acknowledgments

The authors thank the discussion from Gang Chen, Dapeng Liu, Bo Zhou, Hong Liu, Tao Sun, Zhen Cao, Feng Cao et al. in the development of this document.

The efforts of Mohamed Boucadair, Dean Cheng, Lorenzo Colitti, Paco Cortes, Ralph Droms, Stephen Farrell, Fernando Gont, Marnix Goossens, Wassim Haddad, Ala Hamarsheh, Dave Harrington, Ed Jankiewicz, Suresh Krishnan, Julien Laganier, Yiu L. Lee, Jan M. Melen, Qibo Niu, Pierrick Seite, Christian Vogt, Magnus Westerlund, Dan Wing, and James Woodyatt in reviewing this document are gratefully acknowledged.

Special acknowledgements go to Dave Thaler for his extensive review and support.

The authors of RFC2767 acknowledged WIDE Project, Kazuhiko YAMAMOTO, Jun MURAI, Munechika SUMIKAWA, Ken WATANABE, and Takahisa MIYAMOTO. The authors of RFC3338 acknowledged implementation contributions by Wanjik Lee (wjlee@arang.miryang.ac.kr) and i2soft Corporation (www.i2soft.net).

The authors of Bump-in-the-Wire (BIW) (draft-ietf-biw-00.txt, October 2006), P. Moster, L. Chin, and D. Green, are acknowledged. Some ideas and clarifications from BIW have been adapted to this document.

10. References

10.1. Normative References

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

10.2. Informative References

- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.
- [RFC2767] Tsuchiya, K., HIGUCHI, H., and Y. Atarashi, "Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)", RFC 2767, February 2000.
- [RFC3338] Lee, S., Shin, M-K., Kim, Y-J., Nordmark, E., and A. Durand, "Dual Stack Hosts Using "Bump-in-the-API" (BIA)", RFC 3338, October 2002.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W.

- Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.
- [RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, April 2009.
- [RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", BCP 153, RFC 5735, January 2010.
- [RFC6180] Arkko, J. and F. Baker, "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment", RFC 6180, May 2011.

Appendix A. API list intercepted by BIH

The following informational list includes some of the API functions that would be appropriate to intercept by BIH module when implemented at the socket API layer. Please note that this list is not fully exhaustive, as the function names and services that are available on different APIs vary significantly.

The functions that the application uses to pass addresses into the system are:

```
bind()  
  
connect()  
  
sendmsg()  
  
sendto()  
  
gethostbyaddr()  
  
getnameinfo()
```

The functions that return an address from the system to an application are:

```
accept()  
  
recvfrom()  
  
recvmsg()  
  
getpeername()  
  
getsockname()  
  
gethostbyname()  
  
getaddrinfo()
```

The functions that are related to socket options are:

```
getsockopt()  
  
setsockopt()
```

As well, raw sockets for IPv4 and IPv6 may be intercepted.

Most of the socket functions require a pointer to the socket address structure as an argument. Each IPv4 argument is mapped into corresponding an IPv6 argument, and vice versa.

According to [RFC3493], the following new IPv6 basic APIs and structures are required.

IPv4	new IPv6

AF_INET	AF_INET6
sockaddr_in	sockaddr_in6
gethostbyname()	getaddrinfo()
gethostbyaddr()	getnameinfo()
inet_ntoa()/inet_addr()	inet_pton()/inet_ntop()
INADDR_ANY	in6addr_any

Figure 8

BIH may intercept `inet_ntoa()` and `inet_addr()` and use the address mapper for those. Doing that enables BIH to support literal IP addresses. However, IPv4 address literals can only be used after a mapping entry between the IPv4 address and corresponding IPv6 address has been created.

The `gethostbyname()` and `getaddrinfo()` calls return a list of addresses. When the name resolver function invokes `getaddrinfo()` and `getaddrinfo()` returns multiple IP addresses, whether IPv4 or IPv6, they should all be represented in the addresses returned by `gethostbyname()`. Thus if `getaddrinfo()` returns multiple IPv6 addresses, this implies that multiple address mappings will be created; one for each IPv6 address.

Authors' Addresses

Bill Huang
China Mobile
53A,Xibianmennei Ave.,
Xuanwu District,
Beijing 100053
China

Email: bill.huang@chinamobile.com

Hui Deng
China Mobile
53A,Xibianmennei Ave.,
Xuanwu District,
Beijing 100053
China

Email: denghui02@gmail.com

Teemu Savolainen
Nokia
Hermiankatu 12 D
FI-33720 TAMPERE
Finland

Email: teemu.savolainen@nokia.com

Individual Submission
Internet Draft
Intended status: Informational
Expires: April 2011

E. Jankiewicz (Ed.)
SRI International, Inc.
October 25, 2010

An Annotated Bibliography for IPv4-IPv6 Transition and Coexistence
draft-jankiewicz-v6ops-v4v6biblio-03.txt

Abstract

The Internet is in the early stages of what may be a protracted period of coexistence of IPv4 and IPv6. Network operators are challenged with the task of activating IPv6 without negative impact on operating IPv4 networks and their customers. This draft is an informational "annotated bibliography" compiled to help in the analysis and development of basic guidelines and recommendations for network operators. The goal of this document is to survey the current state of RFCs, Internet-Drafts and external reference materials that define the use cases, problem statements, protocols, transition mechanisms and coexistence tools that will be of interest to a network operator planning to turn on IPv6.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 25, 2009.

Copyright Notice

Copyright (c) 2010 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.

Table of Contents

1. Introduction.....	3
1.1. The Three Laws of IPv4/IPv6 Coexistence Mechanisms.....	4
2. IPv6 and related Protocol Specifications.....	6
3. Problem Statements and Use Cases.....	7
4. Transition and Coexistence Scenarios and Architectures.....	8
5. Transition/Coexistence Tools	10
5.1. Address Mapping.....	11
5.1.1. Address Translation in Network Operations.....	11
5.1.2. Application and End-User Considerations With NAT..	13
5.1.3. Dual-Stack Lite (DS-lite).....	15
5.2. Tunneling Mechanisms.....	17
5.2.1. Teredo.....	17
5.2.2. IPv6 Rapid Deployment (6rd)and Extensions.....	18
5.2.3. Tunnel Support Protocol (TSP)	20
5.2.4. Residual IPv4 Deployment over IPv6-only Infrastructure	20
5.2.5. Address Plus Port (AplusP).....	20
5.2.6. IRON-RANGER and ISATAP Solutions.....	21
5.2.7. Softwires Hub and Spoke with L2TP.....	22
5.3. Translation.....	22
5.3.1. Historic Approach.....	22
5.3.2. Current Translation Approaches.....	23
5.3.2.1. An IPv6 network to the IPv4 Internet.....	25
5.3.2.2. The IPv4 Internet to an IPv6 network.....	25
5.3.2.3. The IPv6 Internet to an IPv4 network.....	25
5.3.2.4. An IPv4 network to the IPv6 Internet.....	26
5.3.2.5. An IPv6 network to an IPv4 network.....	26
5.3.2.6. An IPv4 network to an IPv6 network.....	26
5.3.2.7. The IPv6 Internet to the IPv4 Internet.....	26
5.3.2.8. The IPv4 Internet to the IPv6 Internet.....	26
5.4. Connectivity Checking and Delay Avoidance.....	27
6. Prefix and Address Assignment and Distribution.....	28
7. How-to, Whitepapers and FAQs	30

8. Experiments, Trials and Prototypes.....	30
9. Implementation Reports	31
10. Books on IPv6.....	31
11. Miscellaneous.....	32
12. Security Considerations.....	33
13. IANA Considerations.....	33
14. Conclusions.....	33
15. References	33
15.1. Normative References.....	33
15.2. Informative References	33
16. Acknowledgments.....	34

1. Introduction

Since the IPv6 protocol was defined in 1995 as RFC 1883 (replaced in 1998 by RFC 2460) the Internet has been in a long transition from IPv4 to IPv6. In reality, we are still in the early stages of what is likely to be a protracted period of coexistence, where IPv6 penetration in hosts (both servers and clients) will gradually ramp up as networks make IPv6 available through their infrastructures.

Network operators face a daunting task to design and implement plans to activate IPv6 without negative impact on large (in some cases very large) operating IPv4 networks with many live customers. Some basic guidelines and recommendations for network operators are being developed (<http://tools.ietf.org/html/draft-lee-v4v6tran-problem>) and this draft is an informational companion to that effort. The goal of this document is to survey the current state of RFCs, active (and expired but still relevant) Internet-Drafts and external reference materials that define the use cases, problem statements, protocols, transition mechanisms and coexistence tools that will be of interest to a network operator planning to turn on IPv6.

This is a dynamic and evolving marketplace of ideas. At best, this draft is a blurry snapshot of the landscape near to the time of its publication. The editor intends this compendium to be merely the starting point for an active database or wiki available for community contribution including feedback on the real-world experience of network operators as they turn on IPv6. Note that the links to RFCs and drafts are based on the IETF Tools view of the repository at <http://tools.ietf.org/html/>. The links for active drafts are not for a specific revision but should link to the last or latest version.

The following sections comprise an annotated bibliography of the currently available documentation to knowledge of the editor. It is provided as informational guidance only, and any network operator contemplating an IPv6 implementation will of course exercise due

diligence in researching all the issues, standards and recommendations and analyze applicability to the particular network operation.

Note that as the body of this text includes full reference information for the bibliography entries these are not included in the normal Reference section.

[Editor's note to be removed before publication:

While this draft is circulating, the editor is interested in any and all pointers to additional useful references. Contributions of capsule summaries and applicability for any of the listed entries would also be appreciated and will be graciously acknowledged. If I have missed anyone who already chipped in, this will be cheerfully rectified upon your reminder via a private e-mail.]

1.1. The Three Laws of IPv4/IPv6 Coexistence Mechanisms

The Editor of this draft thought it might be helpful to briefly explore the motivations driving the current profusion of coexistence mechanisms. In the not so distant past little or no discussion of this topic was going on in the IETF, as many felt the case was closed. A discussion in the Intarea meeting at IETF 71 in Dublin and a presentation at the plenary at that meeting led to a reawakening of interest in coexistence and transition tools. This discussion continued at a special meeting in Montreal in October 2008, and has occupied substantial time on the mailing lists and meetings of several Working Groups since then. The Internet Area, IPv6 Operation (v6ops), Softwires and Behave WGs have generated many contributions, and an ad-hoc discussion mailing list has been established at <https://www.ietf.org/mailman/listinfo/v4tov6transition>.

Early in the life of IPv6, the assumption was made that IPv6 deployment, based on dual-stack implementations, would be ubiquitous long before the IPv4 address pool would run out. For special cases, tunneling through dissimilar networks or use of an external translation box such as NAT-PT would allow interim operation of legacy equipment. At present, this has not yet come to pass. The impending exhaustion of IPv4 address space renders dual-stack impossible in some deployments and issues have resulted in NAT-PT being deprecated to Historic status.

Nature (and your average Internet-Draft author) abhors a vacuum. With the demise of NAT-PT and the increasing urgency to get moving on IPv6 transition, we are now in a period of "Let 1000 Flowers Bloom" where many ideas are being advanced, and a lot of IETF brainpower is

being spent debating the relative merits and evilness of various approaches. The spectrum of opinion on coexistence mechanisms has two extremes:

IPv4 is so Over: Concentrate on deploying native IPv6 and managing it effectively, rather than spinning more complex webs of IPv4 accommodation. Deploying anything that delays IPv6 and enables more IPv4 usage at this point is irresponsible.

Where's the Business Case: Real customers need IPv4, there is no IPv6 content, no demand for IPv6. Scale up NAT to keep IPv4 viable, provide some sort of artificial IPv6 access, if and when customers ask. No plans for native IPv6 in the foreseeable future.

A reasonable position recognizes the valid motivation on both sides. An ISP may not be able to dictate updates to customer computers and routers, and must provide access to all legacy customers, not just eager IPv6 adopters, so an interim mechanism that minimizes their inconvenience is needed. One size will never fit all, so some solutions may be a good fit for one ISP, and not for others. While evaluating all the alternative documented here, the principle to keep in mind is that the IETF should provide good engineering opinions on all these alternatives, to permit things that will help, and prevent things that will cause problems.

This can be summed up in the "Three Laws of IPv4/IPv6 Coexistence Mechanisms":

1. First, do no harm.
2. Keep it simple.
3. Keep moving towards more native IPv6.

"No harm" in this case means that a good solution will not unduly interfere with good experience for the legacy IPv4 customer, nor will it impede the eager IPv6 adopter. The solution must not cause problems for peer or backbone networks or for the Internet community at large.

"Simple" means to solve particular problems with specific solutions focused to the point of need rather than attempting broad and complex methods that impinge on all traffic. However, do not simplify any more than necessary to avoid harm.

The compulsion to move towards native IPv6 follows from the first two laws. Over time, even minimal harm and complexity that even a good

mechanism presents can and should be reduced over time by continuing to enable, promote and encourage transition to native IPv6. Design and deploy your interim solution(s) with a clear migration path that will eventually render them redundant. Set a date after which you will not deploy any new equipment that does not support IPv6. Set a date to sunset IPv4 access, giving legacy customers plenty of time (and incentive) to upgrade their old equipment.

In summary, it seems that the Robustness Principle (Postel's Law) would apply, as it does in many situations:

"Be conservative in what you do, be liberal in what you accept from others." [RFC 793]

Following the Robustness Principle and the Three Laws should allow an operator complete freedom to manage their own network and to choose and operate any coexistence mechanism as long as they need to for supporting their customers, except where those choices cause harm to someone else. Of course, there is no universal definition of "harm" so reasonable people can disagree, e.g. if a mechanism in use on the access side causes additional delay, content providers may see that as "harming" their users' experience. That's why Working Group mailing lists and IETF meetings are just so much fun.

Oh, and by the way, the Fourth Law should be "Don't reinvent the wheel" so please explore the RFCs, drafts and other citations to see if someone has already proposed something similar to your idea. Your contributions are needed, but time and energy is better spent exploring novel approaches and building on what has already been proposed.

2. IPv6 and related Protocol Specifications

"IPv6 Node Requirements" J. Loughney, Ed. April 2006
<http://tools.ietf.org/html/rfc4294>

"IPv6 Node Requirements RFC 4294-bis" E. Jankiewicz, J. Loughney, T. Narten
<http://tools.ietf.org/html/draft-ietf-6man-node-req-bis>

RFC 4294 and its update draft are included by reference. These provide a comprehensive overview of the IPv6 baseline specifications and the reader is directed to them to avoid a redundant listing here.

3. Problem Statements and Use Cases

"Problem Statements of IPv6 Transition of ISP" Y. Lee, Ed.
<http://tools.ietf.org/html/draft-lee-v4v6tran-problem>

This draft is being developed by an ad-hoc group interested in providing guidance to network operators on the IPv6 transition. It will include high level use cases (as contributed by IETF participants with network operator experience) and a problem statement documenting what additional work IETF could do to provide sufficient tools and guidance for the network operators

"Mobile Networks Considerations for IPv6 Deployment" R. Koodli
<http://tools.ietf.org/html/draft-ietf-v6ops-v6-in-mobile-networks>

Mobile Internet access from smartphones and other mobile devices is accelerating the exhaustion of IPv4 addresses. IPv6 is widely seen as crucial for the continued operation and growth of the Internet, and in particular, it is critical in mobile networks. This document discusses the issues that arise when deploying IPv6 in mobile networks. Hence, this document can be a useful reference for service providers and network designers.

"Routing Loop Attack using IPv6 Automatic Tunnels: Problem Statement and Proposed Mitigations", G. Nakibly and F. Templin
<http://tools.ietf.org/html/draft-ietf-v6ops-tunnel-loops>

This document is concerned with security vulnerabilities in IPv6-in-IPv4 automatic tunnels. These vulnerabilities allow an attacker to take advantage of inconsistencies between the IPv4 routing state and the IPv6 routing state. The attack forms a routing loop which can be abused as a vehicle for traffic amplification to facilitate DoS attacks. If automatic tunnels are used in a deployment the warnings and mitigations in this draft should be considered.

"Use Case for IPv6 Transition for a Large-Scale Broadband Network" CC. Huang (Ed.), XY. Li and LM. Hu
<http://tools.ietf.org/html/draft-huang-v6ops-v4v6tran-bb-usecase>

"IPv6 Transition Cable Access Network Use Cases" Y. Lee and V. Kuarsingh
<http://tools.ietf.org/html/draft-lee-v4v6tran-usecase-cable>

"IPv6 Transition Use Case for a Large Mobile Network" C. Zhou (Ed.) and T. Taylor
<http://tools.ietf.org/html/draft-zhou-v6ops-mobile-use-case-00>

Each of these use case drafts is focused on a particular deployment model for a specific market segment. While each may be based on a singular operator's experience or planning, the intention is to develop the set of use cases drafts to be of interest to any network operator in the segment.

"Considerations for Stateless Translation (IVI/dIVI) in Large SP Network" Q. Sun et al.

<http://tools.ietf.org/html/draft-sunq-v6ops-ivi-sp>

"dIVI" is a prefix-specific and stateless address mapping method based on IVI which can directly translate IPv4 packet to IPv6 packet. This document describes the challenges and requirements for large Service Provider to deploy IPv6 in an operational network and specifically considerations for dIVI deployment.

4. Transition and Coexistence Scenarios and Architectures

RFC 5211 "An Internet Transition Plan." J. Curran, July 2008

<http://tools.ietf.org/html/rfc5211>

While the abstract for this RFC humbly describes it as just "one possible plan" for the IPv6 transition, it provides very good context and a common language to use when talking about transition plans, and can be seen as a call to action. It describes three phases of the transition, and proposes a timeline based on predictions of the imminent exhaustion of the IPv4 address space. The phases are:

1. Preparation, where IPv4 predominates while service providers trial and experiment with IPv6, and end-users prepare to provide Internet-facing IPv6 services in the future. The timeline in the RFC described this phase as in progress, and optimally this phase would have ended already.
2. Transition, where both IPv4 and IPv6 services are offered and used, with production level support for IPv6, although this may be via transition mechanisms rather than native IPv6. The RFC targeted this phase to end in 2011.
3. Post-Transition, where native IPv6 services should be offered while IPv4 services may still be supported.

"Guidelines for Using Transition Mechanisms During IPv6 Deployment"
J. Arkko and F. Baker

<http://tools.ietf.org/html/draft-arkko-ipv6-transition-guidelines>

IPv6 deployment requires some effort, resources, and expertise. The availability of many different deployment models is one reason why expertise is required. This draft discusses the IPv6 deployment models and migration tools, and recommends ones that have been found to work well in operational networks in many common situations.

"IPv6 Transition Guide For A Large ISP Providing Broadband Access", G. Yang (Ed.), L. Hu and J. Lin
<http://tools.ietf.org/html/draft-yang-v6ops-v4v6tran-bb-transition-guide>

This draft is a product of the current v4tov6transition effort and it examines IPv6 migration solutions for each part of the Large-scale broadband infrastructure with a layer 2 access network. The analysis is based on the requirements for providing existing broadband services in v4v6-coexisting or IPv6-only situations. The draft describes the suitable scenarios for each solution.

"IPv6 Transition Guide for a Large Mobile Operator" T. Tsou (Ed.) and T. Taylor
<http://tools.ietf.org/html/draft-tsou-v6ops-mobile-transition-guide>

Similarly, this draft examines IPv6 migration solutions for a large mobile network.

RFC 6036 "Emerging Service Provider Scenarios for IPv6 Deployment", B. Carpenter, S. Jiang
<http://www.rfc-editor.org/rfc/rfc6036.txt>

This document describes practices and plans that are emerging among Internet Service Providers for the deployment of IPv6 services. They are based on practical experience so far, as well as current plans and requirements, reported in a survey of a number of ISPs carried out in early 2010. The document identifies a number of technology gaps, but does not make recommendations.

"Framework for IP Version Transition Scenarios", B. Carpenter, S. Jiang and V. Kuarasingh
<http://tools.ietf.org/html/draft-carpenter-v4v6tran-framework>

This document sets out a framework for the presentation of scenarios and recommendations for a variety of approaches to the transition from IPv4 to IPv6, given the necessity for a long period of co-existence of the two protocols.

5. Transition/Coexistence Tools

As network operators and end-users independently proceed with transition to IPv6 while others continue to use IPv4, a potentially long period of coexistence will ensue. Variations on terminology have been used since the specification of IPv6; transition implies a process whereby the star of IPv6 rises and the star of IPv4 sets; coexistence implies that both will operate together. Due to thoroughly discussed limits to the growth of an Internet using only IPv4, IPv6 is a necessary technology for the future of the Internet. However, nothing compels the elimination of IPv4; no protocol police will forbid its use in the foreseeable future. IPv4 may disappear due to irrelevance when IPv6 is so pervasive to make it redundant, but network operators should be prepared to operate IPv4 and IPv6 in a mixed deployment for some time. However, the techniques and mechanisms supported by a network operator can be expected to evolve and change over time as a rational goal would be to gradually shift coexistence costs (real operational expense as well as convenience) from "early adopters" of IPv6 to the shrinking pool of IPv4 maintainers.

Various techniques are required for coexistence, roughly divided into three categories:

1. Address Mapping: Many situations will require the use of address mapping to maintain scalability in the face of dwindling IPv4 global address space and to support translation and tunneling approaches.
2. Tunneling: A method for the encapsulation and transport of one protocol over or through the infrastructure that favors the other, e.g. IPv6 traffic via an IPv4 infrastructure
3. Protocol Translation: A mechanism for rewriting packets from one protocol to the other so they can be delivered as native (non-encapsulated) packets typically due to incompatible end nodes, e.g. an IPv6 client to an IPv4 server.

These categories are not mutually exclusive, as some scenarios and solutions incorporate aspects of multiple approaches.

RFC 4213 "Basic Transition Mechanisms for IPv6 Hosts and Routers" E. Nordmark and R. Gilligan October 2005
<http://tools.ietf.org/html/rfc4213>

5.1. Address Mapping

The introduction of address family translation presents challenges similar to those experienced with Network Address Translation (NAT) as it has evolved in the IPv4 Internet. The depletion of IPv4 global address space conspires with the continuing need for routable IPv4 address in some coexistence approaches to further press proliferation and scale of NAT. While alternatives exist, some network operators will continue to see the various flavors of NAT as a necessary evil, so it remains important to understand the impact on network operations, on the end-user and on applications.

Dual-Stack Lite (DS-lite) is one of the alternatives to providing dual-stack support to end-users in the face of limited global IPv4 address space.

RFC 2663 "IP Network Address Translator (NAT) Terminology and Considerations" P. Srisuresh and M. Holdrege August 1999
<http://tools.ietf.org/html/rfc2663>

This document attempts to describe the operation of NAT devices and the associated considerations in general, and to define the terminology used to identify various flavors of NAT.

5.1.1. Address Translation in Network Operations

"Common Requirements for IP Address Sharing Schemes" I. Yamagati et al. <http://tools.ietf.org/html/draft-ietf-behave-lsn-requirements>

This document defines common requirements of multiple types of Large Scale Network Address Translation (NAT) that handles Unicast UDP, TCP and ICMP.

"Issues with IP Address Sharing" M. Ford (Ed.) et al.
<http://tools.ietf.org/html/draft-ietf-intarea-shared-addressing-issues>

The completion of IPv4 address allocations from IANA and the RIRs is causing service providers around the world to question how they will continue providing IPv4 connectivity service to their subscribers when there are no longer sufficient IPv4 addresses to allocate them one per subscriber. Several possible solutions to this problem are now emerging based around the idea of shared IPv4 addressing. These solutions give rise to a number of issues and this memo identifies those common to all such address sharing approaches.

"An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", Sheng Jiang, Dayong Guo, Brian Carpenter
<http://tools.ietf.org/html/draft-ietf-v6ops-incremental-cgn>

Carrier-Grade NAT (CGN) devices with integrated transition mechanisms can reduce the operational change required during the IPv4 to IPv6 migration or coexistence period. This document proposes an incremental CGN approach for IPv6 transition. It can provide IPv6 access services for IPv6-enabled hosts and IPv4 access services for IPv4 hosts while leaving much of a legacy IPv4 ISP network unchanged. It is suitable for the initial stage of IPv4 to IPv6 migration. Unlike NAT444 based CGN alone, Incremental CGN also supports and encourages transition towards dual-stack or IPv6-only ISP networks. A smooth transition to IPv6 deployment is also described in this document.

"Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers" Bagnulo, Matthews, van Beijnum
<http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-stateful>

This document describes stateful NAT64 translation, which allows IPv6-only clients to contact IPv4 servers using unicast UDP, TCP, or ICMP. The public IPv4 address can be shared among several IPv6-only clients. When the stateful NAT64 is used in conjunction with DNS64 no changes are usually required in the IPv6 client or the IPv4 server.

"NAT64-CPE Mode Operation for Opening Residential Service" G. Chen and H. Deng
<http://tools.ietf.org/html/draft-chen-v6ops-nat64-cpe>

The authors of this draft describe the application of fundamental NAT64 functionality in CPE deployment scenarios. The approach is intended to eliminate the need for CPE to cooperate with DNS64, and to be compatible with legacy residential servers without changes to DNS requirements.

"Flexible IPv6 Migration Scenarios in the Context of IPv4 Address Shortage" M. Boucadair (Ed.) et al, October 20, 2009 (expired)
<http://tools.ietf.org/html/draft-boucadair-behave-ipv6-portrange-04>

This memo presents a solution to solve IPv4 address shortage and ease IPv4-IPv6 interconnection. The document presents a set of incremental steps for the deployment of IPv6 as a means to solve IPv4 address exhaustion. Stateless IPv4/IPv6 address mapping functions are introduced and IPv4-IPv6 interconnection scenarios presented.

This memo advocates for a more proactive approach for the deployment of IPv6 into operational networks. This memo specifies the IPv6 variant of the A+P. Both encapsulation and translation scheme are covered. Moreover, two modes are elaborated: the binding mode (compatible mode with DS-lite) and the stateless mode.

"A Note on NAT64 Interaction with Mobile IPv6" W. Haddad and C. Perkins

<http://tools.ietf.org/html/draft-haddad-mext-nat64-mobility-harmful>

This memo discusses potential NAT64 technology repercussions for mobile nodes using Mobile IPv6. An ambiguity is identified related to the use of DNS during bootstrapping, which is likely to inhibit proper signaling between mobile node and home agent.

"NAT64 for Dual Stack Mobile IPv6" B. Sarikaya and F. Xia

<http://tools.ietf.org/html/draft-sarikaya-behave-mext-nat64-dsmip>

This memo specifies how IPv6 only mobile nodes (MN) receiving host-based mobility management using Dual Stack Mobile IPv6 (DSMIPv6) can communicate with IPv4 only servers. The protocol is based on home agents maintaining a table similar to NAT64 and linking it to the binding cache. This technique avoids the problems encountered when NAT64 is used for mobile nodes in Dual Stack Mobile IPv6. How IPv6 only mobile nodes can receive multicast data from IPv4 only content providers is also explained.

"NAT64 for Proxy Mobile IPv6" B. Sarikaya and F. Xia

<http://tools.ietf.org/html/draft-sarikaya-behave-netext-nat64-pmip>

Similarly, this memo specifies how IPv6 only mobile nodes (MN) receiving network-based mobility management using Proxy Mobile IPv6 (PMIPv6) can communicate with IPv4 only servers.

5.1.2. Application and End-User Considerations With NAT

"Problem Statement for Referrals" B. Carpenter, S. Jiang and B. Zhou
<http://tools.ietf.org/html/draft-carpenter-referral-ps>

The purpose of a referral is to enable a given entity in a multiparty Internet application to pass information to another party. It enables a communication initiator to be aware of relevant information of its destination entity before launching the communication. This memo discusses the problems involved in referral scenarios.

"Referrals Across an IPv6/IPv4 Translator" D. Wing, October 19, 2009
<http://tools.ietf.org/html/draft-wing-behave-nat64-referrals-01>

While this draft is expired, this issue remains a topic of conversation, including a Bar-BoF at IETF 78. Referrals across disparate address domains may be needed for provision of services such as SIP during transition.

"Legacy NAT Traversal for IPv6: Simple Address Mapping for Premises Legacy Equipment (SAMPLE)"
<http://tools.ietf.org/html/draft-carpenter-softwire-sample>

IPv6 deployment is delayed by the existence of millions of subscriber network address translators (NATs) that cannot be upgraded to support IPv6. This document specifies a mechanism for traversal of such NATs. It is based on an address mapping and on a mechanism whereby suitably upgraded hosts behind a NAT may obtain IPv6 connectivity via a stateless server, known as a SAMPLE server, operated by their Internet Service Provider. SAMPLE is an alternative to the Teredo protocol.

"Some Considerations on the Load-Balancer for NAT64" D. Zhang et al.
<http://tools.ietf.org/html/draft-wang-behave-nat64-load-balancer>

This draft investigates issues with deploying load-balancers with NAT64 devices.

"An FTP ALG for IPv6-to-IPv4 Translation" I. van Beijnum
<http://tools.ietf.org/html/draft-ietf-behave-ftp64>

The File Transfer Protocol (FTP) has a very long history, and despite the fact that today, other options exist to perform file transfers, FTP is still in common use. As such, it is important that in the situation where some client computers are IPv6-only while many servers are still IPv4-only and IPv6-to-IPv4 translators are used to bridge that gap, FTP is made to work through these translators as best it can. This document specifies a middlebox that enables legacy usage of FTP with translation.

"Assessing the Impact of NAT444 on Network Applications" C. Donley et al. <http://tools.ietf.org/html/draft-donley-nat444-impacts>

NAT444 is an IPv4 extension technology being considered by Service Providers to continue offering IPv4 service to customers while transitioning to IPv6. This technology adds an extra Large-Scale NAT ("LSN") in the Service Provider network, thereby resulting in two NATs. CableLabs, Time Warner Cable, and Rogers Communications independently tested the impacts of NAT444 on many popular Internet services using a variety of test scenarios, network topologies, and vendor equipment. This document identifies areas where adding a

second layer of NAT disrupts the communication channel for common Internet applications.

5.1.3. Dual-Stack Lite (DS-lite)

"Understanding Dual-Stack Lite" Jeff Doyle, Network World October 22, 2009 <http://www.networkworld.com/community/node/46600>

This article provides a good introduction to DS-lite, at the time of its publication. Please see the following drafts for details and more current work.

"Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion" A. Durand et al.
<http://tools.ietf.org/html/draft-ietf-softwire-dual-stack-lite>

This document revisits the dual-stack model and introduces the dual-stack lite technology aimed at better aligning the costs and benefits of deploying IPv6 in service provider networks. Dual-stack lite enables a broadband service provider to share IPv4 addresses among customers by combining two well-known technologies: IP in IP (IPv4-in-IPv6) and Network Address Translation (NAT).

"Dual-stack Lite Mobility Solutions" B. Sarikaya and F. Xia October 11, 2009 (expired)
<http://tools.ietf.org/html/draft-sarikaya-softwire-dslitemobility-01>

Two solutions are presented to show how to use Dual-Stack Lite transition technique in mobile networks: one for Proxy Mobile IPv6 and the other for Dual-Stack Mobile IPv6. Proxy Mobile IPv6 allows IPv4 nodes to receive mobility services using an IPv4 home address. In case of client based mobility using DSMIPv6, mobile node is a dual-stack node and it can receive an IPv4 home address from the home agent which is co-located with DS-lite carrier-grade NAT.

"Scalable Operation of Address Translators with Per-Interface Bindings" J. Arkko and L. Eggert February 9, 2009 (expired)
<http://tools.ietf.org/html/draft-arkko-dual-stack-extra-lite-00>

This document explains how to employ address translation in networks that serve a large number of individual customers without requiring a correspondingly large amount of private IPv4 address space.

"Gateway Initiated Dual-Stack Lite Deployment" F. Brockners et al.
<http://tools.ietf.org/html/draft-ietf-softwire-gateway-init-ds-lite>

Gateway-Initiated Dual-Stack lite (GI-DS-lite) is a modified approach to the original Dual-Stack lite (DS-lite) applicable to certain tunnel-based access architectures. GI-DS-lite extends existing access tunnels beyond the access gateway to an IPv4-IPv4 NAT using softwires with an embedded context identifier, that uniquely identifies the end-system the tunneled packets belong to. The access gateway determines which portion of the traffic requires NAT using local policies and sends/receives this portion to/from this softwire tunnel.

"Deployment DS-lite in Point-to-Point Access Network" Y. Lee (Ed.) et al. <http://tools.ietf.org/html/draft-zhou-softwire-ds-lite-p2p>

Gateway-Initiated Dual-Stack lite (GI-DS-lite) is a proposal to logically extend existing access tunnels beyond the access gateway to DS-Lite Address Family Transition Router element (AFTR) using softwires with an embedded context identifier. This memo describes a deployment model using GI-DS-lite in Point-to-Point access network.

"Deploying Dual-Stack Lite in IPv6 Network" M. Boucadair (Ed.) et al. <http://tools.ietf.org/html/draft-boucadair-dslite-interco-v4v6>

Dual-Stack lite requires that the AFTR must have IPv4 connectivity. This forbids a service provider who wants to deploy AFTR in an IPv6-only network. This memo proposes an extension to implement a stateless IPv4-in-IPv6 encapsulation in the AFTR so that AFTR can be deployed in an IPv6-only network.

"IPv6 RA Option for DS-lite AFTR Element" Y. Lee, M. Boucadair and X. Xu <http://tools.ietf.org/html/draft-lee-6man-ra-dslite>

This document specifies a new optional extension to IPv6 Router Advertisement messages to allow IPv6 routers to advertise DS-Lite AFTR addresses to IPv6 hosts (i.e., a default IPv6 route for DS-Lite traffic). The provisioning of the AFTR address is crucial to access IPv4 connectivity services in a DS-Lite context. Means to ensure reliable delivery of this information to connecting hosts is a must.

Furthermore, this RA option can be used as a means to distribute DS-Lite serviced customers among a set of deployed AFTRs without requiring a central knowledge of the underlying topology and deployed AFTRs.

5.2. Tunneling Mechanisms

RFC 2473 "Generic Packet Tunneling in IPv6 Specification." A. Conta and S. Deering, December 1998
<http://tools.ietf.org/html/rfc2473>

This document defines the model and generic mechanisms for IPv6 encapsulation of Internet packets, such as IPv6 and IPv4. The model and mechanisms can be applied to other protocol packets as well, such as AppleTalk, IPX, CLNP, or others.

RFC 2529 "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels" B. Carpenter and C. Jung March 1999.
<http://tools.ietf.org/html/rfc2529>

This memo specifies the frame format for transmission of IPv6 packets and the method of forming IPv6 link-local addresses over IPv4 domains. The motivation for this method is to allow isolated IPv6 hosts, located on a physical link which has no directly connected IPv6 router, to become fully functional IPv6 hosts by using an IPv4 domain that supports IPv4 multicast as their virtual local link.

RFC 3056 "Connection of IPv6 Domains via IPv4 Clouds" B. Carpenter and K. Moore February 2001
<http://tools.ietf.org/html/rfc3056>

This memo specifies an optional interim mechanism for IPv6 sites to communicate with each other over the IPv4 network without explicit tunnel setup, and for them to communicate with native IPv6 domains via relay routers.

RFC 3053 "IPv6 Tunnel Broker" A. Durand, I. Guardini and D. Lento January 2001
<http://tools.ietf.org/html/rfc3053>

The IPv6 global Internet as of today uses a lot of tunnels over the existing IPv4 infrastructure. Those tunnels are difficult to configure and maintain in a large scale environment, and the process is too complex for the isolated end user. The motivation for the development of the tunnel broker model is to help early IPv6 adopters to hook up to an existing IPv6 network with stable, permanent IPv6 addresses and DNS names.

5.2.1. Teredo

RFC 4380 "Teredo: Tunneling IPv6 over UDP" C. Huitema February 2006
<http://tools.ietf.org/html/rfc4380>

This RFC defined a service that enables nodes located behind one or more IPv4 Network Address Translations (NATs) to obtain IPv6 connectivity by tunneling packets over UDP; we call this the Teredo service. Running the service requires the help of "Teredo servers" and "Teredo relays". The Teredo servers are stateless, and only have to manage a small fraction of the traffic between Teredo clients; the Teredo relays act as IPv6 routers between the Teredo service and the "native" IPv6 Internet. The relays can also provide interoperability with hosts using other transition mechanisms such as "6to4". Teredo client capability has been included in Windows operating systems since Windows XP and public servers are available.

RFC 5991 "Teredo Security Extensions" D. Thaler, S. Krishnan and J. Hoagland September 2010
<http://tools.ietf.org/html/rfc5991>

The Teredo protocol defines a set of flags that are embedded in every Teredo IPv6 address. This document specifies a set of security updates that modify the use of this flags field, but are backward compatible.

"Teredo Extensions", D. Thaler
<http://tools.ietf.org/html/draft-thaler-v6ops-teredo-extensions>

This document specifies a set of extensions to the Teredo protocol. These extensions provide additional capabilities to Teredo, including support for more types of Network Address Translations (NATs), and support for more efficient communication.

5.2.2. IPv6 Rapid Deployment (6rd) and Extensions

IPv6 Rapid Deployment (6rd) is an approach that allows a service provider to quickly roll out an IPv6 service offering. Free, a large French ISP, successfully deployed a 6rd offering in 5 weeks. It is also being used in a current IPv6 trial offered by Comcast in the USA.

"How 6rd Eases the Transition to IPv6" Mike Capuano on Cisco SP360 blog, August 5, 2010
http://blogs.cisco.com/sp/how_6rd_eases_the_transition_to_ipv6/

This article provides a quick overview of 6rd. The fundamental protocol specification and initial implementation experience can be found in RFC 5969 and 5569.

RFC 5969 "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)-Protocol Specification" W. Townsley and O. Troan August 2010
<http://tools.ietf.org/html/rfc5969>

RFC 5569 "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)" R. Despres January 2010 <http://tools.ietf.org/html/rfc5569>

"IPv6 Across NAT44 CPEs (6a44)" R. Despres, B. Carpenter and S. Jiang
<http://tools.ietf.org/html/draft-despres-softwire-6a44>

IPv6 Across NAT44 CPEs (6a44) 6a44 is based on an address mapping and on a mechanism whereby suitably upgraded hosts behind a NAT may obtain IPv6 connectivity via a stateless 6a44 server function operated by their Internet Service Provider. With it, traffic between two 6a44 hosts in a single site remains within the site. Except for IANA numbers that remain to be assigned, the specification is intended to be complete enough for running codes to be independently written and interwork.

[Note that this draft converges and supersedes work started in two separate drafts, which are no longer relevant:
<http://tools.ietf.org/html/draft-despres-softwire-6rdplus-00>
<http://tools.ietf.org/html/draft-carpenter-softwire-sample-00>]

"UDP Encapsulation of 6rd" Y. Lee and P. Kapoor
<http://tools.ietf.org/html/draft-lee-softwire-6rd-udp-02>

This memo specifies the UDP encapsulation to IPv6 Rapid Deployment (6rd) protocol which enables hosts behind unmodified Home Gateway device to access 6rd service. One variation (Server Model) avoids host modification by offloading the implementation to a small server (relay) on the home LAN.

"Gateway Initiated 6rd" T. Tsou et al.
<http://tools.ietf.org/html/draft-tsou-softwire-gwinit-6rd>

This document proposes an alternative to the deployment model defined in RFC 5969 for 6rd. This model extends existing access tunnels beyond an operator-owned gateway collocated with the operator's IPv4 network edge to the Border Router. This modification makes it unnecessary to provide IPv4 routes to IPv6 UEs. The gateway serves as an aggregation point for IPv4 routing.

5.2.3. Tunnel Support Protocol (TSP)

RFC 5572 "IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP)" M. Blanchet and F. Parent, February 2010
<http://tools.ietf.org/html/rfc5572>

TSP is an Experimental RFC defining a method for a tunnel client to negotiate tunnel characteristics with a tunnel broker. It enables tunnels in various deployment architectures including NAT traversal and mobility, and for user authentication it utilizes:

RFC 4422 "Simple Authentication and Security Layer (SASL)" A. Melnikov and K. Zeilenga(Eds.) June 2006
<http://tools.ietf.org/html/rfc4422>

5.2.4. Residual IPv4 Deployment over IPv6-only Infrastructure

Further down the transition road, operators may desire to retire IPv4 routing support and move their backbone networks to IPv6-only. There may be residual IPv4 legacy customers (clients and servers) still requiring the delivery of IPv4 packets. While the previously proposed Dual-Stack Transition Mechanism (DSTM) approach attempted to satisfy this use case, it was complex and stateful. A stateless approach to IPv4 residual deployment (4rd) is defined in section 3.2 of the Stateless Address Mapping (SAM) draft. At the time of this publication, several network operators in Japan are planning implementation to support residual IPv4 customers.

"Stateless Address Mapping (SAM) - a Simplified Mesh-Software Model" Despres, R. July 12, 2010
<http://tools.ietf.org/html/draft-despres-softwire-sam>

"IPv4 Residual Deployment across IPv6-Service networks (4rd): A NAT-less Solution" R. Despres
<http://tools.ietf.org/html/draft-despres-softwire-4rd>

5.2.5. Address Plus Port (AplusP)

"The A+P Approach to the IPv4 Address Shortage" R. Bush (Ed.) October 27, 2009 (expired, but authors indicate a new draft is coming)
<http://tools.ietf.org/html/draft-ymbk-aplusp>

This draft discusses the possibility of address sharing by treating some of the port number bits as part of an extended IPv4 address (Address plus Port, or A+P). Instead of assigning a single IPv4

address to a customer device, we propose to extend the address by "stealing" bits from the port number in the TCP/UDP header, leaving the applications a reduced range of ports. This means assigning the same IPv4 address to multiple clients (e.g., CPE, mobile phones), each with its assigned port-range. In the face of IPv4 address exhaustion, the need for addresses is stronger than the need to be able to address thousands of applications on a single host. If address translation is needed, the end-user should be in control of the translation process - not some smart boxes in the core.

"Aplusp Lite - A light weight aplusp approach" Z. Xiaoyu
<http://tools.ietf.org/html/draft-xiaoyu-aplusp-lite>

This document proposes a solution aimed at providing IPv4 continuity in IPv6 environment. The proposed solution is expected to alleviate the public IPv4 depletion problem while maximize the benefits from IPv6 deployment, and meet the desired service availability and reliability with affordable cost.

5.2.6. IRON-RANGER and ISATAP Solutions

A body of RFCs and drafts in progress provide an alternative approach to IPv4/IPv6 coexistence. This approach utilizes tunneling techniques to create "overlay" networks. While currently considered "Experimental" it may be of interest to network operators as an alternative network architecture.

RFC 5214 "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)"
F. Templin et al. March 2008 <http://tools.ietf.org/html/rfc5214>

RFC 5579 "Transmission of IPv4 Packets over Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) Interfaces" F. Templin (Ed.)
February 2010 <http://tools.ietf.org/html/rfc5579>

RFC 5320 "The Subnetwork Encapsulation and Adaptation Layer (SEAL)"
F. Templin (Ed.) February 2010 <http://tools.ietf.org/html/rfc5320>

Fred Templin originally published SEAL as an Experimental RFC, and is currently updating with the intention to publish as Standards Track:
<http://tools.ietf.org/html/draft-templin-intarea-seal>

RFC 5558 "Virtual Enterprise Traversal (VET)" F. Templin (Ed.)
February 2010 <http://tools.ietf.org/html/rfc5558>

Fred Templin originally published VET as an Informational RFC, and is currently updating with the intention to publish as Standards Track:
<http://tools.ietf.org/html/draft-templin-intarea-vet>

RFC 5720 "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER)" F. Templin (Ed.) February 2010
<http://tools.ietf.org/html/rfc5720>

"The Internet Routing Overlay Network (IRON)" F. Templin (Ed.)
<http://tools.ietf.org/html/draft-templin-iron>

5.2.7. Softwires Hub and Spoke with L2TP

RFC 5571 "Softwire Hub and Spoke Deployment Framework with Layer Two Tunneling Protocol Version 2 (L2TPv2)" B. Storer et al. June 2009
<http://tools.ietf.org/html/rfc5571>

This document describes the framework of the Softwire "Hub and Spoke" solution with the Layer Two Tunneling Protocol version 2 (L2TPv2). The implementation details specified in this document should be followed to achieve interoperability among different vendor implementations.

5.3. Translation

From the earliest specification of IPv6 IETF contributors have recognized that translation would be a necessary tool for transition and coexistence, as IPv6 was designed as an incompatible replacement rather than an extension of IPv4. The original approach to stateless translation defined in RFC 2765 and its implementation as NA(P)T-PT as described in RFC 2766 had a number of issues that resulting in the approach being deprecated by RFC 4966. Recently the Behave WG has taken on the work of defining a set of scenarios covering the use cases for translation, prioritizing the work and defining new solutions that overcome the deficiencies of the historic approach.

5.3.1. Historic Approach

RFC 2765 "Stateless IP/ICMP Translation (SIIT)." E. Nordmark, February 2000 <http://tools.ietf.org/html/rfc2765>

This document specifies a transition mechanism algorithm in addition to the mechanisms already specified in RFC 1933 (note that this reference was subsequently obsoleted by RFC 2893 which in turn was obsoleted by RFC 4213). The algorithm translates between IPv4 and IPv6 packet headers (including ICMP headers) in separate translator "boxes" in the network without requiring any per-connection state in those "boxes". This new algorithm can be used as part of a solution that allows IPv6 hosts, which do not have a permanently assigned IPv4 addresses, to communicate with IPv4-only hosts. The document neither

specifies address assignment nor routing to and from the IPv6 hosts when they communicate with the IPv4-only hosts.

SIIT has been applied in several translation implementations, including the historic NAT-PT specified in RFC 2766 and deprecated by RFC 4966. SIIT is currently being revised in "IP/ICMP Translation Algorithm" X. Li, C. Bao and F. Baker
<http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate>

RFC 2766 "Network Address Translation - Protocol Translation (NAT-PT)." G. Tsirtsis and P. Srisresh, February 2000
<http://tools.ietf.org/html/rfc2766>

This solution attempted to provide transparent routing to end-nodes in an IPv6 realm trying to communicate with end-nodes in an IPv4 realm and vice versa. This combined Network Address Translation and Protocol Translation. While it did mandate dual-stack support or special purpose routing requirements (such as requiring tunneling support) on end nodes, it did introduce issues that were considered harmful enough to lead to its deprecation in July 2007 by RFC 4966 "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status" <http://tools.ietf.org/html/rfc4966>.

RFC 2767 "Dual-Stack Hosts Using 'Bump in the Stack' Technique (BIS)" K. Tsuchiay, H. Higuchi and Y. Atarashi February 2000

RFC 3338 "Dual-Stack Hosts Using 'Bump in the API' (BIA)" S. Lee, et al. October 2002
<http://tools.ietf.org/html/rfc3338>

These two RFCs are proposed for obsolescence by a draft that combines both:

"Dual-Stack Hosts Using 'Bump in the Host' (BIH)" B. Huang, H. Deng and T. Savolainen
<http://tools.ietf.org/html/draft-ietf-behave-v4v6-bih>

5.3.2. Current Translation Approaches

A renewed effort to define new translation mechanisms started with discussions in the Internet Area (intarea) meeting and the Technical Plenary at IETF 71 in Dublin, and continued at a special meeting in Montreal in October 2008. This led to a commitment by contributors in the Behave WG to take on the work. A set of scenarios were defined along with a framework for the translation solutions.

"IPv4 Run-Out and IPv4-IPv6 Co-Existence Scenarios" J. Arkko and M. Townsley

<http://tools.ietf.org/html/draft-arkko-townsley-coexistence>

When IPv6 was designed, it was expected that the transition from IPv4 to IPv6 would occur more smoothly and expeditiously than experience has revealed. The growth of the IPv4 Internet and predicted depletion of the free pool of IPv4 address blocks on a foreseeable horizon has highlighted an urgent need to revisit IPv6 deployment models. This document provides an overview of deployment scenarios with the goal of helping to understand what types of additional tools the industry needs to assist in IPv4 and IPv6 co-existence and transition.

This document was originally created as input to the Montreal co-existence interim meeting in October 2008, which led to the rechartering of the Behave and Softwire working groups to take on new IPv4 and IPv6 coexistence work. This document is published as a historical record of the thinking at the time.

"A Framework for IPv4/IPv6 Translation" F. Baker et al.

<http://tools.ietf.org/html/draft-ietf-behave-v6v4-framework>

This draft (Framework) is the place to start to understand the historic context for translation, the definition and rationale for the set of translation scenarios and canonical definitions for some of the terminology that arises when talking about translation and coexistence in general.

The 4 deployment modes for these scenarios are:

1. Connecting between the IPv4 Internet and the IPv6 Internet
2. Connecting an IPv6 network to the IPv4 Internet
3. Connecting an IPv4 network to the IPv6 Internet
4. Connecting between an IPv4 network and an IPv6 network

As solutions may differ with respect to the initiating end of the conversation, 8 scenarios are defined in the Framework draft, as recapped in the following sections along with specifications that fit each scenario.

Some general specifications that are cited in the various solution specifications (or may be in subsequent revisions) are:

"IPv6 Addressing of IPv4/IPv6 Translators" C. Bao et al. August 16, 2010 <http://tools.ietf.org/html/draft-ietf-behave-address-format-10>

"DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers" M. Bagnulo et al. July 5, 2010 <http://tools.ietf.org/html/draft-ietf-behave-dns64-10>

"Analysis of 64 Translation" R. Penno, T. Saxena and D. Wing <http://tools.ietf.org/html/draft-penno-behave-64-analysis>

Due to specific problems, NAT-PT was deprecated by the IETF as a mechanism to perform IPv6-IPv4 translation. Since then, new effort has been undertaken within IETF to standardize alternative mechanisms to perform IPv6-IPv4 translation. This document evaluates how the new translation mechanisms avoid the problems that caused the IETF to deprecate NAT-PT.

5.3.2.1. An IPv6 network to the IPv4 Internet

The Framework defines Scenario 1 for an early adopter (end user or network operator) which establishes an IPv6 network and needs to maintain access to the global IPv4 Internet, preferably without assigning IPv4 addresses to the nodes of the IPv6 network. Either the Stateful or Stateless solutions proposed may satisfy this deployment scenario.

"Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers" M. Bagnulo, P. Matthews and I. van Beijnum <http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-stateful>

"IP/ICMP Translation Algorithm" X. Li, C. Bao and F. Baker <http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate>

5.3.2.2. The IPv4 Internet to an IPv6 network

The Framework defines Scenario 2 for a node on the IPv4 Internet initiating a transmission to a node on an IPv6 network. The original approach to this deployment was the NAT-PT implementation of SIIT (as defined in RFC 2766) which has been deprecated (by RFC 4966). The Stateless Translation solution for Scenario 1 also would work for this case as it does support IPv4-initiated communication with a subset of IPv6 addresses.

5.3.2.3. The IPv6 Internet to an IPv4 network

The Framework defines Scenario 3 where a legacy IPv4 network has a requirement to provide services to users in the IPv6 Internet.

Stateful Translation with static AAAA records in DNS to represent the IPv4-only hosts will work.

"Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers" M. Bagnulo, P. Matthews and I. van Beijnum
<http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-stateful>

"DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers" M. Bagnulo et al.
<http://tools.ietf.org/html/draft-ietf-behave-dns64>

Alternatively, host-based translation (BIH) or tightly-coupled translators may be considered.

5.3.2.4. An IPv4 network to the IPv6 Internet

Scenario 4 is not easy to solve but fortunately will not arise until significant IPv6 uptake. In-network translation is not viable, and other techniques should be considered including host-based translation (BIH) or tightly-coupled translators that adapt legacy hosts or networks to the IPv6 Internet.

5.3.2.5. An IPv6 network to an IPv4 network

Scenario 5 describes a configuration where both the IPv6 network and IPv4 network are within the administrative control of the same organization. It appears amenable to the same solutions proposed for Scenario 1.

5.3.2.6. An IPv4 network to an IPv6 network

Scenario 6 is the mirror image of Scenario 5, with communication initiated from the IPv4 side. It appears amenable to the same solution proposed for Scenario 2.

5.3.2.7. The IPv6 Internet to the IPv4 Internet

The Framework indicates that Scenario 7, the interconnection of the IPv4 Internet with the IPv6 Internet may appear to be an ideal case for an in-network translator (such as the deprecated NAT-PT), but there is no viable way to map the immense IPv6 address space onto IPv4. This situation would not entail until significant IPv6 adoption, and has not been a priority for solution.

5.3.2.8. The IPv4 Internet to the IPv6 Internet

Scenario 8 presents a challenge similar to Scenario 7.

5.4. Connectivity Checking and Delay Avoidance

One important issue that arises in a coexistence environment is negative impact on the initiation of peer-to-peer connections, such as VoIP, video, etc. The initiator doesn't know a priori whether the peer is using the same address family incurring a possible delay as the first attempt may fail. There is also ambiguity, as the IPv6 path may be temporarily broken.

"IPv6 Connectivity Check and Redirection by HTTP Servers" E. Vyncke
<http://tools.ietf.org/html/draft-vyncke-http-server-64aware>

Rather than forcing the client to decide whether IPv4 or IPv6 is more convenient to reach a web server; this document proposes to let the web server check whether there is IPv6 connectivity to the client; then the web server can do a HTTP redirect to force the client to use IPv6.

This is done easily by a script within the server HTML pages and does not require any change in the client applications or configuration. The client still can control whether he/she wants to enable IPv6.

"Happy Eyeballs: Trending Towards Success (IPv6 and SCTP)", D. Wing, A. Yourtchenko, P. Natarajan.
<http://tools.ietf.org/html/draft-wing-http-new-tech>

This draft makes several recommendations to ensure user satisfaction and a smooth transition from HTTP's pervasive IPv4 to IPv6 and from TCP to SCTP. While the target audience is app developers and content providers, network operators should be aware of techniques needed to maintain peaceful coexistence without negative impact on end-user perception of service level.

"Migrating SIP to IPv6 Media Without Connectivity Checks" D. Wing, A. Yourtchenko
<http://tools.ietf.org/html/draft-wing-dispatch-v6-migration>

During the migration from IPv4 to IPv6, it is anticipated that an IPv6 path might be broken for a variety of reasons, causing endpoints to not receive RTP data. Connectivity checks would detect and avoid the user noticing such a problem, but there is industry reluctance to implement connectivity checks.

This document describes a mechanism allowing dual-stack SIP endpoints to attempt communications over IPv6 and fall back to IPv4 if the IPv6 path is not working. The mechanism does not require connectivity checks.

6. Prefix and Address Assignment and Distribution

RFC 4291 "IP Version 6 Addressing Architecture." R. Hinden, S. Deering. February 2006.
<http://tools.ietf.org/html/rfc4291>

RFC 5952 "A Recommendation for IPv6 Text Representation" S. Kawamura and M. Kawashima, August 2010
<http://tools.ietf.org/html/rfc5952>

RFC 4291 defines the addressing architecture of the IP Version 6 (IPv6) protocol. The document includes the IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and an IPv6 node's required addresses. RFC 5952 updates RFC 4291 with a recommended method for rendering IPv6 addresses in a standard form for user interfaces, logging and reporting.

"IPv6 Addressing of IPv4/IPv6 Translators" C. Bao et al. (Status: Standards Track, in RFC Editor will update RFC 4291)
<http://tools.ietf.org/html/draft-ietf-behave-address-format>

This document discusses the algorithmic translation of an IPv6 address to a corresponding IPv4 address, and vice versa, using only statically configured information. It defines a well-known prefix for use in algorithmic translations, while allowing organizations to also use network-specific prefixes when appropriate. Algorithmic translation is used in IPv4/IPv6 translators, as well as other types of proxies and gateways (e.g., for DNS) used in IPv4/IPv6 scenarios.

RFC 3177 "IAB/IESG Recommendations on IPv6 Address Allocations to Sites." IAB, IESG. September 2001.
<http://tools.ietf.org/html/rfc3177>

RFC 3177 provides recommendations to the addressing registries (APNIC, ARIN and RIPE-NCC) on policies for assigning IPv6 address blocks to end sites. In particular, it recommends the assignment of /48 in the general case, /64 when it is known that one and only one subnet is needed and /128 when it is absolutely known that one and only one device is connecting.

"IPv6 Address Assignment to End Sites", T. Narten, G. Huston, R. Roberts, 12-Jul-10
<http://tools.ietf.org/html/draft-ietf-v6ops-3177bis-end-sites>

The proposed update to RFC 3177 revises the recommendation to leave the exact choice to the operational community. The role of the IETF

is limited to providing guidance on IPv6 architectural and operational considerations. This document reviews the architectural and operational considerations of end site assignments as well as the motivations behind the original 3177 recommendations. Moreover, the document clarifies that a one-size-fits-all recommendation of /48 is not nuanced enough for the broad range of end sites and is no longer recommended as a single default.

RFC 4192 "Procedures for Renumbering an IPv6 Network without a Flag Day" F. Baker, E. Lear and R. Droms
<http://www.ietf.org/rfc/rfc4192.txt>

RFC 5942 "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes." H. Singh, W. Beebee, E. Nordmark. July 2010.
<http://tools.ietf.org/html/rfc5942>

IPv6 specifies a model of a subnet that is different than the IPv4 subnet model. The subtlety of the differences has resulted in incorrect implementations that do not interoperate. This document spells out the most important difference: that an IPv6 address isn't automatically associated with an IPv6 on-link prefix. This document also updates (partially due to security concerns caused by incorrect implementations) a part of the definition of "on-link" from RFC 4861.

RFC 4862 "IPv6 Stateless Address Autoconfiguration." S. Thomson, T. Narten, T. Jinmei. September 2007.
<http://tools.ietf.org/html/rfc4862>

RFC 4941 "Privacy Extensions for Stateless Address Autoconfiguration in IPv6." T. Narten, R. Draves, S. Krishnan. September 2007.
<http://tools.ietf.org/html/rfc4941>

The IPv6 addressing architecture presumes that the remaining 64 bits are an endpoint interface identifier. This could be the MAC Address (EUI-64 Address) in an appropriate encoding, or it could be what is called a "privacy address", which is a random number. You will find the most common approach to that, for hosts, in this RFC.

RFC 3315 "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)." R. Droms (Ed.), J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney. July 2003. <http://tools.ietf.org/html/rfc3315>

"Analysis of Solution Proposals for hosts to learn NAT64 Prefixes" J. Korhonen (Ed.) and T. Savolainen
<http://tools.ietf.org/html/draft-korhonen-behave-nat64-learn-analysis>

Hosts and applications may benefit from the knowledge if an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. This document analyses number of proposed solutions for communicating if the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. This enables both NAT64 avoidance and intentional utilization by allowing local IPv6 address synthesis.

7. How-to, Whitepapers and FAQs

"IPv6 Rollout: Where do we start?" O. Crepin-Leblond
<http://www.slideshare.net/ocl999/suggestion-for-an-ipv6-roll-out>

"Everything Sysadmin" T. Limoncelli
<http://everythingsysadmin.com/2009/01/google-enables-ipv6-for-most-s.html>
<http://everythingsysadmin.com/2010/08/methods-of-converting-to-ipv6.html>

"IPv6 Deployment in Internet Exchange Points (IXPs)", Roque Gagliano
<http://tools.ietf.org/html/draft-ietf-v6ops-v6inixp>

This draft suggests that in an Internet Exchange Point one might use an address that helps in debugging routing exchanges. One could also look at what other folks do, embedding identifying marks in addresses. For example, Facebook includes "face:b00c" in the IID portion of their address.

8. Experiments, Trials and Prototypes

6bone (concluded)
<http://go6.net/ipv6-6bone/>

Hurricane Electric (ongoing)
<http://www.he.net/>

T-Mobile USA (ongoing)
<http://groups.google.com/group/tmoipv6beta>

Comcast (ongoing)
<http://www.comcast6.net/>

Internode ADSL (Ongoing)
<http://ipv6.internode.on.net/access/adsl/>

Verizon FiOS (small scale test - concluded)
<http://newscenter.verizon.com/press-releases/verizon/2010/verizon-begins-testing-ipv6.html>

"Considerations for Stateless Translation (IVI/dIVI) in Large SP Network" Q. Sun et al.
<http://tools.ietf.org/html/draft-sunq-v6ops-ivi-sp>

In addition to the deployment use case this draft describes, the draft documents an experimental use of the translation in a research network.

Measurements of IPv6 Path MTU Discovery Behavior
http://www.ripe.net/ripe/meetings/ripe-60/presentations/Stasiewicz-Measurements_of_IPv6_Path_MTU_Discovery_Behaviour.pdf

9. Implementation Reports

"A Basic Guideline for Listing ISPs that Run IPv6" S. Kawamura
<http://tools.ietf.org/html/draft-kawamura-ipv6-isp-listings>

This draft attempts to gather information about currently known sites that rate ISP readiness for IPv6 and to look at their evaluation methods. This document also summarizes basic guidelines that these listings may consider when checking an ISPs IPv6 readiness. As the draft says, there are many opinions about what it means to be ready for IPv6, and it would be helpful to evaluate ISPs based on some common criteria.

IPv6 Rapid Deployment
<http://tools.ietf.org/html/rfc5569>

Google has hosted a meeting of IPv6 Implementers in 2009 and 2010, several presentations covered experimental or live transition experience.

<https://sites.google.com/site/ipv6implementors/2009/agenda>

<https://sites.google.com/site/ipv6implementors/2010/agenda>

10. Books on IPv6

Blanchet, Marc. "Migrating to IPv6: a Practical Guide to Implementing IPv6 in Mobile and Fixed Networks." Chichester, England: J. Wiley & Sons, 2006. Print.

Hagen, Silvia. "IPv6 Essentials - Second Edition" Sebastapol, CA: O'Reilly Media, Inc, 2006. Print.

Loshin, Peter. "IPv6, Second Edition: Theory, Protocol and Practice"
Morgan Kaufmann Publishing, 2003

Popoviciu, Ciprian, Eric Levy-Abengoli and Patrick Grossetete
"Deploying IPv6 Networks" Indianapolis, IN: Cisco Press, 2006.
Print.

Siil, Karl A. "IPv6 Mandates: Choosing a Transition Strategy,
Preparing Transition Plans, and Executing the Migration of a Network
to IPv6." Indianapolis, IN: Wiley, 2008. Print.

11. Miscellaneous

See the Dancing Turtle, but only if you have native IPv6!
<http://www.kame.net/>

A little more detail than a Dancing Turtle, on your IPv6 readiness
can be obtained by using this site put up by Jason Fesler:
<http://test-ipv6.com/>

There is an extension for Firefox (and perhaps other browsers) that
displays the IP address of web pages you visit, clearly indicating
when you are connected via IPv4 or IPv6. In Firefox, click on
Tools..Add-ons..Extensions and search for ShowIP.

Eric Vyncke is collecting some statistics on IPv6 penetration.
<http://www.vyncke.org/ipv6status/>

A reasonable estimation of how fast the sky is falling.
<http://www.potaroo.net/tools/ipv4/>

A graphical representation of IPv4 depletion.
<http://www.ipv4depletion.com/old.html>

"IPv6 Adoption Remains Slow, Survey Says" W. Jackson, GCN Sept. 5,
2101
<http://gcn.com/articles/2010/09/14/adoption-of-ipv6-is-slow.aspx>
<http://www.nro.net/documents/GlobalIPv6SurveySummaryv2.pdf>

Some troubling, yet interesting news about what operators and end-
user organizations are thinking about IPv6 adoption at this time.

A study of some of the brokenness around Path MTU Discovery
[http://www.ripe.net/ripe/meetings/ripe-60/presentations/Stasiewicz-
Measurements_of_IPv6_Path_MTU_Discovery_Behaviour.pdf](http://www.ripe.net/ripe/meetings/ripe-60/presentations/Stasiewicz-Measurements_of_IPv6_Path_MTU_Discovery_Behaviour.pdf)

Cluenet hosts a mailing list with IPv6 operator participation. Various transition-related topics are brought up there from time to time.
<http://lists.cluenet.de/mailman/listinfo/ipv6-ops>

"IPv6 for Dummies, Part 1: It's Time!"
<http://www.xtranormal.com/watch/7201125/>

"IPv6 for Dummies, Part 2: Comparing IPv4 and IPv6"
<http://www.xtranormal.com/watch/7210035/>

12. Security Considerations

This draft does not introduce any security considerations.

13. IANA Considerations

This draft does not require any action from IANA.

[Note to RFC Editor: this section may be removed.]

14. Conclusions

This draft is merely the starting point for a network operator planning an IPv6 rollout. The intention of the editor was to document the great work that is already available that can help in the process and to perhaps save a few hours of redundant effort for someone to find this information. Of course, this will be out of date before it is published as active research continues in coexistence and transition tools. The editor hopes it is at least a useful "You Are Here" map to help navigate the thrill rides available in the IPv6 theme park.

This compendium could serve as an initial set of data to populate an active database or wiki. This would allow continuing community contribution including feedback on the real-world experience of network operators as they turn on IPv6.

15. References

15.1. Normative References

None.

15.2. Informative References

Complete reference information is included in the body of the draft.

16. Acknowledgments

This bibliography is a recapitulation of the contributions of the authors of the cited RFCs, drafts, websites and other publications and many folks on the v6ops and v4v6transition mailing lists, the editor has freely borrowed abstract and summary text from the cited works and e-mail postings. In addition, the editor wishes to acknowledge significant contributions and suggestions from Fred Baker, Brian Carpenter, Remi Despres, Suresh Krishnan, Tina Tsou, Yiu Lee, Marc Blanchet, Med Boucadair, Fred Templin, Andrew Yourtchenko and many contributors on the v4v6trans mailing list. All credit is due to those contributors while the editor takes responsibility for any errors, omissions or mischaracterization of the work in the process of abstracting and summarizing it here.

The IPv4-IPv6 Transition mailing list archive can be found at: <https://www.ietf.org/mailman/listinfo/v4tov6transition> and the readers are also directed to the mailing list archives of the various IETF Working Groups mentioned for the history of the cited drafts and RFCs.

This document was prepared using 2-Word-v2.0.template.dot.

Author's Address

Edward J. Jankiewicz
SRI International, Inc.
333 Ravenswood Ave
Menlo Park, CA USA

Phone: 732-389-1003 or 650-859-2000
Email: edward.jankiewicz@sri.com

Behave Work Group
Internet Draft
Intended status: Standard Track
Expires: January 07, 2011

S. Jiang
D. Gu
Huawei Technologies Co., Ltd
July 09, 2010

Multicast Proxy in IPv6/IPv4 Transition
draft-jiang-behave-v4v6mc-proxy-01.txt

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 07, 2011.

Copyright Notice

Copyright (c) 2010 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.

Abstract

During the long co-existing period of IPv6 and IPv4, the interoperation between IPv6 network and IPv4 network is essential. Multicast services across IPv6 and IPv4 networks are also needed. Besides the packet-based multicast translation mechanism, this document describes a multicast proxy solution. The multicast proxy is deployed at the border of IPv6/IPv4 networks. It is mainly based on content cache concept. Without packet-based translation, it retires the content data from IPvX network, caches the data, and multicasts the data in IPvY network. It acts as a multicast leaf in the IPvX network where the data source locates. It also acts as a multicast source in IPvY network where the multicast client locates.

Table of Contents

1. Introduction.....	3
2. Terminology.....	3
3. Multicast Proxy without packet-based IPv6/IPv4 Translation....	3
3.1. Overview.....	3
3.2. Operation procedure.....	5
4. Security Considerations.....	6
5. IANA Considerations.....	7
6. Change Log [RFC Editor please remove].....	7
7. References.....	7
7.1. Normative References.....	7
7.2. Informative References.....	7
Author's Addresses.....	8

1. Introduction

The confirmation of IPv4 address exhaustion clearly indicates that global IPv6 deployment is inevitably going to happen. However, it is also widely agreed that IPv4 will be still in use for a long period. During the long co-existing period of IPv6 and IPv4, the interoperation between IPv6 network and IPv4 network is essential.

Now, multimedia has been deployed widely, such as IPTV and video conference etc. They also face the IPv6 and IPv4 intercommunication issues. The multicast applications are complicated and face more difficulties than unicast applications deployment.

[I-D.draft-venaas-behave-v4v6mc-framework] proposes a packet-based translation framework between IPv4/IPv6 multicast services. It describes the packet-based translation operations and intercommunication in network layer to support a single source send to multiple receivers in different IP networks.

Besides the packet-based multicast translation mechanism, this document describes a multicast proxy solution, which is mainly based on content cache.

A multicast proxy can be deployed at the border between IPv4/IPv6 networks. Without packet-based translation, it retrieves the content data from IPvX network, caches the data, and multicasts the data in IPvY network. It acts as a multicast leaf in the IPvX network where the data source locates. It also acts as a multicast source in IPvY network where the multicast client locates.

2. Terminology

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

3. Multicast Proxy without packet-based IPv6/IPv4 Translation

3.1. Overview

Within this document, we describe the network where the data source locates as IPvX network and the network where the multicast client locates as IPvY network. When IPvX is IPv6, IPvY must be IPv4, vice versa.

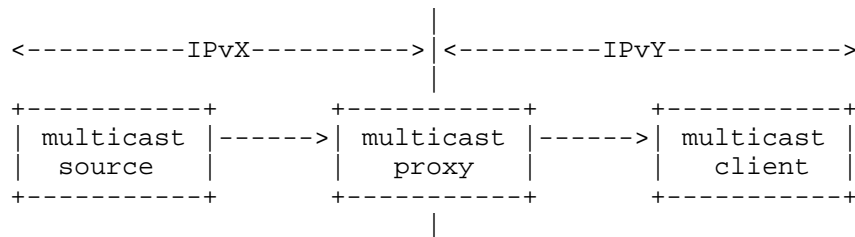


Figure 1: Multicast proxy Forward Contents to different IP networks

As showed in Figure 1, the proposed multicast proxy is deployed at the border of IPv4/IPv6 networks. It MUST support both IPv4 and IPv6. It MUST support both IGMP (Internet Group Management Protocol, [RFC3376]), which is used for IPv4 multicast management functions, and MLD (Multicast Listener Discovery, [RFC3810]), which is used in a similar way in IPv6 Environment. In the IPvX network, the multicast proxy joins the multicast distribution tree as a leaf. In the IPvY network, the multicast proxy broadcasts contents as a multicast source. The establishment of multicast distribution trees obeys the current multicast specifications for each IP family, such as Protocol Independent Multicast (PIM [RFC4601]).

Notice that there are two different multicast distribution trees in two sides of the multicast proxy. They are operational independent from each other in the network layer.

Logically, they are relevant to each other and there are interoperation behaviors between them. The contents published through the multicast distribution tree in IPvY network inherits from the IPvX network. They are received by the multicast proxy, which is a multicast leaf in the multicast distribution tree in IPvX network. Within the multicast proxy, contents are mapped between receiver function and publisher function. The operations of the multicast distribution tree in IPvY network MAY trigger some operations of the multicast distribution tree in IPvX network. For example, a multicast client joins a multicast group in IPvY network, and requests multicast contents may cause the multicast proxy joins a multicast group in IPvX network.

However, as mentioned earlier, in network or IP layer, the two multicast distribution trees are independent from each other. Their operations are separated in two sides of the multicast proxy. Conceptually, the multicast proxy can be presented virtually in functional modules like below Figure 2.

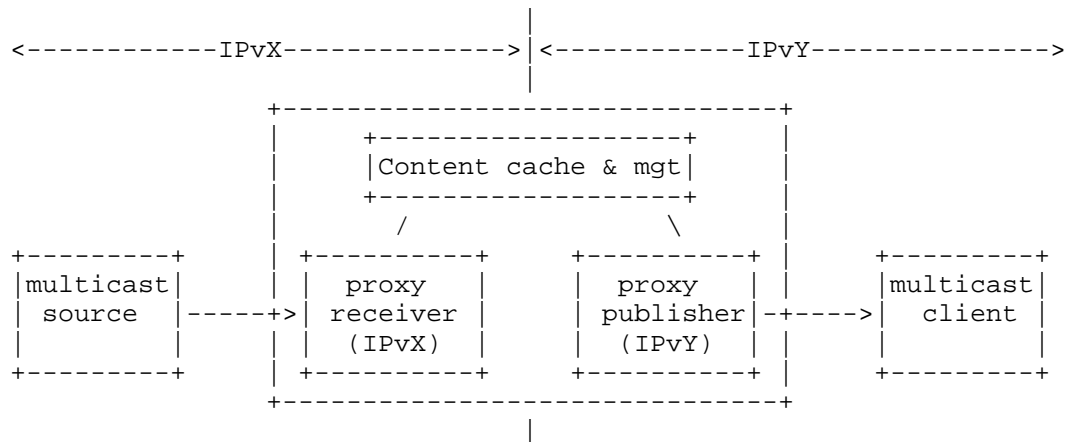


Figure 2: Separate function model of multicast proxy

As shown in Figure 2, the proxy receiver module in IPvX network joins IPvX multicast groups as a receiver client. Thereby it receives packets bound for the IPvX multicast groups, and then hands the content to the content cache and management module. The content cache and management module then forward the on-demand content to the multicast proxy function module in IPvY network, which acts as a publisher and multicast source in IPvY network.

3.2. Operation procedure

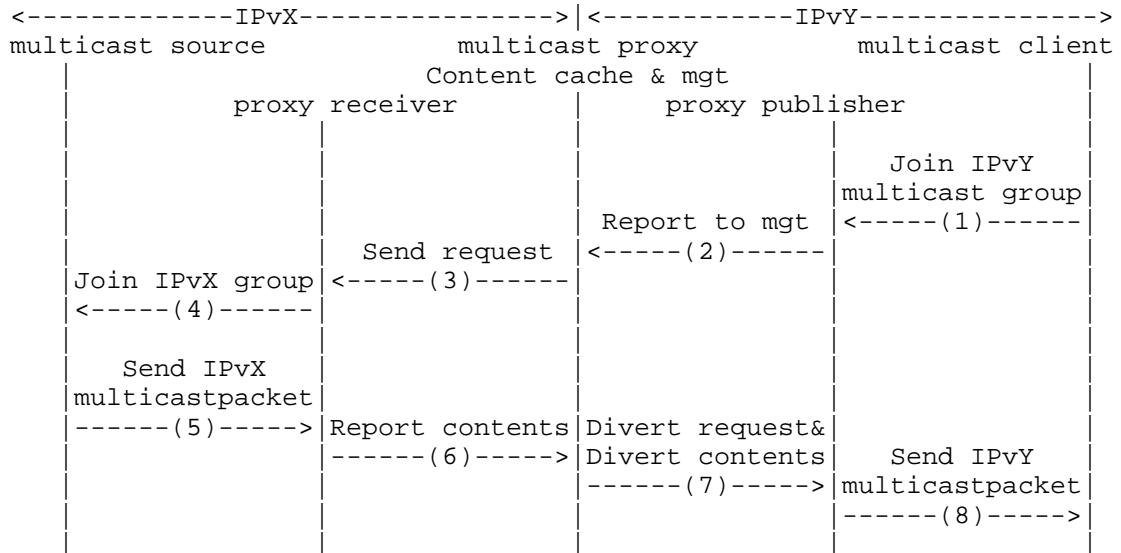


Figure. 3 The interaction communicating from IPvX to IPvY

As shown in Figure 3, a client, which locates in IPvY network, connects to the multicast proxy, requesting a multicast service whose source locates in the IPvX network.

First, the client sends a join IPvY multicast group report (1) to the multicast proxy. The proxy publisher module, which also locates in the IPvY network, receives this report, then forwards the content request to the content cache & management module (2). The content cache & mgt module maintains a content & multicast service table, including all available multicast services from IPvX network. The content cache & mgt module searches the client request in its dynamically updated table.

If the requested content is already multicasted in the IPvY network, the content cache & mgt module diverts the user report back to the proxy publisher module (7). The proxy publisher module adds the new client into its existing multicast tree. Then the requested content can be sent to the client (8).

If the requested content is available but not multicasted in IPvY network yet, the content cache & mgt module sends a request to the proxy receiver module, which locates in the IPvX network (3). It initiates the proxy receiver module to send a join IPvX mutlicast group report (4) to the multicast source. The multicast source adds the multicast proxy into its multicast tree and sends IPvX multicast packets (5). When receiving the multicast packets, the proxy receiver module drops all network layer information, such as IP headers, etc., and only reports contents to the content cache & mgt module (6). The content cache & mgt module then diverts contents to the proxy publisher module (7). The proxy publisher module builds up a new multicast tree in the IPvY network, and sends multicast packets to clients (8).

If all the clients, requesting a certain multicast service in the IPvY network, leave the IPvY multicast group, the multicast proxy MAY leave the IPvX multicast group in IPvX network.

Multicast proxies MAY also perform load-balancing, user authentication and other additional functions.

4. Security Considerations

The multicast proxy solution actually separate the IPv4 and IPv6 multicast services effectively. It prevents the attacks at only one side of it.

However, multicast proxy itself is as vulnerable as normal multicast sources and multicast leafs in each IPv4 or IPv6 environment. The security mechanisms for IGMP/MLD can be used to enhance the security of multicast proxy.

5. IANA Considerations

This draft does not request any IANA action.

6. Acknowledgments

The authors would like to thank Stig Venaas, Cisco for his valuable comments.

7. Change Log [RFC Editor please remove]

draft-jiang-behave-v4v6mc-proxy-00, original version, 2010-03-01

draft-jiang-behave-v4v6mc-proxy-01, private discussion and comments from Stig Venaas, 2010-07-09

8. References

8.1. Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3810] R. Vida and L. Costa, "Multicast Listener Discovery 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC4601] B. Fenner, M. Handley, H. Holbrook and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Specification (Revised)", RFC 4601, August 2006.

8.2. Informative References

- [I-D.draft-venaas-behave-v4v6mc-framework]
S. Venaas, X. Li and C. Bao, "Framework for IPv4/IPv6 Multicast Translation ", draft-venaas-behave-v4v6mc-framework, working in progress, October 2009.

Author's Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Huawei Building, No.3 Xinxu Rd.,
Shang-Di Information Industry Base, Hai-Dian District, Beijing 100085
P.R. China
Email: shengjiang@huawei.com

Dujuan Gu
Huawei Technologies Co., Ltd
156 Bei-Qing Road, Hai-Dian District, Beijing 100085
P.R. China
Email: gudujuan@huawei.com

Behave Work Group
Internet Draft
Intended status: Standard Track
Expires: July 10, 2011

S. Jiang
D. Gu
Huawei Technologies Co., Ltd
January 07, 2011

Multicast Proxy in IPv6/IPv4 Transition
draft-jiang-behave-v4v6mc-proxy-02.txt

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 July 10, 2011.

Copyright Notice

Copyright (c) 2011 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.

Abstract

During the long co-existing period of IPv6 and IPv4, the interoperation between IPv6 network and IPv4 network is essential. Multicast services across IPv6 and IPv4 networks are also needed. Besides the packet-based multicast translation mechanism, this document describes a multicast proxy solution. The multicast proxy is deployed at the border of IPv6/IPv4 networks. It is mainly based on content cache concept. Without packet-based translation, it retires the content data from IPvX network, caches the data, and multicasts the data in IPvY network. It acts as a multicast leaf in the IPvX network where the data source locates. It also acts as a multicast source in IPvY network where the multicast client locates.

Table of Contents

1. Introduction.....	3
2. Terminology.....	3
3. Multicast Proxy without packet-based IPv6/IPv4 Translation...	3
3.1. Overview.....	3
3.2. Operation procedure.....	5
4. Security Considerations.....	6
5. IANA Considerations.....	7
6. Change Log [RFC Editor please remove].....	7
7. References.....	7
7.1. Normative References.....	7
7.2. Informative References.....	7
Author's Addresses.....	8

1. Introduction

The confirmation of IPv4 address exhaustion clearly indicates that global IPv6 deployment is inevitably going to happen. However, it is also widely agreed that IPv4 will be still in use for a long period. During the long co-existing period of IPv6 and IPv4, the interoperation between IPv6 network and IPv4 network is essential.

Now, multimedia has been deployed widely, such as IPTV and video conference etc. They also face the IPv6 and IPv4 intercommunication issues. The multicast applications are complicated and face more difficulties than unicast applications deployment.

[I-D.draft-venaas-behave-v4v6mc-framework] proposes a packet-based translation framework between IPv4/IPv6 multicast services. It describes the packet-based translation operations and intercommunication in network layer to support a single source send to multiple receivers in different IP networks.

Besides the packet-based multicast translation mechanism, this document describes a multicast proxy solution, which is mainly based on content cache.

A multicast proxy can be deployed at the border between IPv4/IPv6 networks. Without packet-based translation, it retrieves the content data from IPvX network, caches the data, and multicasts the data in IPvY network. It acts as a multicast leaf in the IPvX network where the data source locates. It also acts as a multicast source in IPvY network where the multicast client locates.

2. Terminology

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

3. Multicast Proxy without packet-based IPv6/IPv4 Translation

3.1. Overview

Within this document, we describe the network where the data source locates as IPvX network and the network where the multicast client locates as IPvY network. When IPvX is IPv6, IPvY must be IPv4, vice versa.

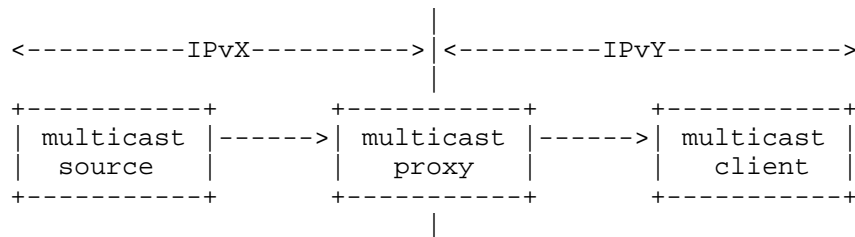


Figure 1: Multicast proxy Forward Contents to different IP networks

As showed in Figure 1, the proposed multicast proxy is deployed at the border of IPv4/IPv6 networks. It MUST support both IPv4 and IPv6. It MUST support both IGMP (Internet Group Management Protocol, [RFC3376]), which is used for IPv4 multicast management functions, and MLD (Multicast Listener Discovery, [RFC3810]), which is used in a similar way in IPv6 Environment. In the IPvX network, the multicast proxy joins the multicast distribution tree as a leaf. In the IPvY network, the multicast proxy broadcasts contents as a multicast source. The establishment of multicast distribution trees obeys the current multicast specifications for each IP family, such as Protocol Independent Multicast (PIM [RFC4601]).

Notice that there are two different multicast distribution trees in two sides of the multicast proxy. They are operational independent from each other in the network layer.

Logically, they are relevant to each other and there are interoperation behaviors between them. The contents published through the multicast distribution tree in IPvY network inherits from the IPvX network. They are received by the multicast proxy, which is a multicast leaf in the multicast distribution tree in IPvX network. Within the multicast proxy, contents are mapped between receiver function and publisher function. The operations of the multicast distribution tree in IPvY network MAY trigger some operations of the multicast distribution tree in IPvX network. For example, a multicast client joins a multicast group in IPvY network, and requests multicast contents may cause the multicast proxy joins a multicast group in IPvX network.

However, as mentioned earlier, in network or IP layer, the two multicast distribution trees are independent from each other. Their operations are separated in two sides of the multicast proxy. Conceptually, the multicast proxy can be presented virtually in functional modules like below Figure 2.

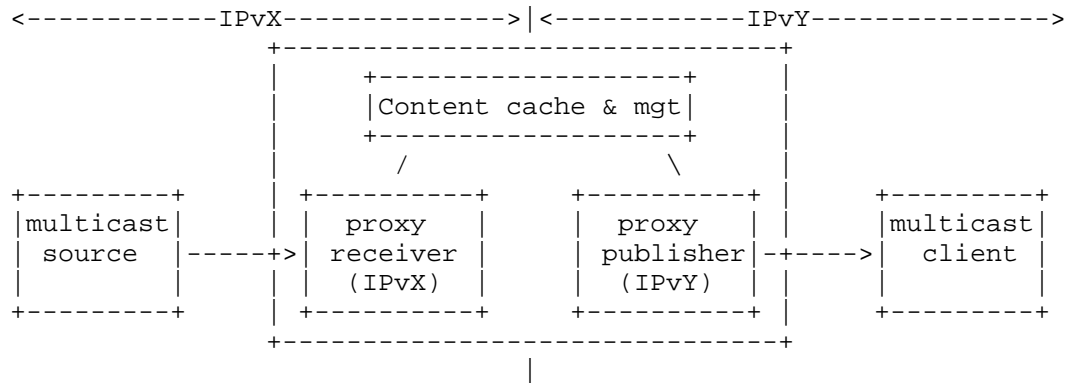


Figure 2: Separate function model of multicast proxy

As shown in Figure 2, the proxy receiver module in IPvX network joins IPvX multicast groups as a receiver client. Thereby it receives packets bound for the IPvX multicast groups, and then hands the content to the content cache and management module. The content cache and management module then forward the on-demand content to the multicast proxy function module in IPvY network, which acts as a publisher and multicast source in IPvY network.

3.2. Operation procedure

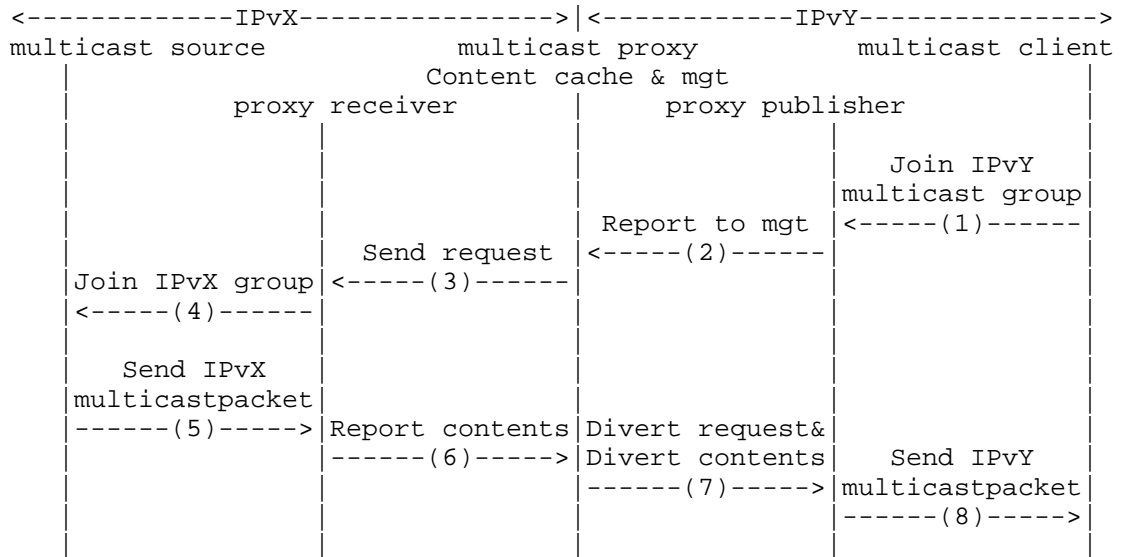


Figure. 3 The interaction communicating from IPvX to IPvY

As shown in Figure 3, a client, which locates in IPvY network, connects to the multicast proxy, requesting a multicast service whose source locates in the IPvX network.

First, the client sends a join IPvY multicast group report (1) to the multicast proxy. The proxy publisher module, which also locates in the IPvY network, receives this report, then forwards the content request to the content cache & management module (2). The content cache & mgt module maintains a content & multicast service table, including all available multicast services from IPvX network. The content cache & mgt module searches the client request in its dynamically updated table.

If the requested content is already multicasted in the IPvY network, the content cache & mgt module diverts the user report back to the proxy publisher module (7). The proxy publisher module adds the new client into its existing multicast tree. Then the requested content can be sent to the client (8).

If the requested content is available but not multicasted in IPvY network yet, the content cache & mgt module sends a request to the proxy receiver module, which locates in the IPvX network (3). It initiates the proxy receiver module to send a join IPvX multicast group report (4) to the multicast source. The multicast source adds the multicast proxy into its multicast tree and sends IPvX multicast packets (5). When receiving the multicast packets, the proxy receiver module drops all network layer information, such as IP headers, etc., and only reports contents to the content cache & mgt module (6). The content cache & mgt module then diverts contents to the proxy publisher module (7). The proxy publisher module builds up a new multicast tree in the IPvY network, and sends multicast packets to clients (8).

If all the clients, requesting a certain multicast service in the IPvY network, leave the IPvY multicast group, the multicast proxy MAY leave the IPvX multicast group in IPvX network.

Multicast proxies MAY also perform load-balancing, user authentication and other additional functions.

4. Security Considerations

The multicast proxy solution actually separate the IPv4 and IPv6 multicast services effectively. It prevents the attacks at only one side of it.

However, multicast proxy itself is as vulnerable as normal multicast sources and multicast leafs in each IPv4 or IPv6 environment. The security mechanisms for IGMP/MLD can be used to enhance the security of multicast proxy.

5. IANA Considerations

This draft does not request any IANA action.

6. Acknowledgments

The authors would like to thank Stig Venaas, Cisco for his valuable comments.

7. Change Log [RFC Editor please remove]

draft-jiang-behave-v4v6mc-proxy-00, original version, 2010-03-01

draft-jiang-behave-v4v6mc-proxy-01, private discussion and comments from Stig Venaas, 2010-07-09

draft-jiang-behave-v4v6mc-proxy-02, regular update, 2011-01-07

8. References

8.1. Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3810] R. Vida and L. Costa, "Multicast Listener Discovery 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC4601] B. Fenner, M. Handley, H. Holbrook and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Specification (Revised)", RFC 4601, August 2006.

8.2. Informative References

- [I-D.draft-venaas-behave-v4v6mc-framework]
S. Venaas, X. Li and C. Bao, "Framework for IPv4/IPv6 Multicast Translation ", draft-venaas-behave-v4v6mc-framework, working in progress, October 2009.

Author's Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Huawei Building, No.3 Xinxu Rd.,
Shang-Di Information Industry Base, Hai-Dian District, Beijing 100085
P.R. China
Email: shengjiang@huawei.com

Dujuan Gu
Huawei Technologies Co., Ltd
156 Bei-Qing Road, Hai-Dian District, Beijing 100085
P.R. China
Email: gudujuan@huawei.com

Behavior Engineering for Hindrance
Avoidance (BEHAVE)
Internet-Draft
Intended status: Informational
Expires: April 20, 2011

J. Korhonen, Ed.
Nokia Siemens Networks
T. Savolainen, Ed.
Nokia
October 17, 2010

Analysis of solution proposals for hosts to learn NAT64 prefix
draft-korhonen-behave-nat64-learn-analysis-00.txt

Abstract

Hosts and applications may benefit from the knowledge if an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. This document analyses number of proposed solutions for communicating if the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. This enables both NAT64 avoidance and intentional utilization by allowing local IPv6 address synthesis.

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 20, 2011.

Copyright Notice

Copyright (c) 2010 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	4
2. Terminology	4
3. Background	5
4. Proposed solutions to learn about synthesis and Network-Specific Prefix	6
4.1. EDNS0 option indicating AAAA Record synthesis and format	6
4.1.1. Solution description	6
4.1.2. Analysis and discussion	7
4.1.3. Summary	7
4.2. EDNS0 flags indicating AAAA Record synthesis and format	8
4.2.1. Solution description	8
4.2.2. Analysis and discussion	8
4.2.3. Summary	9
4.3. Heuristic discovery of NAT64 and Network-Specific Prefix	9
4.3.1. Solution description	9
4.3.2. Analysis and discussion	9
4.3.3. Summary	10
4.4. DNS Resource Record for IPv4-Embedded IPv6 address	10
4.4.1. Solution description	10
4.4.2. Analysis and discussion	10
4.4.3. Summary	11
4.5. Learning the IPv6 Prefix of a Network's NAT64 using DNS	11
4.5.1. Solution description	11
4.5.2. Analysis and discussion	12
4.5.3. Summary	12
4.6. Learning the IPv6 Prefix of a Network's NAT64 using DHCPv6	13
4.6.1. Solution description	13
4.6.2. Analysis and discussion	13
4.6.3. Summary	14
4.7. Learning the IPv6 Prefixes of an IPv6/IPv4 Translator (RA)	14
4.7.1. Solution description	14
4.7.2. Analysis and discussion	14
4.7.3. Summary	15
4.8. Using application layer protocols such as STUN	15
4.8.1. Solution description	15
4.8.2. Analysis and discussion	16
4.8.3. Summary	17

4.9. Hybrid Type Prefix	17
4.9.1. Solution description	17
4.9.2. Analysis and discussion	17
4.9.3. Summary	18
4.10. Provisioning of NAT64 Prefix and the format type	18
4.10.1. Solution description	18
4.10.2. Analysis and discussion	18
4.10.3. Summary	19
5. Conclusion	19
6. Security Considerations	20
7. IANA Considerations	20
8. Contributors	20
9. Acknowledgements	20
10. Informative References	20
Authors' Addresses	23

1. Introduction

Hosts and applications may benefit from the knowledge if an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. There are two issues that can be addressed with solutions that allow hosts and applications to learn the Network Specific Prefix (NSP) [I-D.ietf-behave-address-format] used by the NAT64 [I-D.ietf-behave-v6v4-xlate-stateful] and the DNS64 [I-D.ietf-behave-dns64] devices.

Firstly, finding out whether a particular address is synthetic and therefore learn the presence of a NAT64 in the network. For example, a Dual-Stack (DS) host with IPv4 connectivity could use this information to bypass NAT64 and use native IPv4 transport for destinations that are reachable through IPv4. We will refer this as 'Issue #1' throughout the document.

Secondly, finding out how to construct from an IPv4 address an IPv6 address that will be routable to/by the NAT64. This is useful when IPv4 literals can be found in the payload of some used protocol or applications do not use DNS to resolve names to addresses but know the IPv4 address of the destination by some other means. We will refer this as 'Issue #2' throughout the document.

This document analyses all known solution proposals known at the time of writing for communicating if the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. Based on the analysis we conclude whether the issue of learning the NSP is worth solving and what would be the recommended solution(s) in that case.

2. Terminology

NSP

Network Specific Prefix: A prefix chosen by network administrator for NAT64/DNS64 to present IPv4 addresses in IPv6 namespace.

WKP

Well-Known Prefix: A prefix (64:ff9b::/96) chosen by IETF and configured by network administrator for NAT64/DNS64 to present IPv4 addresses in IPv6 namespace.

NAT64

Network Address and protocol Translation from IPv6 clients to IPv4 servers: A network entity that a host or an application may want to either avoid or utilize. IPv6 packets hosts sends to NSP and/or WKP prefix are routed to NAT64.

DNS64

DNS extensions for network address translation from IPv6 clients to IPv4 servers: A network entity that synthesizes IPv6 addresses and AAAA records out of IPv4 addresses and A records, hence making IPv4 namespaces visible into IPv6 namespace. DNS64 uses NSP and/or WKP in the synthesis process.

Address Synthesis

A mechanism, in the context of this document, where an IPv4 address is represented as an IPv6 address understood by a NAT64 device. The synthesized IPv6 address is formed by embedding an IPv4 address into an IPv6 address prefixed with a NSP/WKP.

Issue #1

The problem of distinguishing between a synthesized and a real IPv6 addresses, which allows a host to learn the presence of a NAT64 in the network.

Issue #2

The problem of learning the NSP used by the access network and needed for local IPv6 address synthesis.

3. Background

Certain applications, operating in protocol translation scenarios, can benefit from knowing the IPv6 prefix used by NAT64. This applies to the Framework document [I-D.ietf-behave-v6v4-framework] Scenario 1 ("IPv6 network to IPv4 Internet") and Scenario 5 ("An IPv6 network to an IPv4 network"). In those scenarios it is important for applications to learn that the network has a NAT64 deployed and what the is prefix used by the NAT64. The NAT64 prefix can be either a Network Specific Prefix (NSP) or the Well-known Prefix (WKP). Below is (an incomplete) list of various use cases where it is beneficial for a host or an application to know the presence of a NAT64 and the NSP/WKP:

- o Host-based DNSSEC validation.
- o Protocols that use IPv4 literals.
- o Multicast translations
[I-D.venaas-behave-mcast46][I-D.venaas-behave-v4v6mc-framework].
- o URI schemes with host IPv4 address literals rather than domain names (e.g., `http://192.0.2.1`, `ftp://192.0.2.1`, `imap://192.0.2.1`, `ipp://192.0.2.1`)).
- o Updating host's [RFC3484] preference table to prefer native prefixes over translated prefixes.
- o Allow a host to perform its own DNS64 function. This allows the host to provide translation functions to IPv4 applications using, for example, Bump In the Host [I-D.huang-behave-bih].
- o Allows a host to bypass NAT64 when native IPv4 connectivity is available and preferred over protocol translation.

DNS64 cannot serve applications that are not using DNS or that obtain referral as an IPv4 literal address. One example application is the Session Description Protocol (SDP) [RFC4566], as used by Real Time Streaming Protocol (RTSP) [RFC2326] and Session Initiation Protocol (SIP) [RFC3261]. Other example applications include web browsers, as IPv4 address literals are still encountered in web pages and URLs. Some of these applications could still work through NAT64, provided they were able to create locally valid IPv6 presentations of peers' IPv4 addresses.

[Editor's note: Should we discuss referral objects in general??]

4. Proposed solutions to learn about synthesis and Network-Specific Prefix

4.1. EDNS0 option indicating AAAA Record synthesis and format

4.1.1. Solution description

Section 3 of [I-D.korhonen-edns0-synthesis-flag] defines a new EDNS0 option [RFC2671], which contain 3 flag bits (called SY-bits). The EDNS0 option serves as an implicit indication of the presence of DNS64 server and the NAT64 device. The EDNS0 option SY-bit values other than '000' and '111' explicitly tell the NSP prefix length. Only the DNS64 server can insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA Resource Record.

4.1.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and is designed to explicitly solve Issue #2.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + Even if the solution is bundled with DNS queries and responses, a standardization of a new DNS record type is not required, rather just defining a new EDNS0 option.
- + EDNS0 option implementation requires changes only to DNS64 servers.
- + Does not require additional provisioning or management as the EDNS0 option is added automatically by the DNS64 server to the responses.
- + Does not involve additional queries towards the global DNS infrastructure as EDNS0 logic can be handled within the DNS64 server.

The CONS of the proposal are listed below:

- Requires end hosts to support [RFC2671] EDNS0 extension mechanism.
- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length.
- Requires a modification to DNS64 servers to include the EDNS0 option to the synthesized responses.
- Does not help to identify a synthesized IPv6 address if the session does not involve any DNS queries.

4.1.3. Summary

The EDNS0 option based solution avoids the trouble of defining and standardizing a new DNS Resource Record type by extending the existing EDNS0 Resource Record. Although the solution has host resolver and DNS64 server impacts, the changes are limited to those entities (end host, applications) that are interested in learning the presence of NAT64 and the used NAT64 prefix. The provisioning and

management overhead is minimal if not non-existent as the EDNS0 options are synthesized in a DNS64 server in a same manner as the synthesized AAAA Resource Records. Moreover, EDNS0 does not induce any load to DNS servers because no new RRTYPE query is defined.

4.2. EDNS0 flags indicating AAAA Record synthesis and format

4.2.1. Solution description

Section 3 of [EDNS0-Flag] defines 3 new flag bits (called SY-bits) into EDNS0 OPT [RFC2671] header, which serve as an implicit indication of the presence of DNS64 server and a NAT64 device. SY-bit values other than '000' or '111' explicitly tell the NSP prefix length. Only the DNS64 server can insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA Resource Record.

4.2.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and is designed to explicitly solve Issue #2.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + EDNS0 option implementation requires changes only to DNS64 servers.
- + Does not require additional provisioning or management as the EDNS0 option is added automatically by the DNS64 server to the responses.
- + Does not require introduction of new EDNS0 option
- + Does not involve additional queries towards the global DNS infrastructure as EDNS0 logic can be handled within the DNS64 server.

The CONS of the proposal are listed below:

- Requires end hosts to support [RFC2671] EDSN0 extension mechanism.
- Consumes scarce flag bits from EDNS0 OPT header.

- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length.
- Requires a modification to DNS64 servers to include the EDNS0 option to the synthesized responses.

4.2.3. Summary

This option is included here for the sake of completeness. The consumption of three bits of the limited EDNS0 OPT space can be considered unfavorable and hence is unlikely to be accepted.

4.3. Heuristic discovery of NAT64 and Network-Specific Prefix

4.3.1. Solution description

Section 3 of [I-D.savolainen-heuristic-nat64-discovery] describes a host behavior for discovering the presence of a DNS64 server and a NAT64 device, and heuristics for discovering the used NSP. A host requiring information for local IPv6 address synthesis or for NAT64 avoidance sends a DNS query for an AAAA record of a (well-)known IPv4-only Fully Qualified Domain Name (FQDN). If a host receives a negative reply, it knows there are no DNS64 and NAT64 in the network.

If a host receives AAAA reply, it knows the network must be utilizing IPv6 address synthesis. After receiving a synthesized AAAA Resource Record, the host may examine the received IPv6 address and use heuristics, such as "subtracting" the known IPv4 address out of synthesized IPv6 address, do find out the NSP.

4.3.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Does not necessarily require any standards effort.
- + Does not require host stack or resolver changes. All required logic and heuristics can be implemented in applications that are interested in learning about address synthesis taking place.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.

- + Hosts or applications interested in learning about synthesis and the used NSP can do the "discovery" proactively at any time, for example every time the host attaches to a new network.
- + Does not require explicit support from the network using NAT64

The CONS of the proposal are listed below:

- A standardization of a domain name for the heuristics purposes would be beneficial (i.e. at least an IANA actions would be needed), but not necessary.
- Any heuristic is less reliable than explicit methods

4.3.3. Summary

This is the only approach that can be deployed without explicit support from the network or the host. This approach could also complement explicit methods and be used as a fallback approach when explicit methods are not supported by an access network.

4.4. DNS Resource Record for IPv4-Embedded IPv6 address

4.4.1. Solution description

Section 4 of [I-D.boucadair-behave-dns-a64] defines a new DNS Resource Record (A64) that is a record specific to store a single IPv4-Embedded IPv6 address [I-D.ietf-behave-address-format]. Using a dedicated Resource Record allows a host distinguish between real IPv6 addresses and synthesized IPv6 addresses. The solution notifies to the requesting host that the resolved address is not a native address but an IPv4-Embedded IPv6 address. This would ease the local policies to prefer direct communications (i.e., avoid using IPv4-Embedded IPv6 addresses when a native IPv6 address or a native IPv4 address is available).

4.4.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.

- + Synthesized addresses can be used in authoritative DNS servers.
- + Maintains the reliability of the DNS model (i.e., a synthesised IPv6 address is presented as such and not as native IPv6 address).
- + When both IPv4-Converted and native IPv6 addresses are configured for the same QNAME, native addresses are preferred.

The CONS of the proposal are listed below:

- Does not address Issue #2 in any way.
- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized.
- Requires a standardization of a new DNS Resource Record type (A64), and the implementation of it in both resolvers and servers.
- Requires a coordinated deployment between different flavors of DNS servers within the provider to work deterministically.
- Additional load the DNS servers (3 Queries, A64, AAAA and A, may be issued by a dual-stack host).
- Does not help to identify synthesized IPv6 addresses if the session does not involve any DNS queries.

4.4.3. Summary

While the proposed solution delivers explicit information about address synthesis taking place solving the Issue #1, a standardization of a new DNS record type might turn out a too overwhelming task for a solution for a temporary transition phase. Defining a new record type increases load towards DNS server as the host issues parallel A64, AAAA and A queries.

4.5. Learning the IPv6 Prefix of a Network's NAT64 using DNS

4.5.1. Solution description

Section 4.1 of [I-D.wing-behave-learn-prefix] actually proposes two DNS-based method for discovering the presence of a DNS64 server and a NAT64 device, and then a mechanism for discovering the used NSP. First, a host may learn the presence of a DNS64 server and a NAT64 device, by receiving a TXT Resource Record with a well-known (TBD IANA registered?) string followed by the NAT64 unicast IPv6 address and the prefix length. The DNS64 server would add the TXT Resource

Record into the DNS response.

Second, Section 4.1 of [I-D.wing-behave-learn-prefix] also proposes specifying a new U-NAPTR [RFC4848] application to discover the NAT64's IPv6 prefix and length. The input domain name is exactly the same as would be used for a reverse DNS lookup, derived from the host's IPv6 in the ".ip6.arpa." tree. The host doing the U-NAPTR queries may need multiple queries until finds the provisioned domain name with the correct prefix length. The response to a successful U-NAPTR query contains the unicast IPv6 address and the prefix length of the NAT64 device.

4.5.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Does not require host stack or resolver changes if the required logic and heuristics is implemented in applications that are interested in learning about address synthesis taking place.

The CONS of the proposal are listed below:

- Requires standardization of a well-known names from IANA for TXT Resource Record and/or a standardization of a new U-NAPTR application.
- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length. However, it is possible that the U-NAPTR application logic is completely implemented by the application itself as noted in PROs list.
- U-NAPTR prefix learning method may entail multiple queries.
- U-NAPTR prefix learning method requires provisioning of NSPs in ".ip6.arpa." tree.

4.5.3. Summary

The implementation of this solution requires some changes to the applications and resolvers in a similar fashion as in solutions in Section 4.1, Section 4.2 and Section 4.4. Unlike the other DNS-based approaches, the U-NAPTR-based solution also requires provisioning information into the '.ip6.arpa.' tree which is not anymore entirely internal to the provider hosting the NAT64/DNS64 service.

The iterative approach of learning the NAT64 prefix in U-NAPTR-based solution may result in multiple DNS queries, which can be considered more complex and inefficient compared to other DNS-based solutions.

4.6. Learning the IPv6 Prefix of a Network's NAT64 using DHCPv6

4.6.1. Solution description

Section 4.2 of [I-D.wing-behave-learn-prefix] describes a new DHCPv6 [RFC3315] option (OPTION_AFT_PREFIX_DHCP) that contains the IPv6 unicast prefix, IPv6 ASM prefix, and IPv6 SSM prefix (and their lengths) for the NAT64.

Section 5 of [RA-Learn-Prefix] describes one solution how to authenticate a learned NAT64 prefix using DNSSEC infrastructure [RFC4033]. Use of DNSSEC would have an undesirable overhead impact and would also require additional provisioning on DNS side. Because the prefix authentication "requirement" is not unique to the NAT64 learning solution described in [I-D.wing-behave-learn-prefix], it has been lifted out of CONs and just described here.

4.6.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Does not involve DNS system. Therefore, applications that would not normally initiate any DNS queries can still learn the NAT64 prefix.
- + DHCPv6 is designed to provide various kinds of configuration information in a centrally managed fashion.

The CONs of the proposal are listed below:

- Change of NSP requires change to DHCPv6 configuration.
- Requires at least Stateless DHCPv6 client on hosts.
- Requires support on DHCPv6 clients, which is not trivial in all operating systems.
- Requires the network advertising the DHCP option cooperate with the network operating the NAT64. This prevents a NAT64 from being operated by a different network (e.g., as a translation service).

- The DHCPv6-based solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

4.6.3. Summary

The DHCPv6-based solution would be a good solution in a sense it hooks into general IP configuration phase, allows easy updates when configuration information changes and does not involve DNS in general. However, use of DHCPv6 would require changes on both end host and network side DHCPv6 implementations. Furthermore, it is not obvious that all devices that need translation services would implement DHCPv6. For example, cellular 3GPP networks do not mandate hosts or network to implement or deploy DHCPv6, and furthermore DHCPv6 is not even supported to configure end host IPv6 address [I-D.korhonen-v6ops-3gpp-eps].

4.7. Learning the IPv6 Prefixes of an IPv6/IPv4 Translator (RA)

4.7.1. Solution description

Section 3.3 of [RA-Learn-Prefix] describes a new Router Advertisement (RA) [RFC4861] option (OPTION_AFT_PREFIX_RA) that contains the IPv6 unicast prefix, IPv6 ASM prefix, and IPv6 SSM prefix (and their lengths) for the NAT64. The RA option is essentially the same as for DHCPv6 discussed in Section 4.6.

4.7.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Does not involve DNS system

The CONS of the proposal are listed below:

- Requires configuration and managements of all access routers to emit correct information in RA. This could, for example, be accomplished somehow piggybacked on top of routing protocols (which would then require enhancements to routing protocols)
- In some operating systems it may not be trivial to transfer information obtained in RA to upper layers

- Requires changes to host operating system's IP stack
- NSP change requires changes to access router configuration
- Requires standardization of a new option to Router Advertisement that is generally unfavored approach
- The RA-based solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

4.7.3. Summary

The RA-based solution would be a good solution in a sense it hooks into general IP configuration phase, allows easy updates when configuration information changes and does not involve DNS in general. However, generally introducing any changes to the Neighbor Discovery Protocol that are not absolutely necessary are unfavored due the impact on both network node side and end host IP stack implementations.

Compared to the DHCPv6 equivalent solution in Section 4.6 the management overhead is greater with RA-based solution. In case of DHCPv6-based solution the management can be centralized to few DHCPv6 servers compared to RA-based solution where each access router is supposed to be configured with the same information.

4.8. Using application layer protocols such as STUN

4.8.1. Solution description

Application layer protocols, such as Session Traversal Utilities for NAT (STUN) [RFC5389], which define methods for endpoints to learn their external IP addresses could be used for NAT64 and NSP discovery. This document focuses on STUN, but the protocol could be something else as well.

A host must first use DNS to discover IPv6 representation(s) of STUN server(s) IPv4 address(es), because the host has no way to directly use IPv4 addresses to contact to STUN server(s).

After learning the IPv6 address of a STUN server the STUN client sends a request to the STUN server containing new 'SENDING-TO' attribute that tells to the server the IPv6 address the client sent the request to. In a reply the server includes another new attribute called 'RECEIVED-AS', which contains server's IP address the request arrived on. After receiving the reply the client compares 'SENDING-TO' and 'RECEIVED-AS' attributes using heuristics similar to

what is described in section 4.3 and finds out NSP candidate.

4.8.2. Analysis and discussion

This solution is relatively similar as described in section 4.3, but instead of using DNS uses STUN to get input for heuristic algorithms.

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Does not require host stack or resolver changes. All required logic and heuristics can be implemented in applications that are interested in learning about address synthesis taking place.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + Hosts or applications interested in learning about synthesis and the used NSP can do the "discovery" proactively at any time, for example every time the host attaches to a new network.
- + Does not require explicit support from the network using NAT64.
- + Does not involve DNS system.
- + Can possibly be bundled to existing STUN message exchanges as new attributes and hence client can learn its external IPv4 address and NSP/WKP with the same exchange
- + Can be used to confirm the heuristics by synthesizing IPv6 address of another STUN server or by synthesizing IPv6 address of first STUN server after host has heuristically determined NSP using method from section 4.3. I.e. the connectivity test could be done with STUN.
- + True IPv4 destination address is used in NSP determination instead of IPv4 address received from DNS. This may increase reliability.
- + The same STUN improvement could also be used to reveal NAT66 on the data path, if the 'RECEIVED-AS' would contain different IPv6 address than 'SENDING-TO'.

The CONS of the proposal are listed below:

- Requires a server on the network to respond the queries.
- Any heuristic is less reliable than explicit methods.
- Requires standardization if done as extension to STUN.
- The solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

4.8.3. Summary

The STUN, or similar, protocol based approach is a second way to solve the problem without explicit access network support. The heuristics for NSP discovery would still be in the client, however, the result may be more reliable as actual IPv4 destination address is compared to IPv6 address used in sending. The additional benefit of STUN is that the client learns its public IPv4 address with the same message exchange. STUN could also be used as the connectivity test tool if the client would first heuristically determine NSP out of DNS as described in section 4.3, synthesize IPv6 representation of STUN server's IPv4 address, and then tests connectivity to the STUN server.

As an additional benefit the STUN improvement could be used for NAT66 discovery.

4.9. Hybrid Type Prefix

4.9.1. Solution description

[I-D.xu-behave-hybrid-type-prefix] proposes a solution to unambiguously distinguish IPv4-Embedded IPv6 addresses from native IPv6 ones. Doing so would guide the address selection and therefore allow avoiding crossing unnecessary address family translators. [I-D.xu-behave-hybrid-type-prefix] proposes a format called Hybrid Type Prefix (64::/16). The proposed format is primary suitable for scenarios where the IPv4-IPv6 interconnection service is not restricted to customers attached to the same domain (e.g., AS) where the translator is located. Hybrid Type Prefix should be used only for IPv4-Converted IPv6 addresses but never for IPv4-Translatable IPv6 ones. The WKP prefixes in the Behave format belong to the prefix proposed in [I-D.xu-behave-hybrid-type-prefix].

4.9.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Addresses both Issue #1 and Issue #2.
- + Load-balancing can be achieved using a WKP prefix (64:ff9b::/80).
- + Even if the NAT64 is not attached to the same domain (autonomous system) as the dual-stack host, NAT64 can be avoided.
- + Unlike DNS-based solutions (EDNS0 and A64 RR), whatever the scheme used to retrieve the destination, a client is able to assess whether its nature (IPv4-Converted IPv6 address or not).

The CONS of the proposal are listed below:

- The solution requires a new registry to be maintained by IANA.

4.9.3. Summary

This solution mitigates both Issue #1 and Issue #2 but it requires a new IPv6 address registry to be maintained by IANA. Two IPv6 prefix blocks need to be maintained by registries. To ensure global reachability in Internet, an additional announcement (HTP) is required to be advertised in the inter-domain routing.

4.10. Provisioning of NAT64 Prefix and the format type

4.10.1. Solution description

Section 4 of [I-D.boucadair-dhcpv6-shared-address-option] defines a DHCPv6 option that can be used to communicate to a requesting host the prefix used for building IPv4-Converted IPv6 addresses together with the format type. Provisioning the format type is required so as to be correctly handled by the NAT64-enabled devices deployed in a given domain.

4.10.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Addresses both Issue #1 and Issue #2.
- + Unlike DNS-based solution, an address can be assessed whether it is synthesised or not.

The CONS of the proposal are listed below:

- A new DHCPv6 option is required and the corresponding changes to both DHCPv6 clients and servers.
- If the IPv4-Converted IPv6 address, received for instance in referrals, is managed by a NAT64 managed by another service provider, IPv4-Converted IPv6 address can be preferred over the native IPv6 address.

4.10.3. Summary

This solution allows a given IPv6 host to assess whether an IPv6 address is IPv6-Converted or not only if the NAT64 is managed by the same service provider as the one offering the connectivity service to the IPv6 host. The solution requires a new DHCPv6 option code to be assigned. This solution can be considered as a complementary solution to DNS-based ones.

5. Conclusion

As a general principle we prefer to have as minimal solution as possible, avoid impacts to entities not otherwise involved in the protocol translation scheme, minimize host impact, and that requires minimal to no operational effort on the network side.

The DNS64 entity has to be configured with WKP/NSP in order for it to do synthetisation and hence using DNS also for delivering the synthetisation information sounds most logical. The fact that the synthetisation information fate-shares the information received in the DNS response is a valuable attribute and reduces the possible distribution of stale prefix information. On the contrary, use of DHCPv6 would require additional trouble configuring DHCPv6 servers and ensuring DHCPv6 clients are in place, and furthermore that the NAT64, DHCPv6 (and possible even some DNS64) servers are all in sync. RA-based mechanisms are operationally expensive as configuration would have to be placed and maintained in the access routers. Furthermore, both DHCPv6 and RA based mechanisms involve entities that do not otherwise need to be aware of protocol translation.

Of the DNS-based mechanisms we favor EDNS0 option due to its lightweight nature. The A64 and DNS SRV record approaches would require significant amount of standardization and deployment effort when compared to the size of the problem. The U-NAPTR-based approach would require provisioning information into the '.ip6.arpa' tree which would not be entirely internal for the provider.

The two heuristic-based approaches could be taken into use at once and would provide benefits in networks utilizing protocol

translation, but on the long term their usefulness depends how well networks will deploy explicit methods.

Our conclusion is to recommend standardization of EDNS0 option and combining the two heuristic-based methods into one informational document for application and host implementors to implement as is and potentially to be used as basis for expanding proprietary- or IETF-protocols such as STUN.

6. Security Considerations

Forgery of information required for IPv6 address synthesis may allow an attacker to insert itself as middle man or to perform denial-of-service attack. The DHCPv6 and RA based approaches are most vulnerable for the forgery as the attacker may send forged RAs or act as a rogue DHCPv6 server. Hence use of DHCPv6 and RA approaches might introduce a new attack vector towards DNS. However, if the attacker is able to modify and forge DNS responses (flags, addresses of known IPv4-only servers, records, etc), ability to influence local address synthesis is likely of low additional value. Also, a DNS-based mechanism is only as secure as how the DNS server's IP address is configured on the host. Therefore, if the host cannot trust e.g. DHCPv6 it cannot trust the DNS server learned via DHCPv6 either, unless the host has a way to authenticate all DNS responses.

7. IANA Considerations

This document is informative and has no actions to IANA.

8. Contributors

The following people have contributed text to this document.

Mohamed Boucadair

9. Acknowledgements

Authors would like to thank Dan Wing and Christian Huitema especially for the STUN idea for their valuable comments and discussions.

10. Informative References

[EDNS0-Flag]

Korhonen, J. and T. Savolainen, "EDNS0 Flags Indicating AAAA Record Synthesis and Format", draft-korhonen-edns0-synthesis-flag-00 (work in progress), June 2010.

[I-D.boucadair-behave-dns-a64]

Boucadair, M. and E. Burgey, "A64: DNS Resource Record for IPv4-Embedded IPv6 Address", draft-boucadair-behave-dns-a64-02 (work in progress), September 2010.

[I-D.boucadair-dhcpv6-shared-address-option]

Boucadair, M., Levis, P., Grimault, J., Savolainen, T., and G. Bajko, "Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions", draft-boucadair-dhcpv6-shared-address-option-01 (work in progress), December 2009.

[I-D.huang-behave-bih]

Huang, B., Deng, H., and T. Savolainen, "Dual Stack Hosts Using "Bump-in-the-Host" (BIH)", draft-huang-behave-bih-01 (work in progress), July 2010.

[I-D.ietf-behave-address-format]

Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-10 (work in progress), August 2010.

[I-D.ietf-behave-dns64]

Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-11 (work in progress), October 2010.

[I-D.ietf-behave-v6v4-framework]

Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-10 (work in progress), August 2010.

[I-D.ietf-behave-v6v4-xlate-stateful]

Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 2010.

- [I-D.korhonen-edns0-synthesis-flag]
Korhonen, J. and T. Savolainen, "EDNS0 Option for Indicating AAAA Record Synthesis and Format", draft-korhonen-edns0-synthesis-flag-01 (work in progress), September 2010.
- [I-D.korhonen-v6ops-3gpp-eps]
Korhonen, J., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3GPP Evolved Packet System", draft-korhonen-v6ops-3gpp-eps-03 (work in progress), June 2010.
- [I-D.savolainen-heuristic-nat64-discovery]
Savolainen, T. and J. Korhonen, "Heuristic discovery of NAT64 and Network-Specific Prefix", draft-savolainen-heuristic-nat64-discovery-00 (work in progress), September 2010.
- [I-D.venaas-behave-mcast46]
Venaas, S., Asaeda, H., SUZUKI, S., and T. Fujisaki, "An IPv4 - IPv6 multicast translator", draft-venaas-behave-mcast46-01 (work in progress), July 2009.
- [I-D.venaas-behave-v4v6mc-framework]
Venaas, S., Li, X., and C. Bao, "Framework for IPv4/IPv6 Multicast Translation", draft-venaas-behave-v4v6mc-framework-01 (work in progress), October 2009.
- [I-D.wing-behave-learn-prefix]
Wing, D., "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-04 (work in progress), October 2009.
- [I-D.xu-behave-hybrid-type-prefix]
Xu, X. and C. Byrne, "Hybrid Type Prefix for IPv4-Embedded IPv6 Addresses", draft-xu-behave-hybrid-type-prefix-00 (work in progress), January 2010.
- [RA-Learn-Prefix]
Wing, D., Wang, X., and X. Xu, "Learning the IPv6 Prefixes of an IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-03 (work in progress), July 2009.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

- [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [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.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

Authors' Addresses

Jouni Korhonen (editor)
Nokia Siemens Networks
Linnoitustie 6
FI-02600 Espoo
Finland

Email: jouni.nospam@gmail.com

Teemu Savolainen (editor)
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com

Behavior Engineering for Hindrance
Avoidance (BEHAVE)
Internet-Draft
Intended status: Informational
Expires: August 26, 2011

J. Korhonen, Ed.
Nokia Siemens Networks
T. Savolainen, Ed.
Nokia
February 22, 2011

Analysis of solution proposals for hosts to learn NAT64 prefix
draft-korhonen-behave-nat64-learn-analysis-02.txt

Abstract

Hosts and applications may benefit from the knowledge if an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. This document analyses a number of proposed solutions for communicating whether the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. This enables both NAT64 avoidance and intentional utilization by allowing local IPv6 address synthesis.

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 August 26, 2011.

Copyright Notice

Copyright (c) 2011 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	4
2. Terminology and Assumptions	4
3. Background	6
4. Proposed solutions to learn about synthesis and Network-Specific Prefix	7
4.1. EDNS0 option indicating AAAA Record synthesis and format	7
4.1.1. Solution description	7
4.1.2. Analysis and discussion	8
4.1.3. Summary	9
4.2. EDNS0 flags indicating AAAA Record synthesis and format	9
4.2.1. Solution description	9
4.2.2. Analysis and discussion	9
4.2.3. Summary	10
4.3. DNS Query for a Well-Known Name	10
4.3.1. Solution description	10
4.3.2. Analysis and discussion	11
4.3.3. Summary	11
4.4. DNS Resource Record for IPv4-Embedded IPv6 address	11
4.4.1. Solution description	12
4.4.2. Analysis and discussion	12
4.4.3. Summary	13
4.5. Learning the IPv6 Prefix of a Network's NAT64 using DNS	13
4.5.1. Solution description	13
4.5.2. Analysis and discussion	13
4.5.3. Summary	14
4.6. Learning the IPv6 Prefix of a Network's NAT64 using DHCPv6	15
4.6.1. Solution description	15
4.6.2. Analysis and discussion	15
4.6.3. Summary	16
4.7. Learning the IPv6 Prefix of a Network's NAT64 using Router Advertisements	16
4.7.1. Solution description	16
4.7.2. Analysis and discussion	16
4.7.3. Summary	17
4.8. Using application layer protocols such as STUN	17
4.8.1. Solution description	17
4.8.2. Analysis and discussion	18
4.8.3. Summary	19
4.9. Learning the IPv6 Prefix of a Network's NAT64 using	

Access Technology Specific Methods	19
4.9.1. Solution description	20
4.9.2. Analysis and discussion	20
4.9.3. Summary	20
5. Conclusion	21
6. Security Considerations	22
7. IANA Considerations	22
8. Contributors	22
9. Acknowledgements	22
10. Informative References	23
Authors' Addresses	25

1. Introduction

Hosts and applications may benefit from the knowledge of whether an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. There are two issues that can be addressed with solutions that allow hosts and applications to learn the Network Specific Prefix (NSP) [RFC6052] used by the NAT64 [I-D.ietf-behave-v6v4-xlate-stateful] and the DNS64 [I-D.ietf-behave-dns64] devices.

Firstly, finding out whether a particular address is synthetic and therefore learning the presence of a NAT64. For example, a Dual-Stack (DS) host with IPv4 connectivity could use this information to bypass NAT64 and use native IPv4 transport for destinations that are reachable through IPv4. We will refer this as 'Issue #1' throughout the document.

Secondly, finding out how to construct from an IPv4 address an IPv6 address that will be routable to/by the NAT64. This is useful when IPv4 literals can be found in the payload of some protocol or applications do not use DNS to resolve names to addresses but know the IPv4 address of the destination by some other means. We will refer this as 'Issue #2' throughout the document.

Additionally three other issues have to be considered by a solution addressing the first two issues: whether DNS is required 'Issue #3', whether a solution supports changing NSP 'Issue #4', and whether multiple NSPs are supported (either of the same or different length) for load-balancing purposes 'Issue #5'.

This document analyses all known solution proposals known at the time of writing for communicating if the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. Based on the analysis we conclude whether the issue of learning the Network-Specific Prefix is worth solving and what would be the recommended solution(s) in that case.

2. Terminology and Assumptions

NSP

Network-Specific Prefix: A prefix chosen by network administrator for NAT64/DNS64 to present IPv4 addresses in IPv6 namespace.

WKP

Well-Known Prefix: A prefix (64:ff9b::/96) chosen by IETF and configured by a network administrator for NAT64/DNS64 to present IPv4 addresses in IPv6 namespace.

NAT64

Network Address and protocol Translation mechanism for translating IPv6 packets to IPv4 packets and vice-versa: A network entity that a host or an application may want to either avoid or utilize. IPv6 packets hosts send to addresses in the NSP and/or WKP are routed to NAT64.

DNS64

DNS extensions for network address translation from IPv6 clients to IPv4 servers: A network entity that synthesizes IPv6 addresses and AAAA records out of IPv4 addresses and A records, hence making IPv4 namespaces visible into IPv6 namespace. DNS64 uses NSP and/or WKP in the synthesis process.

Address Synthesis

A mechanism, in the context of this document, where an IPv4 address is represented as an IPv6 address understood by a NAT64 device. The synthesized IPv6 address is formed by embedding an IPv4 address as-is into an IPv6 address prefixed with a NSP/WKP. It is assumed that the 'unused' suffix bits of the synthesized address are set to zero as described in Section 2.2 of [RFC6052].

Issue #1

The problem of distinguishing between a synthesized and a real IPv6 addresses, which allows a host to learn the presence of a NAT64 in the network.

Issue #2

The problem of learning the NSP used by the access network and needed for local IPv6 address synthesis.

Issue #3

The problem of learning the NSP or WKP used by the access network by a host not implementing DNS (hence applications are unable to use DNS to learn prefix).

Issue #4

The problem of supporting changing NSP. The NSP learned by the host may become stale for multiple reasons. For example, the host might move to a new network that uses different NSP, thus making the previously learned NSP stale. Also, the NSP used in the network may be changed due administrative reasons, thus again making previously learned NSP stale.

Issue #5

The problem of supporting multiple NSPs. A network may be configured with multiple NSPs for address synthesis. For example, for load-balancing purposes each NAT64 device in the same network could be assigned with their own NSP. It should be noted that learning a single NSP is enough for an end host to successfully perform local IPv6 address synthesis but to avoid NAT64 the end host needs to learn all NSPs used by the access network.

3. Background

Certain applications, operating in protocol translation scenarios, can benefit from knowing the IPv6 prefix used by a local NAT64 of the attached access network. This applies to the Framework document [I-D.ietf-behave-v6v4-framework] Scenario 1 ("IPv6 network to IPv4 Internet"), Scenario 5 ("An IPv6 network to an IPv4 network"), and Scenario 7 ("The IPv6 Internet to the IPv4 Internet"). Scenario 3 ("The IPv6 Internet to an IPv4 network") is not considered applicable herein as in that case a NAT64 is located at the front of remote IPv4 network and host in IPv6 Internet can benefit very little of learning NSP IPv6 prefix used by the remote NAT64. The NAT64 prefix can be either a Network Specific Prefix (NSP) or the Well-known Prefix (WKP). Below is (an incomplete) list of various use cases where it is beneficial for a host or an application to know the presence of a NAT64 and the NSP/WKP:

- o Host-based DNSSEC validation: as is documented in DNS64 [I-D.ietf-behave-dns64] section 5.5. point 3, synthetic AAAA records cannot be successfully validated in a host. In order to utilize NAT64 a security-aware and validating host has to perform DNS64 function locally and hence it has to be able to learn WKP or proper NSP.
- o Protocols that use IPv4 literals: in IPv6-only access native IPv4 connections cannot be created. If a network has NAT64 it is possible to synthesize IPv6 address by combining the IPv4 literal and the IPv6 prefix used by NAT64. The synthesized IPv6 address

can then be used to create an IPv6 connection.

- o Multicast translations
[I-D.venaas-behave-mcast46][I-D.venaas-behave-v4v6mc-framework].
- o URI schemes with host IPv4 address literals rather than domain names (e.g., `http://192.0.2.1`, `ftp://192.0.2.1`, `imap://192.0.2.1`, `ipp://192.0.2.1`): a host can synthesize IPv6 address out of the literal in URI and use IPv6 to create connection through NAT64.
- o Updating host's [RFC3484] preference table to prefer native prefixes over translated prefixes: this is useful as applications are more likely able to traverse through NAT44 than NAT64.

DNS64 cannot serve applications that are not using DNS or that obtain referral as an IPv4 literal address. One example application is the Session Description Protocol (SDP) [RFC4566], as used by Real Time Streaming Protocol (RTSP) [RFC2326] and Session Initiation Protocol (SIP) [RFC3261]. Other example applications include web browsers, as IPv4 address literals are still encountered in web pages and URLs. Some of these applications could still work through NAT64, provided they were able to create locally valid IPv6 presentations of peers' IPv4 addresses.

It is a known issue that passing IP address referrals, often fails in today's Internet [I-D.carpenter-referral-ps]. Synthesizing IPv6 addresses does not necessarily make the situation any better as the synthesized addresses are not distinguishable from public IPv6 addresses for the referral receiver. However, the situation is not really any different from the current Internet as using public addresses does not really guarantee reachability (for example due to firewalls). Therefore, we think that it is up to the referral originating host to somehow identify that the IPv6 address is made-up.

4. Proposed solutions to learn about synthesis and Network-Specific Prefix

4.1. EDNS0 option indicating AAAA Record synthesis and format

4.1.1. Solution description

Section 3 of [I-D.korhonen-edns0-synthesis-flag] defines a new EDNS0 option [RFC2671], which contains 3 flag bits (called SY-bits). The EDNS0 option serves as an implicit indication of the presence of DNS64 server and the NAT64 device. The EDNS0 option SY-bit values other than '000' and '111' explicitly tell the NSP prefix length.

Only the DNS64 server can insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA Resource Record.

4.1.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and is designed to explicitly solve Issue #2.
- + Solves issue #4 via DNS record lifetime.
- + Can partially solve issue #5 if multiple synthetic AAAA records are included in the response and all use same format.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + Even if the solution is bundled with DNS queries and responses, a standardization of a new DNS record type is not required, rather just defining a new EDNS0 option.
- + EDNS0 option implementation requires changes only to DNS64 servers.
- + Does not require additional provisioning or management as the EDNS0 option is added automatically by the DNS64 server to the responses.
- + Does not involve additional queries towards the global DNS infrastructure as EDNS0 logic can be handled within the DNS64 server.

The CONS of the proposal are listed below:

- Requires end hosts to support [RFC2671] EDSN0 extension mechanism.
- Requires host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length.
- Requires a modification to DNS64 servers to include the EDNS0 option to the synthesized responses.

- Does not provide solution for issue #3.

4.1.3. Summary

The EDNS0 option based solution works by extending the existing EDNS0 Resource Record. Although the solution has host resolver and DNS64 server impacts, the changes are limited to those entities (end host, applications) that are interested in learning the presence of NAT64 and the used NAT64 prefix. The provisioning and management overhead is minimal if not non-existent as the EDNS0 options are synthesized in a DNS64 server in a same manner as the synthesized AAAA Resource Records. Moreover, EDNS0 does not induce any load to DNS servers because no new RRType query is defined.

4.2. EDNS0 flags indicating AAAA Record synthesis and format

4.2.1. Solution description

Section 3 of [EDNS0-Flag] defines 3 new flag bits (called SY-bits) into EDNS0 OPT [RFC2671] header, which serves as an implicit indication of the presence of DNS64 server and a NAT64 device. SY-bit values other than '000' or '111' explicitly tell the NSP prefix length. Only the DNS64 server can insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA Resource Record.

4.2.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and is designed to explicitly solve Issue #2.
- + Solves issue #4 via DNS record lifetime.
- + Can partially solve issue #5 if multiple synthetic AAAA records are included in the response and all use same format.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + EDNS0 option implementation requires changes only to DNS64 servers.
- + Does not require additional provisioning or management as the EDNS0 option is added automatically by the DNS64 server to the responses.

- + Does not involve additional queries towards the global DNS infrastructure as EDNS0 logic can be handled within the DNS64 server.

The CONS of the proposal are listed below:

- Requires end hosts to support [RFC2671] EDSN0 extension mechanism.
- Consumes scarce flag bits from EDNS0 OPT header.
- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length.
- Requires a modification to DNS64 servers to include the EDNS0 option to the synthesized responses.
- Does not provide solution for issue #3.

4.2.3. Summary

This option is included here for the sake of completeness. The consumption of three bits of the limited EDNS0 OPT space can be considered unfavorable and hence is unlikely to be accepted.

4.3. DNS Query for a Well-Known Name

4.3.1. Solution description

Section 3 of [I-D.savolainen-heuristic-nat64-discovery] describes a host behavior for discovering the presence of a DNS64 server and a NAT64 device, and heuristics for discovering the used NSP. A host requiring information for local IPv6 address synthesis or for NAT64 avoidance sends a DNS query for an AAAA record of a Well-Known IPv4-only Fully Qualified Domain Name (FQDN). If a host receives a negative reply, it knows there are no DNS64 and NAT64 in the network.

If a host receives AAAA reply, it knows the network must be utilizing IPv6 address synthesis. After receiving a synthesized AAAA Resource Record, the host may examine the received IPv6 address and use heuristics, such as "subtracting" the known IPv4 address out of synthesized IPv6 address, to find out the NSP.

The Well-Known Name may be assigned by IANA or provided some third party, including application or operating system vendor. The IPv4 address corresponding to the Well-Known Name may be resolved via A query to Well-Known Name, assigned by IANA, or hard-coded.

4.3.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Solves issue #4 via DNS record lifetime.
- + Can partially solve issue #5 if multiple synthetic AAAA records are included in the response. Can find multiple address formats.
- + Does not necessarily require any standards effort.
- + Does not require host stack or resolver changes. All required logic and heuristics can be implemented in applications that are interested in learning about address synthesis taking place.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + Hosts or applications interested in learning about synthesis and the used NSP can do the "discovery" proactively at any time, for example every time the host attaches to a new network.
- + Does not require explicit support from the network using NAT64

The CONs of the proposal are listed below:

- Requires hosting of a DNS resource record for the Well-Known Name.
- Does not provide solution for issue #3.
- This method is only able to find one NSP even if a network is utilizing multiple NSPs (issue #5) (unless DNS64 includes multiple synthetic AAAA records in response).

4.3.3. Summary

This is the only approach that can be deployed without explicit support from the network or the host. This approach could also complement explicit methods and be used as a fallback approach when explicit methods are not supported by an access network.

4.4. DNS Resource Record for IPv4-Embedded IPv6 address

4.4.1. Solution description

Section 4 of [I-D.boucadair-behave-dns-a64] defines a new DNS Resource Record (A64) that is a record specific to store a single IPv4-Embedded IPv6 address [RFC6052]. Using a dedicated Resource Record allows a host to distinguish between real IPv6 addresses and synthesized IPv6 addresses. The solution requires host to send a query for an A64 record. Positive answer with A64 record informs the requesting host that the resolved address is not a native address but an IPv4-Embedded IPv6 address. This would ease the local policies to prefer direct communications (i.e., avoid using IPv4-Embedded IPv6 addresses when a native IPv6 address or a native IPv4 address is available). Applications may be notified via new or modified API.

4.4.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and #5.
- + Solves issue #4 via DNS record lifetime.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + Synthesized addresses can be used in authoritative DNS servers.
- + Maintains the reliability of the DNS model (i.e., a synthesised IPv6 address is presented as such and not as native IPv6 address).
- + When both IPv4-Converted and native IPv6 addresses are configured for the same QNAME, native addresses are preferred.

The CONS of the proposal are listed below:

- Does not address Issue #2 or #3 in any way.
- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized.
- Requires a standardization of a new DNS Resource Record type (A64), and the implementation of it in both resolvers and servers.
- Requires a coordinated deployment between different flavors of DNS servers within the provider to work deterministically.

- Additional load the DNS servers (3 Queries, A64, AAAA and A, may be issued by a dual-stack host).
- Does not help to identify synthesized IPv6 addresses if the session does not involve any DNS queries.

4.4.3. Summary

While the proposed solution delivers explicit information about address synthesis taking place solving the Issue #1, a standardization of a new DNS record type might turn out a too overwhelming task for a solution for a temporary transition phase. Defining a new record type increases load towards DNS server as the host issues parallel A64, AAAA and A queries.

4.5. Learning the IPv6 Prefix of a Network's NAT64 using DNS

4.5.1. Solution description

Section 4.1 of [I-D.wing-behave-learn-prefix] actually proposes two DNS-based method for discovering the presence of a DNS64 server and a NAT64 device, and then a mechanism for discovering the used NSP. First, a host may learn the presence of a DNS64 server and a NAT64 device, by receiving a TXT Resource Record with a well-known (TBD IANA registered?) string followed by the NAT64 unicast IPv6 address and the prefix length. The DNS64 server would add the TXT Resource Record into the DNS response.

Second, Section 4.1 of [I-D.wing-behave-learn-prefix] also proposes specifying a new U-NAPTR [RFC4848] application to discover the NAT64's IPv6 prefix and length. The input domain name is exactly the same as would be used for a reverse DNS lookup, derived from the host's IPv6 in the ".ip6.arpa." tree. The host doing the U-NAPTR queries may need multiple queries until finds the provisioned domain name with the correct prefix length. The response to a successful U-NAPTR query contains the unicast IPv6 address and the prefix length of the NAT64 device.

4.5.2. Analysis and discussion

[Editor' note: can this be made to solve issue #5 by having multiple NSPs in TXT record?]

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Solves issue #4 via DNS record lifetime.
- + Does not require host stack or resolver changes if the required logic and heuristics is implemented in applications that are interested in learning about address synthesis taking place.

The CONS of the proposal are listed below:

- Requires standardization of a well-known names from IANA for TXT Resource Record and/or a standardization of a new U-NAPTR application.
- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length. However, it is possible that the U-NAPTR application logic is completely implemented by the application itself as noted in PROs list.
- U-NAPTR prefix learning method may entail multiple queries.
- U-NAPTR prefix learning method requires provisioning of NSPs in ".ip6.arpa." tree.
- RFC5507 [RFC5507] specifically recommends against reusing TXT Resource Records to expand DNS.
- Requires configuration on the access network's DNS servers.
- Does not provide solution for issue #3.

4.5.3. Summary

The implementation of this solution requires some changes to the applications and resolvers in a similar fashion as in solutions in Section 4.1, Section 4.2 and Section 4.4. Unlike the other DNS-based approaches, the U-NAPTR-based solution also requires provisioning information into the '.ip6.arpa.' tree which is not any more entirely internal to the provider hosting the NAT64/DNS64 service.

The iterative approach of learning the NAT64 prefix in U-NAPTR-based solution may result in multiple DNS queries, which can be considered more complex and inefficient compared to other DNS-based solutions.

4.6. Learning the IPv6 Prefix of a Network's NAT64 using DHCPv6

4.6.1. Solution description

Two individual drafts specify DHCPv6 based approaches.

Section 4.2 of [I-D.wing-behave-learn-prefix] describes a new DHCPv6 [RFC3315] option (OPTION_AFT_PREFIX_DHCP) that contains the IPv6 unicast prefix, IPv6 ASM prefix, and IPv6 SSM prefix (and their lengths) for the NAT64.

Section 4 of [I-D.boucadair-dhcpv6-shared-address-option] defines a DHCPv6 option that can be used to communicate to a requesting host the prefix used for building IPv4-Converted IPv6 addresses together with the format type. Provisioning the format type is required so as to be correctly handled by the NAT64-enabled devices deployed in a given domain.

4.6.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1, Issue #2, Issue #3 and Issue #4 via DHCPv6 information lifetime.
- + Does not involve DNS system. Therefore, applications that would not normally initiate any DNS queries can still learn the NAT64 prefix.
- + DHCPv6 is designed to provide various kinds of configuration information in a centrally managed fashion.

The CONS of the proposal are listed below:

- Change of NSP requires change to DHCPv6 configuration.
- Requires at least Stateless DHCPv6 client on hosts.
- Requires support on DHCPv6 clients, which is not trivial in all operating systems.
- The DHCPv6-based solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

- A new DHCPv6 option is required and the corresponding changes to both DHCPv6 clients and servers.

If DHCPv6 would include multiple NSPs issue #5 could be solved as well, but only if nodes as a group would select different NSPs hence supporting load-balancing. As this is not clear this item is not yet listed under PRO nor CON.

4.6.3. Summary

The DHCPv6-based solution would be a good solution in a sense it hooks into general IP configuration phase, allows easy updates when configuration information changes and does not involve DNS in general. Use of DHCPv6 requires configuration changes on DHCPv6 clients and servers and in some cases may also require implementation changes. Furthermore, it is not obvious that all devices that need translation services would implement stateless DHCPv6. For example, cellular 3GPP networks do not mandate hosts or network to implement or deploy DHCPv6.

4.7. Learning the IPv6 Prefix of a Network's NAT64 using Router Advertisements

4.7.1. Solution description

Section 3.3 of [RA-Learn-Prefix] describes a new Router Advertisement (RA) [RFC4861] option (OPTION_AFT_PREFIX_RA) that contains the IPv6 unicast prefix, IPv6 ASM prefix, and IPv6 SSM prefix (and their lengths) for the NAT64. The RA option is essentially the same as for DHCPv6 discussed in Section 4.6.

4.7.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1, Issue #2, and Issue #3.
- + Can solve Issue #4 if lifetime information can be communicated.

The CONS of the proposal are listed below:

- Requires configuration and managements of all access routers to emit correct information in RA. This could, for example, be accomplished somehow by piggybacking on top of routing protocols (which would then require enhancements to routing protocols)

- In some operating systems it may not be trivial to transfer information obtained in RA to upper layers
- Requires changes to host operating system's IP stack
- NSP change requires changes to access router configuration
- Requires standardization of a new option to Router Advertisement that is generally unfavored approach
- The RA-based solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

If RA would include multiple NSPs issue #5 could be solved as well, but only if nodes as a group would select different NSPs hence supporting load-balancing. As this is not clear this item is not yet listed under PRO nor CON.

4.7.3. Summary

The RA-based solution would be a good solution in a sense it hooks into general IP configuration phase, allows easy updates when configuration information changes and does not involve DNS in general. However, generally introducing any changes to the Neighbor Discovery Protocol that are not absolutely necessary are unfavored due the impact on both network node side and end host IP stack implementations.

Compared to the DHCPv6 equivalent solution in Section 4.6 the management overhead is greater with RA-based solution. In case of DHCPv6-based solution the management can be centralized to few DHCPv6 servers compared to RA-based solution where each access router is supposed to be configured with the same information.

4.8. Using application layer protocols such as STUN

4.8.1. Solution description

Application layer protocols, such as Session Traversal Utilities for NAT (STUN) [RFC5389], which define methods for endpoints to learn their external IP addresses could be used for NAT64 and NSP discovery. This document focuses on STUN, but the protocol could be something else as well.

A host must first use DNS to discover IPv6 representation(s) of STUN server(s) IPv4 address(es), because the host has no way to directly use IPv4 addresses to contact to STUN server(s).

After learning the IPv6 address of a STUN server the STUN client sends a request to the STUN server containing new 'SENDING-TO' attribute that tells to the server the IPv6 address the client sent the request to. In a reply the server includes another new attribute called 'RECEIVED-AS', which contains server's IP address the request arrived on. After receiving the reply the client compares 'SENDING-TO' and 'RECEIVED-AS' attributes to find out an NSP candidate.

4.8.2. Analysis and discussion

This solution is relatively similar as described in section 4.3, but instead of using DNS uses STUN to get input for heuristic algorithms.

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1 and Issue #2.
- + Does not require host changes or supportive protocols such as DNS or DHCPv6. All required logic and heuristics can be implemented in applications that are interested in learning about address synthesis taking place.
- + The solution is backward compatible from 'legacy' hosts and servers point of view.
- + Hosts or applications interested in learning about synthesis and the used NSP can do the "discovery" proactively at any time, for example every time the host attaches to a new network.
- + Does not require explicit support from the network using NAT64.
- + Can possibly be bundled to existing STUN message exchanges as new attributes and hence client can learn its external IPv4 address and a NSP/WKP with the same exchange
- + Can be used to confirm the heuristics by synthesizing IPv6 address of another STUN server or by synthesizing IPv6 address of first STUN server after host has heuristically determined NSP using method from section 4.3. I.e. the connectivity test could be done with STUN.
- + True IPv4 destination address is used in NSP determination instead of IPv4 address received from DNS. This may increase reliability.

- + The same STUN improvement could also be used to reveal NAT66 on the data path, if the 'RECEIVED-AS' would contain different IPv6 address than 'SENDING-TO'.

The CONS of the proposal are listed below:

- Requires a server on the network to respond the queries.
- Requires standardization if done as extension to STUN.
- The solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).
- Does not solve issue #3 if STUN server's synthetic IPv6 address is provisioned via DNS.
- Does not solve issue #4 as the STUN server would not be aware of learned NSP's validity time.
- Does not solve issue #5 as the STUN server would not be aware of multiple NSP prefixes.
- Heavyweight solution especially if an application does not otherwise support STUN.

4.8.3. Summary

The STUN, or similar, protocol based approach is a second way to solve the problem without explicit access network support. The heuristics for NSP discovery would still be in the client, however, the result may be more reliable as actual IPv4 destination address is compared to IPv6 address used in sending. The additional benefit of STUN is that the client learns its public IPv4 address with the same message exchange. STUN could also be used as the connectivity test tool if the client would first heuristically determine NSP out of DNS as described in section 4.3, synthesize IPv6 representation of STUN server's IPv4 address, and then tests connectivity to the STUN server.

As an additional benefit the STUN improvement could be used for NAT66 discovery.

4.9. Learning the IPv6 Prefix of a Network's NAT64 using Access Technology Specific Methods

4.9.1. Solution description

Several link layers on different access systems have an attachment time signaling protocols to negotiate various parameter used later on the established link layer connection. Examples of such include 3GPP Non-Access-Stratum (NAS) signaling protocol [3GPP.24.301] among other link layers and tunneling solutions. There, using NAS signaling it could be possible to list all NSPs with their respective prefix lengths in generic protocol configuration option containers during the network access establishment. The lack of NSPs in protocol configuration option containers would be an implicit indication that there is no NAT64 present in the network.

4.9.2. Analysis and discussion

The PROs of the proposal are listed below:

- + Can be used to solve Issue #1, Issue #2, Issue #3 and Issue #5.
- + Can solve Issue #4 if lifetime information is also communicated.

The CONs of the proposal are listed below:

- Requires configuration and managements of all access routers/gateway to emit correct information in "link/lower layer" signaling. In a case the NAT64 functionality is implemented into the access router/gateway itself that terminates the generic protocol configuration exchange, then the configuration management can be automated.
- In some operating systems it may not be trivial to transfer information obtained in "link/lower layers" to upper layers.
- NSP change may require changes to access router/gateway configuration.
- Requires standardization of a new configuration parameter exchange/container for each access system of interest. The proposed solution is indeed specific to each access technology.

4.9.3. Summary

The Access Technology-based solution would be a good solution in a sense it hooks into general network access establishment phase, allows easy updates when configuration information changes and does not involve DNS in general. However, generally introducing any changes to the link/lower layers is a long and slow router, and yet is access technology/system specific.

Compared to the RA equivalent solution in Section 4.7 the management overhead is equivalent or even less than RA-based solution.

5. Conclusion

Our conclusion is to recommend publishing the Well-Known DNS Name heuristic-based method (see Section 4.3) as an Informational IETF document for applications and host implementors to implement as-is. If Standards Track work is seen beneficial, then our recommendation is the standardization of EDNS0 option. The reasoning for our conclusion is discussed in the following paragraphs.

Of the different issues we give most weight for issues #1 and #2. We are not giving much weight for the Issue #3 'DNS should not be required', as cases where hosts need to synthesize IPv6 addresses but do not have DNS available seem rare for us. Even if application does not otherwise utilize DNS, it ought to be able to trigger simple DNS query to find out WKP/NSP. Issue #4 is handled by majority of solutions. Issue #5 is considered to be mostly insignificant from an individual hosts point of view as it would use only one NSP at a time, while different hosts could be using different NSPs, hence supporting load-balancing targets. None of the discussed solutions support learning of possible new or indicating support for multiple algorithms for address synthesis other than the one described in [RFC6052].

The DNS64 entity has to be configured with WKP/NSP in order for it to do synthesis and hence using DNS also for delivering the synthesis information sounds logical. The fact that the synthesis information fate-shares the information received in the DNS response is a valuable attribute and reduces the possible distribution of stale prefix information. On the contrary, use of DHCPv6 would require additional trouble configuring DHCPv6 servers and ensuring DHCPv6 clients are in place, and furthermore that the NAT64, DHCPv6 (and possible even some DNS64) servers are all in sync. RA-based mechanisms are operationally expensive as configuration would have to be placed and maintained in the access routers. Furthermore, both DHCPv6 and RA based mechanisms involve entities that do not otherwise need to be aware of protocol translation (only need to know DNS server addresses).

Of the DNS-based mechanisms we favor EDNS0 option due to its lightweight nature. All the A64, DNS SRV, and EDNS0 approaches would require standardization and deployment efforts that may be excessive compared to the size of the problem. The U-NAPTR-based approach would require provisioning information into the '.ip6.arpa' tree which would not be entirely internal for the provider.

The two heuristic-based approaches could be taken into use at once and would provide benefits in networks utilizing protocol translation, but on the long run their usefulness depends how well networks will deploy explicit methods.

6. Security Considerations

The security considerations are essentially similar to what is described in DNS64 [I-D.ietf-behave-dns64]. Forgery of information required for IPv6 address synthesis may allow an attacker to insert itself as middle man or to perform denial-of-service attack. The DHCPv6 and RA based approaches are vulnerable for the forgery as the attacker may send forged RAs or act as a rogue DHCPv6 server (unless DHCPv6 authentication or SEND are used). If the attacker is already able to modify and forge DNS responses (flags, addresses of know IPv4-only servers, records, etc), ability to influence local address synthesis is likely of low additional value. Also, a DNS-based mechanism is only as secure as the method used to configure the DNS server's IP addresses on the host. Therefore, if the host cannot trust e.g. DHCPv6 it cannot trust the DNS server learned via DHCPv6 either, unless the host has a way to authenticate all DNS responses.

7. IANA Considerations

This document is informative and has no actions to IANA.

8. Contributors

The following people have contributed text to this document.

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange-ftgroup.com

9. Acknowledgements

Authors would like to thank Dan Wing and Christian Huitema especially for the STUN idea for their valuable comments and discussions.

10. Informative References

- [3GPP.24.301]
3GPP, "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS)", 3GPP TS 24.301 8.8.0, December 2010.
- [EDNS0-Flag]
Korhonen, J. and T. Savolainen, "EDNS0 Flags Indicating AAAA Record Synthesis and Format", draft-korhonen-edns0-synthesis-flag-00 (work in progress), June 2010.
- [I-D.boucadair-behave-dns-a64]
Boucadair, M. and E. Burgey, "A64: DNS Resource Record for IPv4-Embedded IPv6 Address", draft-boucadair-behave-dns-a64-02 (work in progress), September 2010.
- [I-D.boucadair-dhcpv6-shared-address-option]
Boucadair, M., Levis, P., Grimault, J., Savolainen, T., and G. Bajko, "Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions", draft-boucadair-dhcpv6-shared-address-option-01 (work in progress), December 2009.
- [I-D.carpenter-referral-ps]
Carpenter, B., Jiang, S., and B. Zhou, "Problem Statement for Referral", draft-carpenter-referral-ps-01 (work in progress), August 2010.
- [I-D.ietf-behave-dns64]
Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-11 (work in progress), October 2010.
- [I-D.ietf-behave-v6v4-framework]
Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-10 (work in progress), August 2010.
- [I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in

progress), July 2010.

[I-D.korhonen-edns0-synthesis-flag]

Korhonen, J. and T. Savolainen, "EDNS0 Option for Indicating AAAA Record Synthesis and Format", draft-korhonen-edns0-synthesis-flag-02 (work in progress), February 2011.

[I-D.savolainen-heuristic-nat64-discovery]

Savolainen, T. and J. Korhonen, "Discovery of a Network-Specific NAT64 Prefix using a Well-Known Name", draft-savolainen-heuristic-nat64-discovery-01 (work in progress), February 2011.

[I-D.venaas-behave-mcast46]

Venaas, S., Asaeda, H., SUZUKI, S., and T. Fujisaki, "An IPv4 - IPv6 multicast translator", draft-venaas-behave-mcast46-02 (work in progress), December 2010.

[I-D.venaas-behave-v4v6mc-framework]

Venaas, S., Li, X., and C. Bao, "Framework for IPv4/IPv6 Multicast Translation", draft-venaas-behave-v4v6mc-framework-02 (work in progress), December 2010.

[I-D.wing-behave-learn-prefix]

Wing, D., "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-04 (work in progress), October 2009.

[RA-Learn-Prefix]

Wing, D., Wang, X., and X. Xu, "Learning the IPv6 Prefixes of an IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-03 (work in progress), July 2009.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

- [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.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5507] IAB, Faltstrom, P., Austein, R., and P. Koch, "Design Choices When Expanding the DNS", RFC 5507, April 2009.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

Authors' Addresses

Jouni Korhonen (editor)
Nokia Siemens Networks
Linnoitustie 6
FI-02600 Espoo
Finland

Email: jouni.nospam@gmail.com

Teemu Savolainen (editor)
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com

Behavior Engineering for Hindrance
Avoidance
Internet-Draft
Intended status: Informational
Expires: July 10, 2011

R. Penno
Juniper Networks
T. Saxena
Cisco Systems
M. Boucadair
France Telecom
S. Sivakumar
Cisco Systems
January 6, 2011

Analysis of 64 Translation
draft-penno-behave-64-analysis-06

Abstract

Due to specific problems, NAT-PT was deprecated by the IETF as a mechanism to perform IPv6-IPv4 translation. Since then, new efforts have been undertaken within IETF to standardize alternative mechanisms to perform IPv6-IPv4 translation. This document evaluates how the new translation mechanisms avoid the problems that caused the IETF to deprecate NAT-PT.

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 RFC 2119 [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 July 10, 2011.

Copyright Notice

Copyright (c) 2011 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
1.1. Definition	3
1.2. Context	3
1.3. Scope	4
2. Analysis of 64 Translation Against Concerns of RFC4966	4
2.1. Problems Not Addressed by 64	4
2.2. Problems Addressed by 64	7
3. Conclusions	9
4. IANA Considerations	11
5. Security Considerations	11
6. Acknowledgements	11
7. References	12
7.1. Normative References	12
7.2. Informative References	13
Authors' Addresses	14

1. Introduction

1.1. Definition

This document uses 64 proposal (or 64 for short) to refer to the mechanisms defined in the following documents:

- o Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers [I-D.ietf-behave-v6v4-xlate-stateful]
- o DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers [I-D.ietf-behave-dns64]
- o IPv6 Addressing of IPv4/IPv6 Translators [RFC6052]
- o Framework for IPv4/IPv6 Translation [I-D.ietf-behave-v6v4-framework]

1.2. Context

The current 64 proposal is widely seen as the next step in the evolution of interconnection techniques enabling communications between IPv6-only and IPv4-only networks. One of the building blocks of this proposal is decoupling the DNS functionality from the protocol translation itself.

This approach is pragmatic in the sense that there is no dependency on DNS implementation for the successful NAT handling. As long as there is a function (e.g., DNS64 [I-D.ietf-behave-dns64] or other means) that can construct an IPv6-Embedded IPv4 address with a pre-configured IPv6 prefix, an IPv4 address and a suffix (refer to [RFC6052]), NAT64 will work just fine.

To understand the 64 proposal, we must keep in mind that the focus of this proposal is on the deployment and not the implementation details. As long as a NAT64 implementation conforms to the expected behaviour, as desired in the deployment scenario, the details are not very important as mentioned in this excerpt from [I-D.ietf-behave-v6v4-xlate-stateful]:

"A NAT64 MAY perform the steps in a different order, or MAY perform different steps, but the externally visible outcome MUST be the same as the one described in this document."

1.3. Scope

This document provides an analysis of how the proposed set of documents that specify stateful IPv6-only to IPv4-only translation and replace NAT-PT [RFC2766] address the issues raised in [RFC4966].

As a reminder, it is worth mentioning the 64 proposal analysis is limited in the sense that hosts from IPv6 networks can initiate a communication to IPv4 network/Internet, but not vice-versa. This corresponds to Scenario 1 and Scenario 5 described in [I-D.ietf-behave-v6v4-framework]. Hence, the scenario of servers moving to IPv6 while clients remaining IPv4 remains unaddressed. Of course, IPv6 to IPv4 communications can also be supported if static bindings are configured on the stateful NAT64.

The 64 proposal, just like any other technique under development, has some positives and some drawbacks. The ups and downs of the proposal must be clearly understood while going forward with its future development.

The scope of this document does not include stateless translation.

2. Analysis of 64 Translation Against Concerns of RFC4966

Of the set of problems pointed out in [RFC4966], the 64 proposal addresses some of them, whereas leaves others unaddressed.

Some issues mentioned in [RFC4966] were solved by [RFC4787], [RFC5382] and [RFC5508]. At the time when NAT-PT was published these recommendations were not in place but they are orthogonal to the translation algorithm per se, therefore they could be implemented with NAT-PT. On the other hand, NAT64 explicitly mentions that these recommendations need to be followed and thus should be seen as a complete specification.

It is also worth pointing out that the scope of the 64 proposal is reduced when compared to NAT-PT. Following is a point by point analysis of the problems.

2.1. Problems Not Addressed by 64

Problems discussed in [RFC4966], which are not addressed by the 64 proposal:

1. Disruption of all protocols that embed IP addresses (and/or ports) in packet payloads or apply integrity mechanisms using IP addresses (and ports).

Analysis: In the case of FTP [RFC0959] this problem is addressed by the use of a FTP64 ALG [I-D.ietf-behave-ftp64] which is a workaround solution. In the case of SIP [RFC3261], no specific issue is induced by 64; the same techniques for NAT traversal can be used when a NAT64 is involved in the path (e.g., ICE [RFC5245], Hosted NAT Traversal [RFC5853], embedded SIP ALGs, etc.). The functioning of other protocols is left unaddressed. Note that the traversal of NAT64 by application embedding IP address literal is not specific to NAT64 but generic to all NAT-based solutions.

2. Inability to redirect traffic for protocols that lack de-multiplexing capabilities or are not built on top of specific transport-layer protocols for transport address translations.

Analysis: This issue is not specific to 64 but to all NAT-based solutions.

3. Loss of information due to incompatible semantics between IPv4 and IPv6 versions of headers and protocols.

Analysis: This issue is not specific to 64 but due to the design of IPv4 and IPv6.

4. Need for additional state and/or packet reconstruction in dealing with packet fragmentation. Otherwise, implement no support for fragments.

Analysis: This issue is not specific to 64 but to all NAT-based solutions. [I-D.ietf-behave-v6v4-xlate-stateful] specifies how to handle fragmentation; appropriate recommendations to avoid fragmentation-related DoS attacks are proposed (e.g., limit resources to be dedicated to out of order fragments).

5. Interaction with SCTP [RFC4960] and multihoming.

Analysis: SCTP is out of scope of 64. Only TCP and UDP transport protocols are within the scope of 64.

6. Need for the NAT64-capable device to act as proxy for correspondent node when IPv6 node is mobile, with consequent restrictions on mobility.

Analysis: This is not specific to NAT64 but to all NAT flavors. Refer to [I-D.haddad-mext-nat64-mobility-harmful] for an early analysis on mobility complications encountered

when NAT64 is involved.

7. Inability to handle multicast traffic.

Analysis: This problem is not addressed by the current 64 specifications.

8. Scalability concerns together with introduction of a single point of failure and a security attack nexus.

Analysis: This is not specific to NAT64 but to all stateful NAT flavors.

9. Creation of a DoS (Denial of Service) threat relating to exhaustion of memory and address/port pool resources on the translator.

Analysis: This specific DoS concern on Page 6 of [RFC4966] is under a DNS-ALG heading in that document, and refers to NAT-PT's creation of NAT mapping state when a DNS query occurred. With the new IPv6-IPv4 translation mechanisms, DNS queries do not create any mapping state. Thus, this concern is fully eliminated with the new IPv6-IPv4 translation mechanisms.

10. Restricted validity of translated DNS records: a translated record may be forwarded to an application that cannot use it.

Analysis: If a node on the IPv4 side forwards the address of the other endpoint to a node which cannot reach the NAT box or is not covered under the endpoint-independent constraint of NAT, then the new node will not be able to initiate a successful session.

Actually, this is not a limitation of 64 (or even NAT-PT) but a deployment context where shared IPv4 addresses managed by the NAT64 are not globally reachable. The same limitation can be encountered when referrals (even without any NAT in the path) include reachability information with limited reachability scope (See [I-D.carpenter-behave-referral-object] for more discussion about scope-related issues).

11. Unless UDP encapsulation is used for IPsec [RFC3948], traffic using IPsec AH (Authentication Header), in transport and tunnel mode, and IPsec ESP (Encapsulating Security Payload), in transport mode, is unable to be carried through NAT-PT without terminating the security associations on the NAT-PT, due to their usage of cryptographic integrity protection.

Analysis: This is not specific to NAT64 but to all NAT flavours.

12. Address selection issues when either the internal or external hosts implement both IPv4 and IPv6.

Analysis: This is out of scope of 64 since Scenarios 1 and 5 of [I-D.ietf-behave-v6v4-framework] assume IPv6-only hosts.

Therefore this issue is not resolved and mitigation techniques outside the 64 need to be used. These techniques may allow to offload NAT64 resources and prefer native communications which do not involve address family translation. Avoiding NAT devices in the path is encouraged for mobile nodes in order to save power consumption due to keepalive messages which are required to maintain NAT states ("always-on" services). An in-depth discussion can be found in [I-D.wing-behave-dns64-config].

2.2. Problems Addressed by 64

Problems, identified in [RFC4966], which are adequately addressed by the 64 proposal:

1. Constraints on network topology (as it relates to DNS-ALG; see Section 3.1 of [RFC4966]).

Analysis: This issue has mitigated severity as the DNS is separated from the NAT functionality. Nevertheless, a minimal coordination may be required to ensure that the NAT64 to be crossed (the one to which the IPv4-Converted IPv6 address returned to a requesting host) must be in the path and has also sufficient resources to handle received traffic.

2. Inappropriate translation of responses to A queries from IPv6 nodes.

Analysis: DNS64 [I-D.ietf-behave-dns64] does not resolve A queries.

3. Address selection issues and resource consumption in a DNS-ALG with multi-addressed nodes.

Analysis: Since the DNS-ALG is not there and communications initiated from the IPv4 side are not supported, there is no need to maintain temporary states in anticipation of connections.

4. Limitations on DNS security capabilities when using a DNS-ALG.

Analysis: A DNSSEC validating stub resolver behind a DNS64 in server mode is not supported. Therefore if a host wants to do its own DNSSEC validation, and it wants to use a NAT64, the host has to also perform its own DNS64 synthesis. Refer to Section 3 of [I-D.ietf-behave-dns64] for more details.

5. Creation of a DoS (Denial of Service) threat relating to exhaustion of memory and address/port pool resources on the translator.

Analysis: This specific DoS concern on Page 6 of [RFC4966] is under a DNS-ALG heading in that document, and refers to NAT-PT's creation of NAT mapping state when a DNS query occurred. With the new IPv6-IPv4 translation mechanisms, DNS queries do not create any mapping state in the NAT64. Thus, this concern is fully eliminated in 64.

6. Requirement for applications to use keepalive mechanisms to workaround connectivity issues caused by premature timeout for session table and BIB entries.

Analysis: Since NAT64 follows some of the [RFC4787], [RFC5382] and [RFC5508] requirements, there is a high lower bound for the lifetime of sessions. In NAT-PT this was unknown and applications needed to assume the worst case. For instance, in NAT64, the lifetime for a TCP session is approximately 2 hours, so not much keep-alive signalling overhead is needed.

Application clients (e.g., VPN clients) are not aware of the timer configured in the NAT device. For unmanaged services, a conservative approach would be adopted by applications which issue frequent keepalive messages to be sure that an active mapping is still be maintained by any involved NAT64 device even if the NAT64 complies with TCP/UDP/ICMP BEHAVE WG specifications.

Note that keepalive messages may be issued by applications to ensure that an active entry is maintained by a firewall, with or without a NAT in the path, which is located in the boundaries of a local domain.

7. Lack of address mapping persistence: Some applications require address retention between sessions. The user traffic will be disrupted if a different mapping is used. The use of the DNS-ALG to create address mappings with limited lifetimes means that applications must start using the address shortly after the

mapping is created, as well as keep it alive once they start using it.

Analysis: In the context of 64, the external IPv4 address (representing the IPv6 host in the IPv4 network) is assigned by the NAT64 machinery and not the DNS64 function. Address persistence can be therefore easily ensured by the NAT64 function (which complies with BEHAVE NAT recommendations). Address persistence should be guaranteed for both dynamic and static bindings.

In the IPv6 side of the NAT64, the same IPv6 address is used to represent an IPv4 host; no issue about address persistence is raised in IPv6 network.

3. Conclusions

The above analysis of the solutions provided by the 64 proposal shows that the majority of the problems that are not directly related to the decoupling of NAT and DNS remain unaddressed. Some of these problems are not specific to 64 but are generic to all NAT-based solutions.

This points to several shortcomings of 64 proposal which must be addressed if the future network deployments have to move reliably towards 64 as a solution to IPv6-IPv4 interconnection.

Some of the issues, as pointed out in [RFC4966], have possible solutions. However these solutions will require significant updates to the 64 proposal, increasing its complexity.

The following table summarizes the conclusions based on the analysis of 64 proposal.

Issue	NAT-PT Specific	Exists in NAT64	DNS ALG Specific	Generic NAT	Can be solved?
Protocols embedding addresses	No	Yes	No	Yes	Yes
Protocols without demux capability	No	Yes	No	Yes	No
Binding state decay	No	Yes	No	Yes	No
Loss of information	No	Yes	No	No	No
Fragmentation	No	No	No	Yes	Yes
SCTP and Multihoming interaction	No	Yes	No	Yes	Yes
Proxy corresp node for MIPv6	Yes	Yes	No	Yes	??
Multicast	No	Yes	No	Yes	Yes
Topology constraints with DNS-ALG	Yes	No	Yes	No	Yes
Scale and Single point of failure	No	Yes	No	Yes	Yes
Lack of address persistence	No	Yes	No	Yes	No
DoS attacks	No	Yes	No	Yes	Yes

Address selection issues with Dual stack hosts	Yes	No	Yes	No	Yes
Non-global validity of Translated RR records	Yes	No	Yes	Yes	Yes
Incorrect translation of A responses	Yes	No	Yes	No	Yes
DNS-ALG and Multi-addressed nodes	No	Yes	No	Yes	Yes
DNSSEC limitations	No	Yes	No	Yes	Yes

Table 1: Summary of NAT64 analysis

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

5. Security Considerations

This document does not specify any new protocol or architecture. It only analyses how BEHAVE WG 64 documents mitigate concerns raised in [RFC4966] and which ones are still unaddressed.

6. Acknowledgements

Many thanks to Marcelo Bagnulo for his comments.

7. References

7.1. Normative References

- [I-D.ietf-behave-dns64]
Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum,
"DNS64: DNS extensions for Network Address Translation
from IPv6 Clients to IPv4 Servers",
draft-ietf-behave-dns64-11 (work in progress),
October 2010.
- [I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful
NAT64: Network Address and Protocol Translation from IPv6
Clients to IPv4 Servers",
draft-ietf-behave-v6v4-xlate-stateful-12 (work in
progress), July 2010.
- [RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol",
STD 9, RFC 959, October 1985.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation
(NAT) Behavioral Requirements for Unicast UDP", BCP 127,
RFC 4787, January 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol",
RFC 4960, September 2007.
- [RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network
Address Translator - Protocol Translator (NAT-PT) to
Historic Status", RFC 4966, July 2007.
- [RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P.
Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142,
RFC 5382, October 2008.
- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT
Behavioral Requirements for ICMP", BCP 148, RFC 5508,
April 2009.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X.
Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052,
October 2010.

7.2. Informative References

- [I-D.carpenter-behave-referral-object]
Carpenter, B., Boucadair, M., Halpern, J., Jiang, S., and K. Moore, "A Generic Referral Object for Internet Entities", draft-carpenter-behave-referral-object-01 (work in progress), October 2009.
- [I-D.haddad-mext-nat64-mobility-harmful]
Haddad, W. and C. Perkins, "A Note on NAT64 Interaction with Mobile IPv6", draft-haddad-mext-nat64-mobility-harmful-01 (work in progress), April 2010.
- [I-D.ietf-behave-ftp64]
Beijnum, I., "An FTP ALG for IPv6-to-IPv4 translation", draft-ietf-behave-ftp64-06 (work in progress), November 2010.
- [I-D.ietf-behave-v6v4-framework]
Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-10 (work in progress), August 2010.
- [I-D.wing-behave-dns64-config]
Wing, D., "DNS64 Resolvers and Dual-Stack Hosts", draft-wing-behave-dns64-config-02 (work in progress), February 2010.
- [RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5853] Hautakorpi, J., Camarillo, G., Penfield, R., Hawrylyshen, A., and M. Bhatia, "Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments", RFC 5853, April 2010.

Authors' Addresses

Reinaldo Penno
Juniper Networks
1194 N Mathilda Avenue
Sunnyvale, California 94089
USA

Email: rpenno@juniper.net

Tarun Saxena
Cisco Systems

Email: tasaxena@cisco.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange-ftgroup.com

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, North Carolina 27709
USA

Email: ssenthil@cisco.com

Behavior Engineering for Hindrance
Avoidance
Internet-Draft
Intended status: Informational
Expires: April 3, 2011

T. Tsou, Ed.
Huawei Technologies
W. Li, Ed.
China Telecom
T. Taylor
Huawei Technologies
September 30, 2010

Port Management To Reduce Logging In Large-Scale NATs
draft-tsou-behave-natx4-log-reduction-02

Abstract

Various IPv6 transition strategies require the introduction of large-scale NATs (e.g. AFTR, NAT64) to share the limited supply of IPv4 addresses available in the network until transition is complete. There has recently been debate over how to manage the sharing of ports between different subscribers sharing the same IPv4 address. One factor in the discussion is the operational requirement to log the assignment of transport addresses to subscribers. It has been argued that dynamic assignment of individual ports between subscribers requires the generation of an excessive volume of logs. This document suggests a way to achieve dynamic port sharing while keeping log volumes low.

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 3, 2011.

Copyright Notice

Copyright (c) 2010 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
1.1. Requirements Language	3
2. A Suggested Solution	3
3. Issues Of Traceability	5
4. Other Considerations	7
5. IANA Considerations	7
6. Security Considerations	7
7. Acknowledgements	7
8. Appendix A: Configure Server Software to Log Source Port	7
8.1. Apache	7
8.2. Postfix	8
8.3. Sendmail	8
8.4. sshd	8
8.5. Cyrus IMAP and UW IMAP	9
9. References	9
9.1. Normative References	9
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

During the IPv6 transition period, some large-scale NAT devices may be introduced, e.g. DS-Lite AFTR, NAT64. When a NAT device needs to set up a new connection for a given internal address behind the NAT, it needs to create a new mapping entry for the new connection, which will contain source IP address, source port, converted source IP address, converted source port, protocol (TCP/UDP), etc. If the connection is ICMP, a mapping entry may include source IP address, converted source IP address, source identifier, converted source identifier, etc.

For the purpose of troubleshooting, and also as required by regulations, operators must keep logs of network NAT mapping entries for a period of time, e.g. 6 months or one year [I-D.ietf-intarea-shared-addressing-issues], so the NAT device needs to generate logs for mapping entries in addition to other information. A traditional method is to generate a log for each mapping entry. When a connection expires, the mapping entry will be deleted, and the corresponding log is stored locally or sent to a log storage server.

Some high performance NAT devices may need to create a large amount of new sessions per second. If logs are generated for each mapping entry, the log traffic could reach tens of megabytes per second or more, which would be a problem for log generation, transmission and storage.

To reduce the cost of log storage, [I-D.nishitani-cgn] proposes to fix the port range for each user/CPE, and only one log will be generated for each user. But this would significantly reduce the number of subscribers that could share a public IP address, as discussed in [I-D.softwire-dual-stack-lite].

1.1. Requirements Language

This draft includes no requirements language.

2. A Suggested Solution

We propose a solution that allows dynamic sharing of port ranges between users while minimizing the number of logs that have to be generated. Briefly, ports are allocated to the user in blocks. Logs are generated only when blocks are allocated or deallocated. This provides the necessary traceability while reducing log generation by a factor equal to the block size, as compared with fully dynamic port allocation.

Here is how the proposal would work in greater detail. When the user sends out the first packet, a port resource pool is allocated for the user, e.g. assign ports 2001~2300 of a public IP address to the user's resource pool. Only one log should be generated for this port block. When the NAT needs to set up a new mapping entry for the user, it can use a port in the user's resource pool and the corresponding public IP address. If the user needs more port resources, the NAT can allocate another port block, ports 3501~3800, to the user's resource pool. Again, just one log needs to be generated for this port block. A log may contain the following information: source IP address, converted source IP address, port range, start time, end time, and some other necessary information.

There is an alternative way of allocating port blocks [I-D.bajko-priaddrassign]. The ports in a block do not have to be contiguous. Due to security concerns, the port numbers could be worked out using some random algorithm along with some initial parameters. The randomization algorithm would be applied at the NAT when it generates a new mapping. The algorithm and initial parameters together are required to define a discrete subset of the entire available port range (1024 to 65535), such that it is possible to assign the complete range to different internal addresses as required by varying the initial parameters. When generating a log message, these parameters instead of the upper and lower bounds of a port range would be included in the log.

Suppose now that a given internal address has been assigned more than one block of ports. Regardless of whether the ports within a block are specified by a simple range or a random algorithm, it is clear that the overall preference for port randomization will be better achieved by spreading out new port assignments over all of the blocks assigned to that internal address. That means that the NAT should first select one of the assigned blocks pseudo-randomly before applying any randomization algorithm within the block. Further discussion of this point occurs below as part of the discussion of block deallocation.

The individual sessions using ports within a port block will start and end at different times. If no ports in some port block are used for some configurable time, the NAT can remove the port block from the resource pool allocated to a given internal address, and make it available for other users. The deallocation may be logged when it occurs, although some would view such logging as redundant.

The deallocation procedure presents a number of difficulties in practice. The first problem is the choice of timeout value for the block. If idle timers are applied for the individual mappings (sessions) within the block, and these conform to the recommendations

for NAT behaviour for the protocol concerned, then the additional time that might be configured as a guard for the block as a whole need not be more than a few minutes. The block timer in this case serves only as a slightly more conservative extension of the individual session idle timers. If, instead, a single idle timer is used for the whole block, it must itself conform to the recommendations for the protocol with which that block of ports is associated. For example, REQ-5 of [RFC5382] requires an idle timer expiry duration of at least 2 hours and 4 minutes.

The next issue with port block deallocation is the conflict between the desire to randomize port allocation and the desire to make unused resources available to other internal addresses. As mentioned above, ideally port selection will take place over the entire set of blocks allocated to the internal address. However, taken to its fullest extent, such a policy will minimize the probability that all ports in any given block are idle long enough for it to be released. As an alternative, it is suggested that when choosing which block to select a port from, the NAT should omit from its range of choice the block that has been idle the longest, unless no ports are available in any of the other blocks. The expression "block that has been idle the longest" designates the block in which the time since the last packet was observed in any of its sessions, in either direction, is earlier than the corresponding time in any of the other blocks assigned to that internal address. As [I-D.ietf-intarea-shared-addressing-issues] points out, port randomization is just one security measure of several, and the loss of randomness incurred by the suggested procedure is justified by the increased utilization of port resources it allows.

3. Issues Of Traceability

The whole point of this proposal is to allow the NAT to support regulatory requirements for traceability of usage. So it is only right to verify that these requirements can be met with the proposal made in the previous section. There are two cases:

1. the investigating authority requires a complete record of the activities of a target individual;
2. the investigating authority is concerned with tracking down the user responsible for wrongful behaviour at a specific end point (e.g., server, individual user, enterprise network).

Assuming that per-session logging at the NAT is to be avoided in general (the whole point of this draft), the first requirement can only be met by identifying a target device in advance and enabling

per-session logging for the internal address assigned to that device (... and variations for multi-address situations). This case is basically out of scope of the draft.

Section 11 of [I-D.ietf-intarea-shared-addressing-issues] provides a good discussion of the traceability issue. Complete traceability given the NAT logging practices proposed in this draft requires that the remote destination record the source port of a request along with the source address (and presumably protocol, if not implicit). In addition, the logs at each end must be timestamped, and the clocks must be synchronized within a certain degree of accuracy. Here is one reason for the guard timing on block release, to increase the tolerable level of clock skew between the two ends.

The ability to configure various server applications to record source ports has been investigated, with the following results:

- o Source port recording can be configured in Apache, Postfix, sendmail and sshd. Please refer to the appendix for configuration guide.
- o Source port recording is not supported by IIS, Cyrus IMAP and UW IMAP. But it should not be too difficult to get Cyrus IMAP and UW IMAP to support it by modifying the source code.

Where source port logging can be enabled, this memo strongly urges the operators to do so. Similarly, intrusion detection systems should capture source port as well as source address of suspect packets.

In some cases [I-D.ietf-intarea-shared-addressing-issues], a server may not record the source port of a connection. To allow traceability, the NAT device needs to record the destination IP address of a connection. As [I-D.ietf-intarea-shared-addressing-issues] points out, this will provide an incomplete solution to the issue of traceability because multiple users of the same shared public IP address may access the service at the same time. From the point of view of this draft, in such situations the game is lost, so to speak, and port allocation at the NAT might as well be completely dynamic.

The final possibility to consider is where the NAT does not do per-session logging even given the possibility that the remote end is failing to capture source ports. In that case, the port allocation policy proposed in this draft can be used. The impact on traceability is that the investigating authority would be able to collect only the list of all internal addresses mapped to a given public address during the period of time concerned. This has an

impact on privacy as well as traceability, depending on the follow-up actions taken by the investigating authority to achieve its objectives.

4. Other Considerations

[I-D.ietf-intarea-shared-addressing-issues] notes several issues introduced by the use of dynamic as opposed to static port assignment. For example, Section 12.2 of that document notes the effect on authentication procedures. These issues must be resolved, but are not specific to the port allocation policy described in this document.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

The security considerations applicable to NAT operation for various protocols as documented in, for example, [RFC4787] and [RFC5382] also apply to this proposal.

7. Acknowledgements

Mohamed Boucadair reviewed the initial document and provided useful comments to improve it. Reinaldo Penno, Joel Jaegli, and Dan Wing provided comments on the subsequent version that resulted in major revisions.

8. Appendix A: Configure Server Software to Log Source Port

8.1. Apache

The user can use LogFormat command to define a customized log format and use CustomLog command to apply that log format. "%a" and "%{remote}p" can be used in the format string to require logging the client's IP address and source port respectively. This feature is available since Apache version 2.1.

Detail configuration guide can be found at [APACHE_LOG_CONFIG].

8.2. Postfix

In order to log the client source port, macro `smtpd_client_port_logging` should be set to "yes" in the configuration file. [POSTFIX_LOG_CONFIG]

This feature is available since Postfix version 2.5.

8.3. Sendmail

Sendmail has a macro `${client_port}` storing the client port. To log the source port, the user can define some check rules. Here is an example which should be in the `.mc` configuration macro[SENDMAIL_LOG_CONFIG]:

```
LOCAL_CONFIG
Klog syslog
```

```
LOCAL_RULESETS
SLocal_check_mail
R $* $@ $(log Port_Stat $&{client_addr} $&{client_port} $)
```

This feature is available since version 8.10.

8.4. sshd

sshd supports logging the client IP address and client port when a client starts connection since version 1.2.2, here is the source code in `sshd.c`:

```
...
verbose("Connection from %.500s port %d", remote_ip, remote_port);
...
```

sshd supports logging the client IP address when a client disconnect from version 1.2.2 to version 5.0. since version 5.1 sshd supports logging the client IP address and source port. Here is the source code in `sshd.c`:

```
...
/* from version 1.2.2 to 5.0*/
verbose("Closing connection to %.100s", remote_ip);
...

/* since version 5.1*/
verbose("Closing connection to %.500s port %d",
remote_ip, remote_port);
```

In order to log the source port, the LogLevel should be set to VERBOSE [SSHD_LOG_CONFIG] in the configuration file:

```
LogLevel      VERBOSE
```

8.5. Cyrus IMAP and UW IMAP

Cyrus IMAP and UW IMAP do not support logging the source port for the time being. Both software use syslog to create logs; it should not be too difficult to get it supported by adding some new code.

9. References

9.1. Normative References

[I-D.bajko-pripaddrassign]
Bajko, G., Savolainen, T., Boucadair, M., and P. Levis,
"Port Restricted IP Address Assignment (Work in
progress)", October 2009.

[I-D.ietf-intarea-shared-addressing-issues]
Ford, M., Boucadair, M., Durand, A., Levis, P., and P.
Roberts, "Issues with IP Address Sharing (Work in
progress)", June 2010.

9.2. Informative References

[APACHE_LOG_CONFIG]
"http://httpd.apache.org/docs/2.1/mod/
mod_log_config.html".

[I-D.nishitani-cgn]
Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida,
"Common requirements for IP address sharing schemes (Work
in progress)", July 2010.

[I-D.softwire-dual-stack-lite]
Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-
Stack Lite Broadband Deployments Following IPv4 Exhaustion
(Work in progress)", July 2010.

[POSTFIX_LOG_CONFIG]
"http://www.postfix.org/postconf.5.html".

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation
(NAT) Behavioral Requirements for Unicast UDP", BCP 127,
RFC 4787, January 2007.

[RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

[SENDMAIL_LOG_CONFIG] O'Reilly, "Sendmail, 3rd Edition, Page 798", December 2002.

[SSHD_LOG_CONFIG] "http://www.openbsd.org/cgi-bin/man.cgi?query=sshd_config&sektion=5".

Authors' Addresses

Tina Tsou (editor)
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Phone:
Email: tena@huawei.com

Weibo Li (editor)
China Telecom
109, Zhongshan Ave. West, Tianhe District
Guangzhou 510630
P.R. China

Phone:
Email: mweiboli@gmail.com

Tom Taylor
Huawei Technologies
1852 Lorraine Ave.
Ottawa K1H 6Z8
Canada

Phone:
Email: tom111.taylor@bell.net

Behavior Engineering for Hindrance
Avoidance
Internet-Draft
Intended status: Informational
Expires: May 5, 2016

T. Tsou
Huawei Technologies (USA)
W. Li
China Telecom
T. Taylor
J. Huang
Huawei Technologies
November 2, 2015

Port Management To Reduce Logging In Large-Scale NATs
draft-tsou-behave-natx4-log-reduction-06

Abstract

Various IPv6 transition strategies require the introduction of large-scale NATs (e.g. AFTR, NAT64) to share the limited supply of IPv4 addresses available in the network until transition is complete. There has recently been debate over how to manage the sharing of ports between different subscribers sharing the same IPv4 address. One factor in the discussion is the operational requirement to log the assignment of transport addresses to subscribers. It has been argued that dynamic assignment of individual ports between subscribers requires the generation of an excessive volume of logs. This document suggests a way to achieve dynamic port sharing while keeping log volumes low.

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 May 5, 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	3
2. A Suggested Solution	3
3. Issues Of Traceability	5
4. Other Considerations	6
5. IANA Considerations	6
6. Security Considerations	6
7. Acknowledgements	7
8. Appendix A: Configure Server Software to Log Source Port	7
8.1. Apache	7
8.2. Postfix	8
8.3. Sendmail	8
8.4. sshd	8
8.5. Cyrus IMAP and UW IMAP	9
9. Informative References	9
Authors' Addresses	10

1. Introduction

During the IPv6 transition period, some large-scale NAT devices may be introduced, e.g. DS-Lite AFTR, NAT64. When a NAT device needs to set up a new connection for a given internal address behind the NAT, it needs to create a new mapping entry for the new connection, which will contain source IP address, source port or ICMP identifier, converted source IP address, converted source port, protocol (TCP/UDP), etc.

Due to legislation and law enforcement requirement, sometimes it is necessary to log these mapping for a period of time, such as 6 months. The mapping information is highly privacy sensitive, if possible, it should be deleted as soon as possible. Some high performance NAT devices may need to create a large amount of new sessions per second. If logs are generated for each mapping entry, the log traffic could reach tens of megabytes per second or more, which would be a problem for log generation, transmission and storage. According to a test done by [I-D.ietf-sunset4-nat64-port-allocation], in a network with 20,000 subscribers, over 60 days period, the raw log size can reach 42.5TB if it is based on per-session log, while the log size will be 40.6 GB if it is based on port blocks. Although compression technologies can be used before the log storage, the log size is still big.

[RFC6888], REQ-13 suggest "maximize port utilization", REQ-14 suggest "minimize log volume". However, it is difficult to achieve both, there will be a tradeoff between the efficiency with which ports are used and the rate of generation of log records.

2. A Suggested Solution

This document proposes a solution that allows dynamic sharing of port ranges between users while minimizing the number of logs that have to be generated. Briefly, ports are allocated to the user in blocks. Logs are generated only when blocks are allocated or deallocated. This provides the necessary traceability while reducing log generation by a factor equal to the block size, as compared with fully dynamic port allocation.

Here is how the proposal would work in greater detail. When the user sends out the first packet, a port resource pool is allocated for the user, e.g., assigning ports 2001~2300 of a public IP address to the user's resource pool. Only one log should be generated for this port block. When the NAT needs to set up a new mapping entry for the user, it can use a port in the user's resource pool and the corresponding public IP address. If the user needs more port

resources, the NAT can allocate another port block, e.g., ports 3501~3800, to the user's resource pool. Again, just one log needs to be generated for this port block.

[RFC6431] takes this idea further by allocating non-contiguous sets of ports using a pseudorandom function. Scattering the allocated ports in this way provides a modest barrier to port guessing attacks. The use of randomization is discussed further in Section 6.

Suppose now that a given internal address has been assigned more than one block of ports. The individual sessions using ports within a port block will start and end at different times. If no ports in some port block are used for some configurable time, the NAT can remove the port block from the resource pool allocated to a given internal address, and make it available for other users. In theory, it is unnecessary to log deallocations of blocks of ports, because the ports in deallocated blocks will not be used again until the blocks are reallocated. However, the deallocation may be logged when it occurs add robustness to troubleshooting or other procedures.

The deallocation procedure presents a number of difficulties in practice. The first problem is the choice of timeout value for the block. If idle timers are applied for the individual mappings (sessions) within the block, and these conform to the recommendations for NAT behaviour for the protocol concerned, then the additional time that might be configured as a guard for the block as a whole need not be more than a few minutes. The block timer in this case serves only as a slightly more conservative extension of the individual session idle timers. If, instead, a single idle timer is used for the whole block, it must itself conform to the recommendations for the protocol with which that block of ports is associated. For example, REQ-5 of [RFC5382] requires an idle timer expiry duration of at least 2 hours and 4 minutes for TCP.

The next issue with port block deallocation is the conflict between the desire to randomize port allocation and the desire to make unused resources available to other internal addresses. As mentioned above, ideally port selection will take place over the entire set of blocks allocated to the internal address. However, taken to its fullest extent, such a policy will minimize the probability that all ports in any given block are idle long enough for it to be released.

As an alternative, it is suggested that when choosing which block to select a port from, the NAT should omit from its range of choice the block that has been idle the longest, unless no ports are available in any of the other blocks. The expression "block that has been idle the longest" designates the block in which the time since the last packet was observed in any of its sessions, in either direction, is

earlier than the corresponding time in any of the other blocks assigned to that internal address.

3. Issues Of Traceability

Section 11 of [RFC6269] provides a good discussion of the traceability issue. Complete traceability given the NAT logging practices proposed in this draft requires that the remote destination record the source port of a request along with the source address (and presumably protocol, if not implicit). In addition, the logs at each end must be timestamped, and the clocks must be synchronized within a certain degree of accuracy. Here is one reason for the guard timing on block release, to increase the tolerable level of clock skew between the two ends.

The ability to configure various server applications to record source ports has been investigated, with the following results:

- o Source port recording can be configured in Apache, Postfix, sendmail and sshd. Please refer to the appendix for configuration guide.
- o Source port recording is not supported by IIS, Cyrus IMAP and UW IMAP. But it should not be too difficult to get Cyrus IMAP and UW IMAP to support it by modifying the source code.

Where source port logging can be enabled, this memo strongly urges the operators to do so. Similarly, intrusion detection systems should capture source port as well as source address of suspect packets.

In some cases [RFC6269], a server may not record the source port of a connection. To allow traceability, the NAT device needs to record the destination IP address of a connection. As [RFC6269] points out, this will provide an incomplete solution to the issue of traceability because multiple users of the same shared public IP address may access the service at the same time. From the point of view of this draft, in such situations the game is lost, so to speak, and port allocation at the NAT might as well be completely dynamic.

The final possibility to consider is where the NAT does not do per-session logging even given the possibility that the remote end is failing to capture source ports. In that case, the port allocation policy proposed in this draft can be used. The impact on traceability is that analysis of the logs would yield only the list of all internal addresses mapped to a given public address during the period of time concerned. This has an impact on privacy as well as

traceability, depending on the follow-up actions taken.

4. Other Considerations

[RFC6269] notes several issues introduced by the use of dynamic as opposed to static port assignment. For example, Section 12.2 of that document notes the effect on authentication procedures. These issues must be resolved, but are not specific to the port allocation policy described in this document.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

The discussion which follows addresses an issue that is particularly relevant to the proposal made in this document. The security considerations applicable to NAT operation for various protocols as documented in, for example, [RFC4787] and [RFC5382] also apply to this proposal.

[RFC6056] summarizes the TCP port-guessing attack, by means of which an attacker can hijack one end of a TCP connection. One mitigating measure is to make the source port number used for a TCP connection less predictable. [RFC6056] provides various algorithms for this purpose.

As Section 3.1 of that RFC notes: "...provided adequate algorithms are in use, the larger the range from which ephemeral ports are selected, the smaller the chances of an attacker are to guess the selected port number." Conversely, the reduced range sizes proposed by the present document increase the attacker's chances of guessing correctly. This result cannot be totally avoided. However, mitigating measures to improve this situation can be taken both at port block assignment time and when selecting individual ports from the blocks that have been allocated to a given user.

At assignment time, one possibility is to assign ports as non-contiguous sets of values as proposed in [RFC6431]. However, this approach creates a lot of complexity for operations, and the pseudo randomization can create uncertainty when the accuracy of logs is important to protect someone's life or liberty.

Alternatively, the NAT can assign blocks of contiguous ports.

However, at assignment time the NAT could attempt to randomize its choice of which of the available idle blocks it would assign to a given user. This strategy has to be traded off against the desirability of minimizing the chance of conflict between what [RFC6056] calls "transport protocol instances" by assigning the most-idle block, as suggested in Section 2. A compromise policy might be to assign blocks only if they have been idle for a certain amount of time whenever possible, and select pseudorandomly between the blocks available according to this criterion. In this case it is suggested that the time value used be greater than the guard timing mentioned in Section 2, and that no block should ever be reassigned until it has been idle at least for the duration given by the guard timer.

While the block assignment strategy can provide some mitigation of the port guessing attack, the largest contribution will come from pseudo randomization at port selection time. [RFC6056] provides a number of algorithms for achieving this pseudorandomization. When the available ports are contained in blocks which are not in general consecutive, the algorithms clearly need some adaptation. The task is complicated by the fact that the number of blocks allocated to the user may vary over time. Adaptation is left as an exercise for the implementor.

7. Acknowledgements

Mohamed Boucadair reviewed the initial document and provided useful comments to improve it. Reinaldo Penno, Joel Jaeggli, and Dan Wing provided comments on the subsequent version that resulted in major revisions. Serafim Petsis provided encouragement to publication after a hiatus of two years.

The authors are grateful to Dan Wing for his help in moving this document forward, and in particular for his helpful comments on its content.

8. Appendix A: Configure Server Software to Log Source Port

8.1. Apache

The user can use LogFormat command to define a customized log format and use CustomLog command to apply that log format. "%a" and "%{remote}p" can be used in the format string to require logging the client's IP address and source port respectively. This feature is available since Apache version 2.1.

A detailed configuration guide can be found at [APACHE_LOG_CONFIG].

8.2. Postfix

In order to log the client source port, macro `smtpd_client_port_logging` should be set to "yes" in the configuration file. `[POSTFIX_LOG_CONFIG]`

This feature is available since Postfix version 2.5.

8.3. Sendmail

Sendmail has a macro `#{client_port}` storing the client port. To log the source port, the user can define some check rules. Here is an example which should be in the `.mc` configuration macro `[SENDMAIL_LOG_CONFIG]`:

```
LOCAL_CONFIG
Klog syslog
```

```
LOCAL_RULESETS
SLocal_check_mail
R $* $@ $(log Port_Stat ${client_addr} ${client_port} $)
```

This feature is available since version 8.10.

8.4. sshd

`SSHD_CONFIG(5)` OpenBSD Programmer's Manual `SSHD_CONFIG(5)` NAME
`sshd_config` - OpenSSH SSH daemon configuration file `LogLevel` Gives the verbosity level that is used when logging messages from `sshd(8)`. The possible values are: `QUIET`, `FATAL`, `ERROR`, `INFO`, `VERBOSE`, `DEBUG`, `DEBUG1`, `DEBUG2`, and `DEBUG3`. The default is `INFO`. `DEBUG` and `DEBUG1` are equivalent. `DEBUG2` and `DEBUG3` each specify higher levels of debugging output. Logging with a `DEBUG` level violates the privacy of users and is not recommended. `SyslogFacility` Gives the facility code that is used when logging messages from `sshd(8)`. The possible values are: `DAEMON`, `USER`, `AUTH`, `LOCAL0`, `LOCAL1`, `LOCAL2`, `LOCAL3`, `LOCAL4`, `LOCAL5`, `LOCAL6`, `LOCAL7`. The default is `AUTH`.

`sshd` supports logging the client IP address and client port when a client starts connection since version 1.2.2, here is the source code in `sshd.c`:

```
...
verbose("Connection from %.500s port %d", remote_ip, remote_port);
...
```

`sshd` supports logging the client IP address when a client disconnects, from version 1.2.2 to version 5.0. Since version 5.1

sshd supports logging the client IP address and source port. Here is the source code in sshd.c:

```
...
/* from version 1.2.2 to 5.0*/
verbose("Closing connection to %.100s", remote_ip);
...

/* since version 5.1*/
verbose("Closing connection to %.500s port %d",
remote_ip, remote_port);
```

In order to log the source port, the LogLevel should be set to VERBOSE [SSHD_LOG_CONFIG] in the configuration file:

```
LogLevel      VERBOSE
```

8.5. Cyrus IMAP and UW IMAP

Cyrus IMAP and UW IMAP do not support logging the source port for the time being. Both software use syslog to create logs; it should not be too difficult to get it supported by adding some new code.

9. Informative References

- [APACHE_LOG_CONFIG]
The Apache Software Foundation,
"http://httpd.apache.org/docs/2.4/mod/
mod_log_config.html", 2013.
- [I-D.ietf-sunset4-nat64-port-allocation]
Chen, G., Li, W., Tsou, T., Huang, J., Taylor, T., and J.
Tremblay, "Analysis of NAT64 Port Allocation Methods for
Shared IPv4 Addresses",
draft-ietf-sunset4-nat64-port-allocation-01 (work in
progress), July 2015.
- [POSTFIX_LOG_CONFIG]
"http://www.postfix.org/postconf.5.html", 2013.
- [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>>.
- [RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P.
Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142,

- RFC 5382, DOI 10.17487/RFC5382, October 2008,
<<http://www.rfc-editor.org/info/rfc5382>>.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011,
<<http://www.rfc-editor.org/info/rfc6056>>.
- [RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011,
<<http://www.rfc-editor.org/info/rfc6269>>.
- [RFC6431] Boucadair, M., Levis, P., Bajko, G., Savolainen, T., and T. Tsou, "Huawei Port Range Configuration Options for PPP IP Control Protocol (IPCP)", RFC 6431, DOI 10.17487/RFC6431, November 2011,
<<http://www.rfc-editor.org/info/rfc6431>>.
- [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>>.
- [SENDMAIL_LOG_CONFIG]
O'Reilly, "Sendmail, 3rd Edition, Page 798",
December 2002.
- [SSHD_LOG_CONFIG]
"http://www.openbsd.org/cgi-bin/man.cgi?query=sshd_config&sektion=5", April 2013.

Authors' Addresses

Tina Tsou
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1 408 330 4424
Email: tina.tsou.zouting@huawei.com

Weibo Li
China Telecom
109, Zhongshan Ave. West, Tianhe District
Guangzhou 510630
P.R. China

Phone:
Email: mweiboli@gmail.com

Tom Taylor
Huawei Technologies
Ottawa
Canada

Phone:
Email: tom.taylor.stds@gmail.com

James Huang
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Phone:
Email: James.huang@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 23, 2011

X. Xu
Huawei Technologies Co.,Ltd
M. Boucadair
France Telecom
Y. Lee
Comcast
G. Chen
China Mobile
October 20, 2010

Redundancy Requirements and Framework for Stateful Network Address
Translators (NAT)
draft-xu-behave-stateful-nat-standby-06

Abstract

This document defines a set of requirements and a framework for ensuring redundancy for stateful Network Address Translators (NAT), including NAT44, NAT64 and NAT46.

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 RFC 2119 [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 April 23, 2011.

Copyright Notice

Copyright (c) 2010 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. Terminology and Acronyms	3
2.1. Acronyms	3
2.2. Terminology	4
3. Reference Architecture	5
4. Dynamic and Static States	6
5. Overview of Redundancy Mechanisms	6
6. Cold Standby Mode	8
6.1. Internal Realm	8
6.2. External Realm	8
6.3. NAT Reachability Announcement	9
7. Hot Standby Mode	10
7.1. Internal Realm	10
7.2. External Realm	10
7.3. NAT Reachability Announcement	10
8. IANA Considerations	11
9. Security Considerations	11
10. Acknowledgements	11
11. References	11
11.1. Normative References	11
11.2. Informative References	11
Appendix A. State Synchronization Protocol Considerations	12
Appendix B. Election Protocol Considerations	13
Authors' Addresses	14

1. Introduction

Network Address Translation (NAT) has been used as an efficient way to share the same IPv4 address among several hosts. Recently, due to IPv4 address shortage, several proposals have been elaborated to rely on Carrier Grade NAT (CGN, a.k.a.- LSN for Large Scale NAT) (e.g., [I-D.shirasaki-nat444-isp-shared-addr], [I-D.ietf-softwire-dual-stack-lite] and [I-D.ietf-behave-v6v4-xlate-stateful]). In such models, CGN function (which may be embedded in a router or be deployed in standalone devices) is deployed within large-scale networks, such as ISP networks or enterprise ones, where a large number of customers are located. These customers within a network which is served by a single CGN device may experience service degradation due to the presence of the single point of failure or loss of state information. Therefore, redundancy capabilities of the CGN devices are strongly desired in order to deliver highly available services to customers. Failure detection and repair time must be therefore shortened.

This document describes a framework of redundancy for stateful NAT including: NAT44 including DS-Lite, NAT64 and NAT46.

The main purpose of this document is to analyze means to ensure high availability in environments where carrier grade NAT44, NAT64 and NAT46 are deployed. Some engineering recommendations are provided for the selection of the IPv6 prefix to build IPv4-Embedded IPv6 addresses [I-D.ietf-behave-address-format] and the routing configuration.

Except dealing with the exceptional failures (e.g., power outage, OS crash-down or link failure, etc.), the redundancy mechanism described in this document can also be used for planned maintenance operations (i.e., graceful shutdown of the Primary NAT due to maintenance needs).

Unless otherwise mentioned, NAT and CGN terms throughout this document, pertain to stateful NAT and stateful CGN. Stateless NAT is out of the scope of this document.

2. Terminology and Acronyms

2.1. Acronyms

CGN	Carrier Grade NAT
LSN	Large Scale NAT
DS-Lite	Dual Stack Lite
AFTR	Address Family Transition Router
NAT	Network Address Translation
ISP	Internet Service Provider

2.2. Terminology

This document makes use of the terms defined in [RFC2663]. Below are provided terms specific to this document:

- o CGN (Carrier Grade NAT) or LSN (Large Scale NAT): a NAT device placed within a large-scale network (e.g., ISP network, enterprise network, or campus network). CGN may be placed at the boundary between the large-scale private network and the public Internet, between a private network and a large-scale public network or between two heterogeneous IP realms (i.e., IPv4 and IPv6).
- o CGN internal address realm (internal realm for short): a realm internal to the CGN.

For NAT44, the internal realm refers to the private networks.

For NAT64, the internal realm means IPv6 network or IPv6 Internet.

For NAT46, the internal realm refers to IPv4 network or IPv4 Internet.

For DS-Lite, the internal address realm is assumed to be private IPv4 addresses even if the transport mode used to convey exchanged traffic is IPv6. A DS-Lite CGN device (a.k.a., Address Family Transition Router) is a special NAT44 function which uses the IPv6 address as a means to de-multiplex users sharing the same IPv4 address [I-D.ietf-software-dual-stack-lite].

- o The hosts located in the internal realm are called internal hosts, and the addresses used in the internal realm are called internal addresses.
- o CGN external address realm (external realm for short): a realm external to the CGN.

For NAT44, the external realm refers to the IPv4 Internet.

For NAT64, the external realm means the IPv4 Internet or IPv4 network.

For NAT46, the external realm refers to the IPv6 Internet or IPv6 network.

- o The hosts located in the external realm are called external hosts, and the addresses used in the external realm are called external addresses.
- o Internal address pool: an address pool used for assigning internal addresses to represent the external hosts in the internal realm. This address pool is specific to NAT46 and NAT64.

For NAT46, the CGN will allocate one internal address (which is an IPv4 address) from the pool to an external IPv6 host and map the external IPv6 host's IPv6 address to this IPv4 address.

For NAT64, the CGN internal address pool is the Prefix64 [I-D.ietf-behave-address-format]. Prefix64 is used for synthesizing internal IPv6 addresses to represent external IPv4 hosts in the internal realm.

- o External address pool: an address pool used by the CGN for assigning external addresses to the internal hosts.

For NAT44 and NAT64, the external address pool contains a set of public IPv4 addresses.

For NAT46, the external IPv6 address pool is the Prefix64. Prefix64 is used by the CGN for synthesizing the external IPv6 addresses to represent internal IPv4 hosts in the external realm.

- o CPE (Customer Premises Equipment): a device which is used to interconnect the customer premise with the service provider's network.

3. Reference Architecture

In a typical operational scenario, as illustrated in Figure 1, two NAT devices are deployed for redundancy purposes. This is the reference architecture for the mechanisms we describe in this memo. Note that these mechanisms are also suitable in scenarios where more than two NAT devices are used.

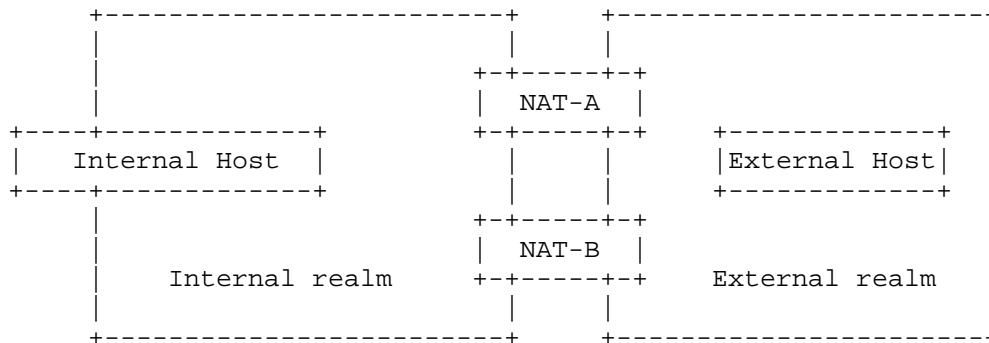


Figure 1: General Scenario of Dual NAT Routers

The redundancy mechanisms for NAT44, NAT46 and NAT64 are almost identical. In all cases, the NAT device or the immediate router of the NAT device announces the reachability of the NAT device to the external realm. The slight difference is the NAT reachability information. For example, NAT64 announces an IPv6 route for the Prefix64; NAT44 announces an IPv4 default route; DS-Lite AFTR announces an IPv6 route pointing to itself; and NAT46 announces a route for its internal address pool. This difference does not affect the general redundancy mechanisms, so the mechanisms described in this memo can be applied to NAT44, NAT64 and NAT46 devices.

4. Dynamic and Static States

The NAT states mentioned in this document only mean those NAT states which are created dynamically by outgoing packets, rather than those static NAT states which are configured manually or with automatic means such as UPnP or PCP. For those static NAT states (a.k.a., port forwarding entries), they are essentially part of the configuration data.

Port forwarding entries SHOULD be stored in permanent storage whatever the deployed redundancy mode.

5. Overview of Redundancy Mechanisms

The fundamental principle of NAT redundancy is to make two or more NAT devices function as a redundancy group, and select one as the Primary NAT and the other(s) as the Backup NAT through a dedicated election procedure or manual configuration.

In the nominal regime, traffic exchanged between one host in the

internal realm and another host in the external realm is handled by the Primary NAT. Once the Primary NAT is out of service, the Backup NAT with the highest priority (if several Backup NAT devices are deployed) takes over and provides the NAT services to the internal hosts. This Backup NAT is then identified as new Primary NAT. Once the former Primary NAT became operational, it could either preempt the role of Primary NAT or stay as a candidate in the redundancy group. This is part of administrative policies and out of scope of this memo.

In order to implement the aforementioned procedure, means to detect and to notify the failure of the Primary NAT to the redundancy group SHOULD be activated.

To ensure a coherent behavior when NAT device fails, this document assumes that both Primary and Backup NAT devices are managed by the same administrative domain. Thus, consistent configuration policies SHOULD be enforced in all devices. Note that the election process MUST be deterministic and does not lead to ambiguous situation where two or more NAT devices become Primary NAT. Moreover, the failover SHOULD be quick to ensure service continuity and keep end-users from perceiving service unavailability.

Three redundancy modes are described hereafter: the cold standby, the hot standby and the partial hot standby modes:

1. The cold standby mode is simple. The NAT states are not replicated from the Primary NAT to the Backup NAT. When the Primary NAT fails, all the existing established sessions will be flushed out. The internal hosts are required to re-establish sessions to the external hosts;
2. The hot standby mode keeps established sessions while failover happens. NAT states are replicated from the Primary NAT to the Backup NAT. When the Primary NAT fails, the Backup NAT will take over all the existing established sessions. The internal hosts are not required to re-establish sessions to the external hosts.
3. The partial hot standby mode is a flavor of the hot standby mode described above. It is used to avoid replicating NAT states of trivial sessions (e.g., short lifetime sessions) while achieving hot standby for significant sessions (e.g., critical protocols or applications, long lifetime sessions etc.). Criteria for sessions to be replicated on backup NATs SHOULD be explicitly configured on the NAT devices of a redundancy group.

The following sections provide more information about the cold standby and the hot standby modes.

6. Cold Standby Mode

6.1. Internal Realm

The internal addresses used to represent the external hosts in the internal realm SHOULD be retained after the NAT failover. The following assesses how this requirement is met in each NAT flavor:

- o For NAT44 and DS-Lite, the external hosts' internal addresses (i.e., the addresses used to represent the external hosts in the internal realm) are unchanged (i.e., not NAT-ed). Therefore, the above requirement is met without additional work.
- o For NAT64, the NAT devices belonging to a redundancy group SHOULD be configured with an identical Prefix64. Since the NAT64 uses stateless address translation for the external hosts, using the same Prefix64 in the Backup NAT can guarantee the internal hosts to see the same internal addresses for the external hosts.
- o For NAT46, NAT devices in a redundancy group SHOULD be configured with an identical IPv4 address pool. A subset of translation state information SHOULD be synchronized among these NAT devices through a dedicated state synchronization protocol such as [I-D.xu-behave-nat-state-sync]. This translation state ensures that the Backup NAT, once taking over as a Primary NAT, will assign the same IPv4 addresses to the external IPv6 hosts for the internal IPv4 hosts.

6.2. External Realm

Each NAT device in a NAT redundancy group is configured with a different external address pool. A route to that external pool is then announced into the external realm by the NAT device or the NAT immediate router.

- o For NAT44, DS-Lite and NAT64: NAT devices SHOULD be configured with different external IPv4 address pools. These address pools are not overlapped. Otherwise, when the Primary NAT fails and the Backup NAT takes over the Primary NAT, a NAT collision may happen. For example, assuming a Primary NAT NAT-ed internal host Host-A to IPv4-A. IPv4-A is an address which belongs to the external address pool. If the Backup NAT after taking over the primary NAT was configured with the same pool, the Backup NAT MAY assign the same IPv4-A to another internal host Host-B. So, Host-B may receive datagrams originally targeted for Host-A. This might cause confusion to Host-B. In addition, by using different external address pools on each NAT device, incoming datagrams of a given session from the external hosts are ensured to always

traverse through the Backup NAT device after the Primary NAT failover happens.

- o For NAT46, the issue occurred in NAT44 and NAT64 cases will not happen. NAT46 relies on stateless address translation for the internal hosts. The Primary and Backup NAT SHOULD use the same external Prefix64, hence the external hosts can use the Backup NAT46 to reach the internal hosts. In Cold Standby mechanism, the Primary and Backup NAT MAY use different Prefix64s. In contrast, the Primary and Backup NAT in Hot Standby mechanism MUST use an identical Prefix64.

6.3. NAT Reachability Announcement

In order to force the IP datagrams from the internal realm to always traverse through the Primary NAT to the external realm, the Primary NAT SHOULD announce into the internal realm a route towards the external realm.

- o For NAT44, the Primary NAT announces an IPv4 default route into the internal realm.
- o For DS-Lite, the Primary NAT announces a host route into the internal realm.
- o For NAT64, the Primary NAT announces a route for the Prefix64 into the internal realm.
- o For NAT46, the Primary NAT announces a route for the internal address pool into the internal realm (If the internal address pool can be aggregated to one prefix).

The Primary NAT SHOULD attempt to withdraw its previously announced routes when it ceases the Primary role due to pre-configured conditions, e.g.- it loses the IP connectivity to the external realm.

When the Primary NAT fails and the Backup NAT takes over, datagrams from the internal hosts destined for the external realm SHOULD pass through the Backup NAT. Hence, when the Backup NAT is manually configured to switch over to become the Primary NAT, the Backup NAT (or associated router) SHOULD announce the same route into the internal realm, but the routing cost of this route MUST be set to a higher value than the route announced by the Primary NAT.

Alternatively, the Primary NAT announces several more specific routes into the internal realm while the Backup NAT announces an aggregate route. Taken the NAT46 as an example, assuming the internal address pool is 10.0.0.0/8, the Primary NAT announces two more specific

routes to 10.0.0.0/9 and 10.128.0.0/9 respectively while the Backup NAT announces an aggregate route to 10.0.0.0/8. In case the Primary NAT and the Backup NAT are automatically elected through a dedicated election process, the Backup NAT would be elected as a new Primary NAT once the old Primary one fails, so it is not necessary for the Backup NAT to make the above route announcements until it is elected as a new Primary NAT.

In order for the external hosts to traverse through the NAT to reach the internal hosts, the Primary and Backup NAT SHOULD announce a route of its own external address pool into the external realm.

7. Hot Standby Mode

7.1. Internal Realm

The procedure is identical to Section 6.1.

7.2. External Realm

To preserve the established sessions during the failover and to keep the internal addresses unchanged for the external hosts, the external addresses for the internal hosts MUST also be preserved. To preserve the external address of the internal host after NAT-ed, the NAT devices in a redundancy group MUST use an identical external address pool. In addition, they MUST assign the same external address (or address/port pair) to a given internal host.

- o For NAT46, the Primary NAT and Backup NAT MUST use an identical Prefix64.
- o For NAT44, DS-Lite and NAT64, the NAT devices in a redundancy group MUST use the same external address pool and the translation states on the Primary NAT device MUST be synchronized to the Backup NAT(s) in a timely fashion.

7.3. NAT Reachability Announcement

In order to force IP datagrams between the internal realm and the external realm always traverse through the Primary NAT, the Primary NAT (or its associated router) SHOULD announce into the internal realm a route towards the external realm and announce into the external realm a route towards the external address pool.

Once the connectivity to either the external realm or the internal realm is lost, the Primary NAT device itself or a third party SHOULD attempt to withdraw the above routes. If the Primary NAT and the

Backup NAT are specified manually, the Backup NAT device (or its associated router) SHOULD also announce those routes, but with higher enough cost or larger granularity, so as to prepare for the failover.

When the Primary NAT fails, the datagrams towards the external realm will pass through the Backup NAT device. In case the Primary NAT and the Backup are automatically elected through a dedicated election procedure, the Backup NAT would be elected as a new Primary NAT when the old Primary NAT device fails. Consequently, it is not necessary for the Backup NAT to make the above route announcement until it is elected as a new Primary NAT.

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. Security Considerations

TBD.

10. Acknowledgements

The author would like to thank Dan Wing and Dave Thaler for their insightful comments and reviews, and thank Dacheng Zhang and Xuewei Wang for their valuable editorial reviews.

11. References

11.1. Normative References

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

11.2. Informative References

[I-D.ietf-behave-address-format]
 Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-10 (work in progress), August 2010.

- [I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 2010.
- [I-D.ietf-softwire-dual-stack-lite]
Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", draft-ietf-softwire-dual-stack-lite-06 (work in progress), August 2010.
- [I-D.shirasaki-nat444-isp-shared-addr]
Shirasaki, Y., Miyakawa, S., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444 addressing models", draft-shirasaki-nat444-isp-shared-addr-04 (work in progress), July 2010.
- [I-D.xu-behave-nat-state-sync]
Cheng, D. and X. Xu, "NAT State Synchronization Using SCSP", draft-xu-behave-nat-state-sync-02 (work in progress), August 2010.
- [RFC2334] Luciani, J., Armitage, G., Halpern, J., and N. Doraswamy, "Server Cache Synchronization Protocol (SCSP)", RFC 2334, April 1998.
- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.
- [RFC4761] Kompella, K. and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, January 2007.
- [RFC4762] Lasserre, M. and V. Kompella, "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, January 2007.
- [RFC5798] Nadas, S., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, March 2010.

Appendix A. State Synchronization Protocol Considerations

[I-D.xu-behave-nat-state-sync] defines a candidate solution to NAT state synchronization by using Server Cache Synchronization Protocol

(SCSP) [RFC2334]. For more information about the proposed solution, refer to [I-D.xu-behave-nat-state-sync].

[[Note: What to do with this section?]]

Appendix B. Election Protocol Considerations

[[Note: What to do with this section?]]

An election process and associated protocol(s) can be used to automatically elect one NAT device among a NAT redundancy group as the Primary NAT and the others as Backup NATs. Once the Primary NAT fails, the Backup NAT with the highest priority SHOULD take over the Primary NAT role after a short delay. The election protocol is also used to track the connectivity to the external realm and the internal realm. Once connections to the external realm or the internal realm lost, the NAT device is not qualified to be the Primary NAT and it will withdraw the route towards the external realm announced previously. In the case of hot standby, it SHOULD also withdraw the route towards the external address pool.

As an implementation example, VRRP [RFC5798] can be used as the automatic election protocol. In addition, an interface tracking mechanism can also be used to adjust the priority to influence the election results.

If two NAT devices are directly connected via an Ethernet network, VRRP can run directly on the Ethernet interfaces. Otherwise, some extra configuration or protocol changes need to be implemented. One option is to create conditions for VRRP to run among these devices. For example, to create a VPLS [RFC4761][RFC4762] instance and enable IP functions and run VRRP on those VLAN interfaces which are bound to that VPLS instance. If enabling IP on those interfaces is not supported, the following trick to realize the same goal, but at a cost of consuming two physical interfaces on each NAT router: create a VPLS instance among a set of NAT devices, and on each of them one Ethernet interface is bound to that VPLS instance, and another IP-enabled Ethernet interface is locally connected with that interface. Then VRRP can run on those IP enabled Ethernet interfaces which are all connected to that VPLS instance. Another option is to extend VRRP so that VRRP neighbors can be specified manually and VRRP messages can be exchanged directly among VRRP neighbors in unicast.

VRRP is only an implementation example of the election process. Other protocols MAY be used to manage the roles of Primary and Backup.

Authors' Addresses

Xiaohu Xu
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: xuxh@huawei.com

Mohamed Boucadair
France Telecom
Rennes
France

Email: mohamed.boucadair@orange-ftgroup.com

Yiu L. Lee
Comcast

Email: yiu_lee@cable.comcast.com
URI: <http://www.comcast.com>

Gang Chen
China Mobile
53A,Xibianmennei Ave.
Beijing, 100053
P.R.China

Email: phdgang@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 17, 2011

D. Zhang, Ed.
X. Xu, Ed.
Huawei Technologies Co.,Ltd
M. Boucadair, Ed.
France Telecom
X. Wang
Huawei Technologies Co.,Ltd
Y. Wang
CNNIC
C. Byrne
T-Mobile USA
D. Zhang
Huawei Symantec
October 14, 2010

Considerations on NAT64 Load-Balancing
draft-zhang-behave-nat64-load-balancing-00

Abstract

This document investigates several load-balancing approaches for NAT64 devices and analyses the advantages and disadvantages of various prefix selection policies. Both stateless and stateful NAT64 schemes are considered in this document.

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 RFC 2119 [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 April 17, 2011.

Copyright Notice

Copyright (c) 2010 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. Terminology	3
3. Reminder on the Load Balancing Objectives	3
3.1. Inbound Load Balancing	4
3.2. Outbound Load Balancing	5
4. Basic Load Balancing Considerations	5
5. Stateless NAT64	6
5.1. Anycast-based Mode	6
5.2. DHCPv6-based Mode	7
5.3. NAT64 Farm	7
6. Stateful NAT64	7
6.1. Anycast-based Mode	8
6.2. Prefix64 Selection Policy	8
6.2.1. Source-Based Prefix Selection Policy	8
6.2.2. Destination-Based Prefix Selection Policy	9
6.2.3. Round-Robin Prefix Selection Policy	10
6.2.4. Dynamic Prefix Selection Policy	10
6.3. Options for Implementing Load-balancers	11
6.3.1. DNS64 Servers	11
6.3.2. Prefix64 Assigners	11
7. IANA Considerations	13
8. Security Considerations	13
9. Acknowledgements	13
10. Informative References	14
Authors' Addresses	14

1. Introduction

NAT64 devices [I-D.ietf-behave-v6v4-xlate-stateful] are facilities deployed on the boundaries between IPv6 and IPv4 networks to facilitate the communication between IPv6-only clients and IPv4 servers.

This document proposes several load-balancing approaches for NAT64 devices, which can be utilized in the delivery of highly available services, and compares the advantages and disadvantages of various prefix selection policies.

The issues with failover and redundancy are outside the scope of this document. An in depth analysis of those issues is elaborated in [I-D.xu-behave-stateful-nat-standby].

2. Terminology

This document makes use of the terms defined in [I-D.ietf-behave-v6v4-xlate-stateful], [I-D.ietf-behave-dns64] and [I-D.ietf-behave-address-format]. Below are provided the terms specific to this document:

- o Prefix64: an IPv6 prefix used for synthesizing IPv6 addresses representing IPv4 hosts in the IPv6 realm. See [I-D.ietf-behave-address-format] for more details on how IPv4-Embedded IPv6 addresses are built.
- o NAT64-enabled device (or NAT64 device for short): is a device which embeds a NAT64 function as defined in [I-D.ietf-behave-v6v4-xlate-stateful].
- o A load-balancer is a facility which can select a NAT64 device from a set of deployed devices for a given IPv6-only client according to the pre-specified policy. Typically, a load-balancer provisions a client with an IPv6 address of the IPv4-only server that the client is going to access. Prefix64 is elaborately selected by the load-balancer so that the corresponding NAT64 device can intercept the packets between the above communicating parties and correctly process them.

3. Reminder on the Load Balancing Objectives

Load balancing is a technique used by network operators (including service providers, enterprise networks, and etc.) to distribute the load among several ingress/egress points, several paths, several

topologies, etc. In the context of IPv4-IPv6 interconnection, load balancing is mainly motivated by the needs listed below:

- o to optimize the resources usage of deployed NAT64 devices (e.g., several ingress/egress NAT64 devices);
- o to avoid congesting a single NAT64 device while other free resources are still available;
- o to optimize IPv6-IPv4 interconnection costs especially when several NAT64 providers are involved.

Techniques to balance the load among a set of NAT64 devices can be achieved on a load-balancer managing a farm of NAT64 devices, on a single NAT64 device to select the appropriate outbound interface, or it can be implicitly achieved owing to dedicated tweaking operations (e.g., use of anycast-based service, use of distinct Prefix64, etc.).

Note that considerations to balance the traffic between several outbound interfaces of a given NAT64 device are out of scope of this document. Various types of load balancing can be considered as defined in the following sub-sections.

3.1. Inbound Load Balancing

Inbound load balancing means that incoming traffic (i.e., the traffic received from an IPv4-only host) is distributed among a set of NAT64 devices located at the boundary of the IPv4 realm and the IPv6 one.

In a stateful NAT64 case, inbound load balancing cannot be explicitly configured because IPv4-only clients can not initiate sessions to an IPv6-only server (except for IPv6-only hosts which pre-installed static entries in the NAT64 using PCP [I-D.wing-software-port-control-protocol] for instance).

In a stateless NAT64 case, inbound load balancing can be achieved by configuring distinct IPv4 address pools on each stateless NAT64 device. This practice may lead to asymmetric paths (i.e., distinct NAT64 devices will handle the outgoing and the inbound packets) if the same NSP (Network Specific Prefix, [I-D.ietf-behave-address-format]) is provisioned on those NAT64 devices. As an alternative to address this problem, a load balancer needs to be deployed on each side of the NAT64 devices. The balancers need to know the policies of each other so as to work in a cooperative way. Refer to Section 6.2 for more discussion.

3.2. Outbound Load Balancing

Outbound load balancing means the effort of distributing outgoing traffic (i.e., the traffic issued by IPv6-only hosts) among a set of NAT64 devices.

Only this flavor of load balancing is elaborated in the remainder of this document.

4. Basic Load Balancing Considerations

In practice, it is important to guarantee the outgoing traffic and the associated incoming traffic is stuck to a same stateful NAT64 device, so that the NAT64 device can get essential knowledge to correctly process the incoming packets. That is, if a stateful NAT64 device processes a request from an IPv6 client to an IPv4 server, it MUST be able to intercept and process the correspondent reply from the server. To achieve this, distinct external IPv4 addresses SHOULD be configured on each stateful NAT64 device. Otherwise, if the same IPv4 address pool is provisioned on two distinct NAT64 devices (e.g., NAT64-A and NAT64-B) and since the routing paths may not be symmetric, outbound packets may be intercepted by NAT64-A while inbound packets may be received by NAT64-B. NAT64-B will reject the packets it received because no state to process these packets was instantiated beforehand, and thus the communication will fail.

The load balancing SHOULD NOT be solely done based on the traffic load distribution but SHOULD persevere the assignment of the same external IPv4 address for all active sessions initiated by an IPv6-only host. The criteria for distributing customers among a set of NAT64 devices may be implemented during the IPv6 configuration phase of a IPv6-only host or during the processing of actual traffic issued by that host.

In the circumstances where the static mapping entries of an IPv6-only client are pre-installed in a given stateful NAT64 device, the enforced load-balancing technique SHOULD "redirect" the traffic from the client to the NAT64-device where its static mappings are pre-installed.

[[Note: Some dynamic protocols such as PCP may include manes to detect the unavailability of a NAT64 and the re-install the mapping in the new discovered NAT64. But, for the manually configured mappings, the issue is still there.]]

From the operator's perspective, a load balancing solution SHOULD be deterministic, that is, that the actual traffic distribution should

be strictly compliant with what is expected by the system manager. Furthermore, the operations of distributing the load among multiple NAT64 devices SHOULD be covered from end-users. This means that end-users should not be aware of the presence of multiple NAT64 devices in the core network and the selection of the appropriate NAT64 device should not assume any intervention by the customer/host.

When implementing load balancing, load balancing should not lead to (severe) QoS degradation between potential paths. Note that the perceived quality may not only depend on the load balancing technique to distribute the traffic among available path/nodes but it is closely related to the underlying topology (i.e., location of the NAT64 devices, routing metrics configuration, etc.).

An efficient load balancing system SHOULD NOT redirect the traffic to a congested NAT64 device while other NAT64 resources are available. Load indicators (i.e., the data reflecting the load imposed on NAT64 devices) may be disseminated to drive the process of selecting a NAT64 device to handle an ongoing IPv6 packet. These indicators may be based on (almost) real-time measurement tools or based on a traffic logic configured on the load-balancer (e.g., a NAT64 device can handle N IPv6-only hosts).

5. Stateless NAT64

According to [I-D.ietf-behave-address-format], IPv4-Translated and IPv4-Converted IPv6 addresses SHOULD use the same Network Specific Prefix (NSP). To distribute the traffic among a set of stateless NAT64 devices, the alternatives described hereafter can be envisaged. For the anycast-based mode the same Prefix64 is used while for the remaining options, specific Prefix64s are used.

5.1. Anycast-based Mode

The same IPv6 NSP is provisioned to all stateless NAT64 devices; IPv6 hosts are distributed natively among several stateless NAT64 devices. This means that the closest (from a routing perspective) stateless NAT64 device will be used to process IPv6 (resp. IPv4) packets destined to an IPv4 (resp. IPv6) destination.

The efficiency of this mode largely depends on the underlying topology (e.g., location of NAT64 devices) and routing engineering policies. Moreover, a stateless NAT64 device may be overloaded if the routing is not appropriately tuned and/or if the NAT64 devices are not appropriately dimensioned.

The introduction or the removal of NAT64 device(s) can be achieved

without modifying the configuration of DHCPv6 servers. During failure events of a NAT64 system, other NAT64 devices can handle the traffic without any intervention.

5.2. DHCPv6-based Mode

To implement this mode, each stateless NAT64 device is configured with a dedicated NSP. During the IPv6 prefix assignment phase, requesting IPv6 hosts are provided with IPv4-Translated IPv6 prefix using the NSP of the NAT64 device that will be used to handle traffic issued from those hosts and destined to an IPv4 host.

If DHCPv6 is used for the provisioning of IPv6 prefixes, DHCPv6 servers SHOULD be provided with the number of customers to be serviced per dedicated NSP (i.e., an NSP prefix identifies a stateless NAT64 device). Dynamic load information (based on real time monitoring) MAY be provided to the DHCPv6 to drive the process of IPv6 prefix assignment and for better utilization of available NAT64 resources. Furthermore, and for routing optimization purposes and for service stability purpose (e.g., use the same NAT64 device hosting PCP-instructed port forwarding entries), other topological information such as Line-ID SHOULD be used to tag the customers that should be serviced by each NAT64 device.

5.3. NAT64 Farm

An additional scheme would be the deployment of a farm of NAT64 devices with a load-balancer which is responsible for redirecting the traffic to the appropriate NAT64 instance. Unlike stateful NAT64, both IPv4 and IPv6 flows can be load balanced.

In this scenario, the same NSP SHOULD be used for all NAT64 devices belonging to the same farm.

Techniques to distribute the load among the NAT64 devices of the farm are similar to load-balancing techniques among several outbound interfaces of the same NAT64 system.

6. Stateful NAT64

Two variant of the load balancing techniques are elaborated hereafter. Unlike the first mode, anycast-based, the second category requires Prefix64 selection to achieve load balancing. More details are provided below.

[[Note: Add in the next version a comparison between anycast-based approach and the ones based on the Prefix64 selection.]]

6.1. Anycast-based Mode

This mode assumes that the same IPv6 prefix (i.e., NSP or WKP, see [I-D.ietf-behave-address-format]) is provisioned to all deployed stateful NAT64 devices. DNS64 function is provisioned with that prefix used for synthesizing AAAA records.

As stated in Section 4, distinct IPv4 address pools are configured to each NAT64 device. This ensures path symmetry; which means that the same NAT64 device will be used for handling both outbound and inbound packets exchanged in a same stateful conversation between two hosts.

The distribution of the traffic among deployed NAT64 devices is natively achieved relying on the underlying routing configuration. Off-line traffic engineering tools can be used to appropriately tweak the routing metrics so as to allow for acceptable traffic distribution.

The same NAT64 device SHOULD be used to handle all the packets issued by a given IPv6 host so that the same external IPv4 address is used to represent that host in the IPv4 realm. This means that oscillation phenomena induced by underlying routing SHOULD be avoided. By oscillation it is meant that the traffic customer is balanced between two NAT64 devices.

6.2. Prefix64 Selection Policy

It is RECOMMENDED that the functionality of load balancers should be integrated into undedicated servers (e.g., DNS servers, DHCP servers). Therefore, load-balancing can be transparent for IPv6-only hosts. The design options of load balancers are discussed in Section 6.3.

The following sub-sections elaborate on various modes for the prefix selection.

6.2.1. Source-Based Prefix Selection Policy

A source-based prefix selection policy allows a load-balancer to select Prefix64s according to the IPv6 addresses of its IPv6-only clients. For instance, when using a source-based prefix selection policy, the load-balancer in the above example can allocate an IPv6 address with Prefix64-A for the IPv4-only server if the IPv6 address of the client is odd, and Prefix64-B otherwise.

6.2.1.1. Pros

- + It is simple and has enough entropy to ensure reasonable load balancing across different NAT64 devices. 2.
- + The users are consistently assigned to the same NAT64 device for every outbound session. This is important because some applications identify a unique user across multiple transactions using the source IP address; examples include FTP and SSL VPNs. In addition, it is easier for a network management system (NMS) to monitor and manage the activities of a user. For instance, a NMS can collect the information about number of the concurrent sessions initiated by a user from a single NAT64 device. However, when using other policies, a user is not stuck to a NAT64 device, and thus NMS may have to collect such information from multiple NAT64 devices.

6.2.1.2. Cons

- The efficiency of this procedure depends on the selection criteria and may not be deterministic in some cases where the traffic may be redirected to a congested NAT64 device.

6.2.2. Destination-Based Prefix Selection Policy

A destination-based prefix selection policy requires a load-balancer to choose Prefix64s according to the IP addresses of the IPv4 targets. For instance, when using a destination-based prefix selection policy, the load-balancer in the above example can allocate an IPv6 address with Prefix64-A for the IPv4 server if the IPv4 address of the server is odd, and prefix64-B otherwise. In practice, this type of policy can have lots of variations. For instance, when a DNS server is utilized as a load balancer, the server can select a prefix64 according to the hash value of the FQDN of the target server.

6.2.2.1. Pros

- + It is simple to implement;

6.2.2.2. Cons

- A user accessing multiple IPv4 servers may be represented by multiple public IPv4 addresses since its traffic may be processed by different NAT64 systems. This will cause authentication problems in the applications (e.g., FTP and SSL VPNs) which take advantage of the source IP addresses to identify users across different sessions. 2.

- A user can not be redirected to the NAT64 device where it has instructed its port forwarding entries; 3.
- Since there are more users than the servers providing contents, there is not enough entropy to ensure good load balancing. The NAT64 device that services a popular web-site will have to undertake much traffic. It is possible to define some strategies to make major sites evenly assigned to different NAT64s, e.g.- Google to NAT64-A, Facebook to NAT64-B, However, this solution can be onerous and requires heavy human involvement.

6.2.3. Round-Robin Prefix Selection Policy

A round-robin prefix selection policy allows a load-balancer to use various Prefix64s circularly in different requesting sessions. For instance, in the above example, the load-balancer can choose Prefix64-A in the first requesting session, choose Prefix64-B in the second, choose Prefix64-A in the third, choose Prefix64-B in the fourth, and so on.

6.2.3.1. Pros

- + Ensures reasonable distribution among a set of NAT64 devices.

6.2.3.2. Cons

- A given IPv6-only hosts may be redirected to distinct NAT64 devices. Therefore, distinct IPv4 address may be used to represent the IPv6-only host in the IPv4 realm;
- A user can not be redirected to the NAT64 device where it has instructed its port forwarding entries;
- Requires a load balancer (e.g., a DNS64 or a DHCP server) to keep track of the assignments.

6.2.4. Dynamic Prefix Selection Policy

Although the capability of NAT64 devices can be considered as a factor in the designing of the above three types of policies, they are still static and not able to be adjusted according the load changes of NAT64 devices in a timely fashion. If we intend to enable a load-balancer to dynamically modify its policies according to NAT64s' real-time load changes, a dynamic prefix selection policy is necessary. For instance, a DNS64 system or DHCPv6 can use SNMP to collect the information of the overheads (e.g., CPU utilizing rates, free memory amounts, concurrent session numbers, and session numbers per second) imposed on different NAT64-based devices. Based on such

information, the load-balancer can distribute the loads on different NAT64 devices in a more reasonable way.

6.2.4.1. Pros

- + This type of policy can effectively avoid the unbalanced distribution of overheads on different NAT64 devices.

6.2.4.2. Cons

- Such a policy may introduce additional communication and management complexities to a NAT64 device. The complexity depends on the means used to disseminate the NAT64 load.

6.3. Options for Implementing Load-balancers

In practice, the functionality of a load-balancer can be performed by, e.g.- a DNS64 server, a DNS server, a DHCP server, an edge router, or even an IPv6 host itself.

6.3.1. DNS64 Servers

When collaborating with NAT64 devices, a DNS64 server can be solicited by an IPv6-only client to initiate communications to an IPv4-only server identified by a FQDN.

Let us assume there is an IPv6-only client connected to an IPv6 network which attempts to communicate to an IPv4-only server. For the purpose of load balancing, the DNS64 server needs to select a Prefix64 based on one of the prefix selection policies defined in Section 6.2 and use it when synthesizing AAAA RRs. Using the synthesized IPv6 address, the IPv6-only client will take advantage of the NAT64 associated with the Prefix64 to communicate with the IPv4-only server.

When DNS64 is used as a means to load balance the hosts among a group of NAT, DNS64 SHOULD be able to assign the same NAT64 to the same hosts. Means to identify the host SHOULD be supported. This is not natively supported by DNS servers.

A drawback of this mode is that for traffic which does not require a DNS resolution, the packets may flow using a distinct NAT device, and therefore use a distinct external IP address.

6.3.2. Prefix64 Assigners

[I-D.wing-behave-learn-prefix] proposes various solutions for a host in an IPv6-only network to obtain the Prefix64 of a NAT64 device.

With the Prefix64, the hosts can synthesize an appropriate IPv6 address which can route packets to the translator. In the designing of load balancers for NAT64 devices, these approaches are worthwhile to consider.

6.3.2.1. DNS64 Servers

In [I-D.wing-behave-learn-prefix], a NAPTR RR to represent NAT64's Prefix64 is specified. When using DNS servers to act as load balancers for NAT64 devices, multiple NAPTR RRs need to be added the zone file. Every NAPTR RR consists of a Prefix64. Upon receiving a NAPTR query, the DNS server replies the requester with a NAPTR RR according to a pre-specified selection policy. Note that the destination-based prefix selection policy is not applicable in this case because the DNS server may lack the knowledge of the IP address of the queried IPv4 host.

6.3.2.2. DHCPv6 Servers

It is mentioned in [I-D.wing-behave-learn-prefix] that a DHCPv6 server can be used to allocate Prefix64s for hosts, and so a DHCP server has a potential to act as a load balancer for NAT64 devices. Similar with the solution proposed in Section 6.3.2.1, it is difficult for a DHCP server to identify the IP addresses of the IPv4 hosts which its clients intend to communicate with. Therefore, only the source-based policy, the round-robin policy, or the dynamic policy can be used in this approach.

Also, a DHCPv6 server can be adopted to allocate different DNS64 servers for its users in various standard DHCPv6 host configuration processes according to certain selection policies. Unlike the DNS64 servers discussed in Section 6.3.1, in this case a DNS64 server needs to only synthesize AAAA records using a single Prefix64.

The load of NAT devices may be provided to DHCP servers to assist the selection of the DNS64 to be used for new connecting hosts.

6.3.2.3. Default Gateways

[I-D.wing-behave-learn-prefix] also discusses the possibility of using Router Advertisement (RA) messages to transfer Prefix64s for IPv6 users. If the edge router is attached to only one multicast link, no prefix selection policy defined in Section 6.2 can be used. If the edge router is attached to multiple multicast links, the source-based policy, the round-robin policy or the dynamic policy can be used. Because at this phase it is difficult for an edge router to identify the IP addresses of the IPv4 hosts which the IPv6 hosts will communicate with, the destination-based prefix selection policy is

unfeasible.

6.3.2.4. IPv6 Clients

It is possible for an IPv6 host to learn multiple Prefix64s through the approaches defined in [I-D.wing-behave-learn-prefix] and then select one based on a certain prefix selection policy. Such a policy can be the destination-based policy, the source-based policy (only one prefix64 is used), the round-robin policy or the dynamic policy.

This solution is not deterministic and can lead to congesting a given NAT64 device.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

As mentioned previously, all the traffic between an IPv6 host and an IPv4 host should be intercepted and processed by a same NAT64 device. However, when using certain policies (e.g., the destination-based prefix selection policy and the Round-Robin prefix selection policy), this requirement cannot be fulfilled. The traffic of a user will be distributed to different NAT64 devices. Under such a circumstance, it may be difficult for network management systems to collect information from different NAT64 devices in order to monitor users' behavior in a real in time fashion. In addition, it can be difficult for intrusion detection/prevision systems to combine the operations of a user so as to reason whether she is trying to perform a multi-step attack.

Another security concern is load balancers. Because load balancers play an important role in distributing traffic to different NAT64 devices, the communication between users and load balancers should be secured. Otherwise, attackers may disturb load balancing and carry out DDOS attacks by modifying the packets sent from load balancers.

9. Acknowledgements

TBC.

10. Informative References

- [I-D.ietf-behave-address-format]
 Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-10 (work in progress), August 2010.
- [I-D.ietf-behave-dns64]
 Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-11 (work in progress), October 2010.
- [I-D.ietf-behave-v6v4-xlate-stateful]
 Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 2010.
- [I-D.wing-behave-learn-prefix]
 Wing, D., "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-04 (work in progress), October 2009.
- [I-D.wing-software-port-control-protocol]
 Wing, D., Penno, R., and M. Boucadair, "Pinhole Control Protocol (PCP)", draft-wing-software-port-control-protocol-02 (work in progress), July 2010.
- [I-D.xu-behave-stateful-nat-standby]
 Xu, X., "Redundancy and Load Balancing Framework for Stateful Network Address Translators (NAT)", draft-xu-behave-stateful-nat-standby-05 (work in progress), September 2010.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Dacheng Zhang (editor)
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: zhangdacheng@huawei.com

Xiaohu Xu (editor)
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: xuxh@huawei.com

Mohamed Boucadair (editor)
France Telecom
Rennes
France

Email: mohamed.boucadair@orange-ftgroup.com

Xuewei Wang
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: wangxuewei@huawei.com

Yan Wang
CNNIC
No.4 South 4th Street,
Beijing, Zhongguancun 100190
P. R. China

Email: wangyan-lab@cnnic.cn

Cameron Byrne
T-Mobile USA
3617 131st Ave SE
Bellevue, WA 98006
US

Email: cameron.byrne@t-mobile.com

Dong Zhang
Huawei Symantec
KuiKe Building, No.9 Xinxu Rd.,
Beijing, Hai-Dian District 100085
P. R. China

Email: zhangdong_rh@huaweisymantec.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 12, 2012

D. Zhang
X. Xu
Huawei Technologies Co.,Ltd
M. Boucadair
France Telecom
July 11, 2011

Considerations on NAT64 Load-Balancing
draft-zhang-behave-nat64-load-balancing-03

Abstract

This document investigates several load-balancing approaches for NAT64 devices and analyzes the advantages and disadvantages of various prefix selection policies. Both stateless and stateful NAT64 schemes are considered in this document.

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 RFC 2119 [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 12, 2012.

Copyright Notice

Copyright (c) 2011 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. Terminology	3
3. Reminder on the Load Balancing Objectives	3
3.1. Inbound Load Balancing	4
3.2. Outbound Load Balancing	5
4. Basic Load Balancing Considerations	5
5. Stateless NAT64	6
5.1. Anycast-based Mode	6
5.2. DHCPv6-based Mode	7
5.3. NAT64 Farm	7
6. Stateful NAT64	7
6.1. Anycast-based Mode	8
6.2. Prefix64 Selection Policy	8
6.2.1. Source-Based Prefix Selection Policy	8
6.2.2. Destination-Based Prefix Selection Policy	9
6.2.3. Round-Robin Prefix Selection Policy	10
6.2.4. Dynamic Prefix Selection Policy	10
6.3. Options for Implementing Load-balancers	11
6.3.1. DNS64 Servers	11
6.3.2. Prefix64 Assigners	12
7. IANA Considerations	13
8. Security Considerations	13
9. Contributors	14
10. References	15
10.1. Normative References	15
10.2. Informative References	15
Authors' Addresses	15

1. Introduction

NAT64 devices [I-D.ietf-behave-v6v4-xlate-stateful] are facilities deployed on the boundaries between IPv6 and IPv4 networks to facilitate the communication between IPv6-only clients and IPv4 servers.

This document proposes several load-balancing approaches for NAT64 devices, which can be utilized in the delivery of highly available services, and compares the advantages and disadvantages of various prefix selection policies.

The issues with failover and redundancy are outside the scope of this document. An in depth analysis of those issues is elaborated in [I-D.xu-behave-stateful-nat-standby].

2. Terminology

This document makes use of the terms defined in [I-D.ietf-behave-v6v4-xlate-stateful], [I-D.ietf-behave-dns64] and [RFC6052]. Below are provided the terms specific to this document:

- o Prefix64: an IPv6 prefix used for synthesizing IPv6 addresses representing IPv4 hosts in the IPv6 realm. See [RFC6052] for more details on how IPv4-embedded IPv6 addresses are built.
- o NAT64-enabled device (or NAT64 device for short): is a device which embeds a NAT64 function as defined in [I-D.ietf-behave-v6v4-xlate-stateful].
- o A load-balancer is a facility which can select a NAT64 device from a set of deployed devices for a given IPv6-only client according to the pre-specified policy. Typically, a load-balancer provisions a client with an IPv6 address of the IPv4-only server that the client is going to access. Prefix64 is elaborately selected by the load-balancer so that the corresponding NAT64 device can intercept the packets between the above communicating parties and correctly process them.

3. Reminder on the Load Balancing Objectives

Load balancing is a technique used by network operators (including service providers, enterprise networks, etc.) to distribute the load among several ingress/egress points, several paths, several topologies, etc.

In the context of IPv4-IPv6 interconnection, load balancing is mainly motivated by the needs listed below:

- o to optimize the resources usage of deployed NAT64 devices (e.g., several ingress/egress NAT64 devices);
- o to avoid congesting a single NAT64 device while other free resources are still available;
- o to optimize IPv6-IPv4 interconnection costs especially when several NAT64 providers are involved.

Techniques to balance the load among a set of NAT64 devices can be achieved on a load-balancer managing a farm of NAT64 devices, on a single NAT64 device to select the appropriate outbound interface, or it can be implicitly achieved owing to dedicated tweaking operations (e.g., use of anycast-based service, use of distinct Prefix64, etc.). Note that considerations to balance the traffic between several outbound interfaces of a NAT64 device are out of scope of this document.

Various types of load balancing can be considered as defined in the following sub-sections.

3.1. Inbound Load Balancing

Inbound load balancing means that incoming traffic (i.e., the traffic received from an IPv4-only host) is distributed among a set of NAT64 devices located at the boundary of the IPv4 realm and the IPv6 one.

In a stateful NAT64 case, inbound load balancing cannot be explicitly configured because IPv4-only clients can not initiate sessions to an IPv6-only server (except for IPv6-only hosts which pre-installed static entries in the NAT64 using PCP [I-D.ietf-pcp-base] for instance).

In a stateless NAT64 case, inbound load balancing can be achieved by configuring distinct IPv4 address pools on each stateless NAT64 device (or these pools are to be configured with distinct routing metrics in each NAT64 device). This practice may lead to asymmetric paths (i.e., distinct NAT64 devices will handle the outgoing and the inbound packets) if the same NSP (Network Specific Prefix, [RFC6052]) is provisioned on those NAT64 devices. This can be seen as an issue for some operators because the legal stored data and activity logs can be increased. If downstream and upstream paths have similar characteristics (e.g., one-way delay, one-way delay variation, throughput), the path asymmetry is not an issue from a service perspective.

Note: As an alternative to address this problem, a load balancer needs to be deployed on each side of the NAT64 devices. The balancers need to know the policies of each other so as to work in a cooperative way. Refer to Section 6.2 for more discussion.

3.2. Outbound Load Balancing

Outbound load balancing means the effort of distributing outgoing traffic (i.e., the traffic issued by IPv6-only hosts) among a set of NAT64 devices.

Only this flavor of load balancing is elaborated in the remainder of this document.

4. Basic Load Balancing Considerations

In practice, it is important to guarantee the outgoing traffic and the associated incoming traffic is stuck to a same stateful NAT64 device, so that the NAT64 device can get essential knowledge to correctly process the incoming packets. That is, if a stateful NAT64 device processes a request from an IPv6 client to an IPv4 server, it MUST be able to intercept and process the correspondent reply from the server. To achieve this, distinct external IPv4 addresses SHOULD be configured on each stateful NAT64 device. Otherwise, if the same IPv4 address pool is provisioned on two distinct NAT64 devices (e.g., NAT64-A and NAT64-B) and since the routing paths may not be symmetric, outbound packets may be intercepted by NAT64-A while inbound packets may be received by NAT64-B. NAT64-B will reject the packets it received because no state to process these packets was instantiated beforehand, and thus the communication will fail.

The load balancing SHOULD NOT be solely done based on the traffic load distribution but SHOULD persevere the assignment of the same external IPv4 address for all active sessions initiated by an IPv6-only host. The criteria for distributing customers among a set of NAT64 devices may be implemented during the IPv6 configuration phase of a IPv6-only host or during the processing of actual traffic issued by that host.

In the circumstances where the static mapping entries of an IPv6-only client are pre-installed in a given stateful NAT64 device, the enforced load-balancing technique SHOULD "redirect" the traffic from the client to the NAT64-device where its static mappings are pre-installed.

Note: Some dynamic protocols such as PCP [I-D.ietf-pcp-base] may include manes to detect the unavailability of a NAT64 and to re-

install the mappings in the new discovered NAT64. But, for the manually configured mappings, the issue is still there.

From the operator's perspective, a load balancing solution SHOULD be deterministic, that is, that the actual traffic distribution should be strictly compliant with what is expected by the system manager. Furthermore, the operations of distributing the load among multiple NAT64 devices SHOULD be covered from end-users. This means that end-users should not be aware of the presence of multiple NAT64 devices in the core network and the selection of the appropriate NAT64 device should not assume any intervention by the customer/host.

When implementing load balancing, it should not lead to (severe) QoS degradation between potential paths. Note that the perceived quality may not only depend on the load balancing technique to distribute the traffic among available path/nodes but it is closely related to the underlying topology (i.e., location of the NAT64 devices, routing metrics configuration, etc.).

An efficient load balancing system SHOULD NOT redirect the traffic to a congested NAT64 device while other NAT64 resources are available. Load indicators (i.e., the data reflecting the load imposed on NAT64 devices) may be disseminated to drive the process of selecting a NAT64 device to handle an ongoing IPv6 packet. These indicators may be based on (almost) real-time measurement tools or based on a traffic logic configured on the load-balancer (e.g., a NAT64 device can handle N IPv6-only hosts).

5. Stateless NAT64

According to [RFC6052], IPv4-Translated and IPv4-Converted IPv6 addresses SHOULD use the same Network Specific Prefix (NSP). To distribute the traffic among a set of stateless NAT64 devices, the alternatives described hereafter can be envisaged. For the anycast-based mode the same Prefix64 is used while for the remaining options, specific Prefix64s are used.

5.1. Anycast-based Mode

The same IPv6 NSP is provisioned to all stateless NAT64 devices; IPv6 hosts are distributed natively among several stateless NAT64 devices. This means that the closest (from a routing perspective) stateless NAT64 device will be used to process IPv6 (resp. IPv4) packets destined to an IPv4 (resp. IPv6) destination.

The efficiency of this mode largely depends on the underlying topology (e.g., location of NAT64 devices) and routing engineering

policies. Moreover, a stateless NAT64 device may be overloaded if the routing is not appropriately tuned and/or if the NAT64 devices are not appropriately dimensioned.

The introduction or the removal of NAT64 device(s) can be achieved without modifying the configuration of DHCPv6 servers. During failure events of a NAT64 system, other NAT64 devices can handle the traffic without any intervention.

5.2. DHCPv6-based Mode

To implement this mode, each stateless NAT64 device is configured with a dedicated NSP. During the IPv6 prefix assignment phase, requesting IPv6 hosts are provided with IPv4-Translated IPv6 prefix using the NSP of the NAT64 device that will be used to handle traffic issued from those hosts and destined to an IPv4 host.

If DHCPv6 is used for the provisioning of IPv6 prefixes, DHCPv6 servers SHOULD be provided with the number of customers to be serviced per dedicated NSP (i.e., an NSP prefix identifies a stateless NAT64 device). Dynamic load information (based on real time monitoring) MAY be provided to the DHCPv6 to drive the process of IPv6 prefix assignment and for better utilization of available NAT64 resources. Furthermore, and for routing optimization purposes and for service stability purpose (e.g., use the same NAT64 device hosting PCP-instructed port forwarding entries), other topological information SHOULD be used to tag the customers that should be serviced by each NAT64 device.

5.3. NAT64 Farm

An additional scheme would be the deployment of a farm of NAT64 devices with a load-balancer which is responsible for redirecting the traffic to the appropriate NAT64 instance. Unlike stateful NAT64, both IPv4 and IPv6 flows can be load balanced.

In this scenario, the same NSP SHOULD be used for all NAT64 devices belonging to the same farm.

Techniques to distribute the load among the NAT64 devices of the farm are similar to load-balancing techniques among several outbound interfaces of the same NAT64 system.

6. Stateful NAT64

Two variant of the load balancing techniques are elaborated hereafter. Unlike the first mode, anycast-based, the second category

requires Prefix64 selection to achieve load balancing. More details are provided below.

6.1. Anycast-based Mode

This mode assumes that the same IPv6 prefix (i.e., NSP or WKP, see [RFC6052]) is provisioned to all deployed stateful NAT64 devices. DNS64 function is provisioned with that prefix used for synthesizing AAAA records.

As stated in Section 4, distinct IPv4 address pools are configured to each NAT64 device. This ensures path symmetry; which means that the same NAT64 device will be used for handling both outbound and inbound packets exchanged in a same stateful conversation between two hosts.

The distribution of the traffic among deployed NAT64 devices is natively achieved relying on the underlying routing configuration. Off-line traffic engineering tools can be used to appropriately tweak the routing metrics so as to allow for acceptable traffic distribution.

The same NAT64 device SHOULD be used to handle all the packets issued by a given IPv6 host so that the same external IPv4 address is used to represent that host in the IPv4 realm. This means that oscillation phenomena induced by underlying routing MUST be avoided. By oscillation it is meant that the traffic customer is balanced between two NAT64 devices. The routing oscillation can be avoided owing to (off-line/on-line) traffic engineering techniques to select the appropriate location of the NAT64 devices in the network, the setting of underlying routing weights, establishment of explicit MPLS LSPs, etc.

6.2. Prefix64 Selection Policy

It is RECOMMENDED that the functionality of load balancers should be integrated into dedicated servers. Therefore, load-balancing can be transparent for IPv6-only hosts. The design options of load balancers are discussed in Section 6.3.

The following sub-sections elaborate on various modes for the prefix selection.

6.2.1. Source-Based Prefix Selection Policy

A source-based prefix selection policy allows a load-balancer to select Prefix64s according to the IPv6 addresses of its IPv6-only clients. For instance, when using a source-based prefix selection policy, the load-balancer in the above example can allocate an IPv6

address with Prefix64-A for the IPv4-only server if the IPv6 address of the client is odd, and Prefix64-B otherwise.

6.2.1.1. Pros

It is simple and has enough entropy to ensure reasonable load balancing across different NAT64 devices. 2.

The users are consistently assigned to the same NAT64 device for every outbound session. This is important because some applications identify a unique user across multiple transactions using the source IP address; examples include FTP and SSL VPNs. In addition, it is easier for a network management system (NMS) to monitor and manage the activities of a user. For instance, a NMS can collect the information about number of the concurrent sessions initiated by a user from a single NAT64 device. However, when using other policies, a user is not stuck to a NAT64 device, and thus NMS may have to collect such information from multiple NAT64 devices.

6.2.1.2. Cons

The efficiency of this procedure depends on the selection criteria and may not be deterministic in some cases where the traffic may be redirected to a congested NAT64 device.

6.2.2. Destination-Based Prefix Selection Policy

A destination-based prefix selection policy requires a load-balancer to choose Prefix64s according to the IP addresses of the IPv4 targets. For instance, when using a destination-based prefix selection policy, the load-balancer in the above example can allocate an IPv6 address with Prefix64-A for the IPv4 server if the IPv4 address of the server is odd, and prefix64-B otherwise. In practice, this type of policy can have lots of variations. For instance, when a DNS server is utilized as a load balancer, the server can select a prefix64 according to the hash value of the FQDN (Fully Qualified Domain Name) of the target server.

6.2.2.1. Pros

It is simple to implement;

6.2.2.2. Cons

A user accessing multiple IPv4 servers may be represented by multiple public IPv4 addresses since its traffic may be processed by different NAT64 systems. This will cause authentication problems in the applications (e.g., FTP and SSL VPNs) which take advantage of the

source IP addresses to identify users across different sessions. 2.

A user can not be redirected to the NAT64 device where it has instructed its port forwarding entries; 3.

Since there are more users than the servers providing contents, there is not enough entropy to ensure good load balancing. The NAT64 device that services a popular web-site will have to undertake much traffic. It is possible to define some strategies to make major sites evenly assigned to different NAT64s, e.g.- Google to NAT64-A, Facebook to NAT64-B, However, this solution can be onerous and requires heavy human involvement.

6.2.3. Round-Robin Prefix Selection Policy

A round-robin prefix selection policy allows a load-balancer to use various Prefix64s circularly in different requesting sessions. For instance, in the above example, the load-balancer can choose Prefix64-A in the first requesting session, choose Prefix64-B in the second, choose Prefix64-A in the third, choose Prefix64-B in the fourth, and so on.

6.2.3.1. Pros

Ensures reasonable distribution among a set of NAT64 devices.

6.2.3.2. Cons

A given IPv6-only hosts may be redirected to distinct NAT64 devices. Therefore, distinct IPv4 address may be used to represent the IPv6-only host in the IPv4 realm;

A user can not be redirected to the NAT64 device where it has instructed its port forwarding entries;

Requires a load balancer (e.g., a DNS64 or a DHCP server) to keep track of the assignments.

6.2.4. Dynamic Prefix Selection Policy

Although the capability of NAT64 devices can be considered as a factor in the designing of the above three types of policies, they are still static and not able to be adjusted according the load changes of NAT64 devices in a timely fashion. If we intend to enable a load-balancer to dynamically modify its policies according to NAT64s' real-time load changes, a dynamic prefix selection policy is necessary. For instance, a DNS64 system or DHCPv6 can use SNMP to collect the information of the overheads (e.g., CPU utilizing rates,

free memory amounts, concurrent session numbers, and session numbers per second) imposed on different NAT64-based devices. Based on such information, the load-balancer can distribute the loads on different NAT64 devices in a more reasonable way.

6.2.4.1. Pros

This type of policy can effectively avoid the unbalanced distribution of overheads on different NAT64 devices.

6.2.4.2. Cons

Such a policy may introduce additional communication and management complexities to a NAT64 device. The complexity depends on the means used to disseminate the NAT64 load.

6.3. Options for Implementing Load-balancers

In practice, the functionality of a load-balancer can be performed by, e.g.- a DNS64 server, a DNS server, a DHCP server, an edge router, or even an IPv6 host itself.

6.3.1. DNS64 Servers

When collaborating with NAT64 devices, a DNS64 server can be solicited by an IPv6-only client to initiate communications to an IPv4-only server identified by a FQDN.

Let us assume there is an IPv6-only client connected to an IPv6 network which attempts to communicate to an IPv4-only server. For the purpose of load balancing, the DNS64 server needs to select a Prefix64 based on one of the prefix selection policies defined in Section 6.2 and use it when synthesizing AAAA RRs. Using the synthesized IPv6 address, the IPv6-only client will take advantage of the NAT64 associated with the Prefix64 to communicate with the IPv4-only server.

When DNS64 is used as a means to load balance the hosts among a group of NAT, DNS64 SHOULD be able to assign the same NAT64 to the same hosts. Means to identify the host SHOULD be supported. This is not natively supported by DNS servers.

A drawback of this mode is that for traffic which does not require a DNS resolution, the packets may flow using a distinct NAT device, and therefore use a distinct external IP address.

6.3.2. Prefix64 Assigners

[I-D.korhonen-behave-nat64-learn-analysis] analyzes various solutions for a host in an IPv6-only network to obtain the Prefix64 of a NAT64 device. With the Prefix64, the hosts can synthesize an appropriate IPv6 address which can route packets to the translator. In the designing of load balancers for NAT64 devices, these approaches are worthwhile to consider.

6.3.2.1. DNS64 Servers

In [I-D.korhonen-behave-nat64-learn-analysis], a NAPTR RR to represent NAT64's Prefix64 is analyzed as part of the candidate solutions. When using DNS servers to act as load balancers for NAT64 devices, multiple NAPTR RRs need to be added the zone file. Every NAPTR RR consists of a Prefix64. Upon receiving a NAPTR query, the DNS server replies the requester with a NAPTR RR according to a pre-specified selection policy. Note that the destination-based prefix selection policy is not applicable in this case because the DNS server may lack the knowledge of the IP address of the queried IPv4 host.

6.3.2.2. DHCPv6 Servers

It is mentioned in [I-D.korhonen-behave-nat64-learn-analysis] that a DHCPv6 server can be used to allocate Prefix64s for hosts, and so a DHCP server has a potential to act as a load balancer for NAT64 devices. Similar with the solution proposed in Section 6.3.2.1, it is difficult for a DHCP server to identify the IP addresses of the IPv4 hosts which its clients intend to communicate with. Therefore, only the source-based policy, the round-robin policy, or the dynamic policy can be used in this approach.

Also, a DHCPv6 server can be adopted to allocate different DNS64 servers for its users in various standard DHCPv6 host configuration processes according to certain selection policies. Unlike the DNS64 servers discussed in Section 6.3.1, in this case a DNS64 server needs to only synthesize AAAA records using a single Prefix64.

The load of NAT devices may be provided to DHCP servers to assist the selection of the DNS64 to be used for new connecting hosts.

6.3.2.3. Default Gateways

[I-D.korhonen-behave-nat64-learn-analysis] also discusses the possibility of using Router Advertisement (RA) messages to transfer Prefix64s for IPv6 users. If the edge router is attached to only one multicast link, no prefix selection policy defined in Section 6.2 can

be used. If the edge router is attached to multiple multicast links, the source-based policy, the round-robin policy or the dynamic policy can be used. Because at this phase it is difficult for an edge router to identify the IP addresses of the IPv4 hosts which the IPv6 hosts will communicate with, the destination-based prefix selection policy is unfeasible.

6.3.2.4. IPv6 Clients

It is possible for an IPv6 host to learn multiple Prefix64s through the approaches defined in [I-D.korhonen-behave-nat64-learn-analysis] and then select one based on a certain prefix selection policy. Such a policy can be the destination-based policy, the source-based policy (only one prefix64 is used), the round-robin policy or the dynamic policy.

This solution is not deterministic and can lead to congesting a given NAT64 device.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

As mentioned previously, all the traffic between an IPv6 host and an IPv4 host should be intercepted and processed by a same NAT64 device. However, when using certain policies (e.g., the destination-based prefix selection policy and the Round-Robin prefix selection policy), this requirement cannot be fulfilled. The traffic of a user will be distributed to different NAT64 devices. Under such a circumstance, it may be difficult for network management systems to collect information from different NAT64 devices in order to monitor users' behavior in a real in time fashion. In addition, it can be difficult for intrusion detection/prevision systems to combine the operations of a user so as to reason whether she is trying to perform a multi-step attack.

Another security concern is load balancers. Because load balancers play an important role in distributing traffic to different NAT64 devices, the communication between users and load balancers should be secured. Otherwise, attackers may disturb load balancing and carry out DDoS attacks by modifying the packets sent from load balancers.

9. Contributors

The following individuals have contributed to this document:

Xuewei Wang
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: wangxuewei@huawei.com

Yan Wang
CNNIC
No.4 South 4th Street,
Beijing, Zhongguancun 100190
P. R. China

Email: wangyan-lab@cnnic.cn

Cameron Byrne
T-Mobile USA
3617 131st Ave SE
Bellevue, WA 98006
US
Email: cameron.byrne@t-mobile.com

Dong Zhang
Huawei Symantec
KuiKe Building, No.9 Xinxu Rd.,
Beijing, Hai-Dian District 100085
P. R. China

Email: zhangdong_rh@huaweisymantec.com

Zhenqiang Li
China Mobile
Unit2, Dacheng Plaza, No. 28 Xuanwumenxi Ave, Xicheng District
Beijing 100053
P.R. China

Email: lizhenqiang@chinamobile.com

10. References

10.1. Normative References

- [I-D.ietf-behave-dns64]
Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum,
"DNS64: DNS extensions for Network Address Translation
from IPv6 Clients to IPv4 Servers",
draft-ietf-behave-dns64-11 (work in progress),
October 2010.
- [I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful
NAT64: Network Address and Protocol Translation from IPv6
Clients to IPv4 Servers",
draft-ietf-behave-v6v4-xlate-stateful-12 (work in
progress), July 2010.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X.
Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052,
October 2010.

10.2. Informative References

- [I-D.ietf-pcp-base]
Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P.
Selkirk, "Port Control Protocol (PCP)",
draft-ietf-pcp-base-13 (work in progress), July 2011.
- [I-D.korhonen-behave-nat64-learn-analysis]
Korhonen, J. and T. Savolainen, "Analysis of solution
proposals for hosts to learn NAT64 prefix",
draft-korhonen-behave-nat64-learn-analysis-02 (work in
progress), February 2011.
- [I-D.xu-behave-stateful-nat-standby]
Xu, X., Boucadair, M., Lee, Y., and G. Chen, "Redundancy
Requirements and Framework for Stateful Network Address
Translators (NAT)",
draft-xu-behave-stateful-nat-standby-06 (work in
progress), October 2010.

Authors' Addresses

Dacheng Zhang
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: zhangdacheng@huawei.com

Xiaohu Xu
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxu Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: xuxh@huawei.com

Mohamed Boucadair
France Telecom
Rennes,
France

Email: mohamed.boucadair@orange-ftgroup.com

