Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 5, 2012

M. Boucadair
France Telecom
J. Touch
USC/ISI
P. Levis
France Telecom
R. Penno
Juniper Networks

September 2, 2011

Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments

draft-boucadair-intarea-nat-reveal-analysis-04

Abstract

This document analyzes a set of solution candidates which have been proposed to mitigate some of the issues encountered when address sharing is used. In particular, this document focuses on means to reveal a host identifier when a Carrier Grade NAT (CGN) or application proxies are involved in the path. This host identifier must be unique to each host under the same shared IP address.

The ultimate goal is to assess the viability of proposed solutions and hopefully to make a recommendation on the more suitable solution(s).

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 March 5, 2012.

Table of Contents

1. Introduction

As reported in [RFC6269], several issues are encountered when an IP address is shared among several subscribers. Examples of such issues are listed below:

- Implicit identification (Section 13.2 of [RFC6269])
- SPAM (Section 13.3 of [RFC6269])
- Blacklisting a mis-behaving user (Section 13.1 of [RFC6269])
- Redirect users with infected machines to a dedicated portal (Section 5.1 of [RFC6269])

The sole use of the IPv4 address is not sufficient to uniquely distinguish a host. As a mitigation, it is tempting to investigate means which would help in disclosing an information to be used by the remote server as a means to uniquely disambiguate packets of hosts using the same IPv4 address.

The risk of not mitigating these issues are: OPEX increase for IP connectivity service providers (costs induced by calls to a hotline), revenue loss for content providers (loss of users audience), customers unsatisfaction (low quality of experience, service segregation, etc.).

1.1. Problem to Be Solved

Observation: Today, servers use the source IPv4 address as an identifier to treat some incoming connections differently. Tomorrow, due to the introduction of CGNs (e.g., NAT44 [I-D.ietf-behave-lsn-requirements], NAT64 [RFC6146]), that address will be shared. In particular, when a server receives packets from the same source address. Because this address is shared, the server does not know which host is the sending host.

Objective: The server should be able to sort out the packets by sending host.

Requirement: The server must have extra information than the source IP address to differentiate the sending host. We call HOST_ID this information.

For all solutions analyzed, we provide answers to the following questions:

What is the HOST_ID? It must be unique to each host under the same IP address. It does not need to be globally unique. Of course, the combination of the (public) IPv4 source address and the identifier (i.e., HOST_ID) ends up being relatively unique. As unique as today's 32-bit IPv4 addresses which, today, can change

when a host re-connects.

Where is the HOST_ID? (which protocol, which field): If the HOST_ID is put at the IP level, all packets will have to bear the identifier. If it is put at a higher connection-oriented level, the identifier is only needed once in the session establishment phase (for instance TCP three-way-handshake), then, all packets received in this session will be attributed to the HOST_ID designated during the session opening.

Who puts the HOST_ID? For almost all the analyzed solutions, the address sharing function injects the HOST_ID. When there are several address sharing functions in the data path, we describe to what extent the proposed solution is efficient. Another option to avoid potential performance degradation is to let the host inject its HOST_ID but the address sharing function will check its content (just like an IP anti-spoofing function).

What are the security considerations? Security considerations are common to all analyzed solutions (see Section 5). Privacy-related aspect are discussed in Section 1.2.

1.2. HOST_ID and Privacy

HOST_ID provides an additional information to uniquely disambiguate a host among those sharing the same IP address. Unlike URIs, HOST_ID does not leak user's identity information.

The HOST_ID does not reveal more privacy information than what the source IP address does in a non-shared address environment (see [I-D.morris-privacy-considerations]).

The volatility of the HOST_ID information is similar to the source IP address: a distinct HOST_ID may be used by the address sharing function when the host reboots or gets a new internal IP address. If the HOST_ID is persistent it may be used to track a host (similar to persistent IP addresses).

The trust on the information conveyed in the HOST_ID is likely to be the same as for current practices with the source IP address. In that sense, a HOST_ID can be spoofed as this is also the case for spoofing an IP address.

It is the responsibility of the remote server to rely or not on the content of the HOST_ID to enforce its policies and to log or not the content conveyed in the HOST_ID.

Enabling explicit identification means and adequate security suite is

more robust than relying on source IP address or HOST_ID. But tension may appear between strong privacy and usability (see Section 4.2 of [I-D.iab-privacy-workshop]).

1.3. IPv6 May Also Be Concerned

Issues similar to the ones described in Section 1.1 may be encountered also in an IPv6 environment (e.g., when the same /64 is used among several hosts).

1.4. Purpose and Scope

The purpose of this document is to analyze the solutions that have been proposed so far and to assess to what extent they solve the problem (see Section 1.1).

The purpose of this document is not to argue in favor of mandating the use of a HOST_ID but to document encountered issues, proposed solutions and their limitations.

Only IPv4-based solutions are analyzed in the following sections:

- define a new IP option (Section 3.1)
- define a new TCP option (Section 3.2)
- use the Identification field of IP header (denoted as IP-ID, Section 3.3)
- inject application headers (Section 3.4)
- enable Proxy Protocol ((Section 3.5))
- use of port set (Section 3.6)
- activate HIP (Section 3.7).

2. Recommendations

The following Table 1 summarizes the approaches analyzed in this document.

- "Success ratio" indicates the ratio of successful communications when the option is used. Provided figures are inspired from the results documented in [Options].
- "Deployable today" indicates if the solution can be generalized without any constraint on current architectures and practices.
- "Possible Perf Impact" indicates the level of expected performance degradation. The rationale behind the indicated potential performance degradation is whether the injection requires some treatment at the IP level or not.

o "OS TCP/IP Modif" indicates whether a modification of the OS TCP/IP stack is required at the server side.

	IP Option	TCP Option	IP-ID	HTTP Header (XFF)	Proxy Protocol	Port Set	HIP
UDP	Yes	No	Yes	No	No	Yes	
TCP	Yes	Yes	Yes	No	Yes	Yes	
HTTP	Yes	Yes	Yes	Yes	Yes	Yes	
Encrypted Traffic	Yes	Yes	Yes	No	Yes	Yes	
Success Ratio	30%	99%	100%	100%	Low	100%	Low
Possible Perf Impact	High	Med to High	Low to Med	Med to High	High	No	N/A
OS TCP/IP Modif	Yes	Yes	Yes	No	No	No	
Deployable Today	Yes	Yes	Yes	Yes	No	Yes	No
Notes			(1)	(2)		(1) (3)	(4) (5)

Table 1: Summary of analyzed solutions.

Notes for the above table:
(1) Requires mechanism to advertise NAT is participating in this scheme (e.g., DNS PTR record)
(2) This solution is widely deployed
(3) When the port set is not advertised, the solution is less efficient for third-party services.
(4) Requires the client and the server to be HIP-compliant and HIP infrastructure to be deployed.

(5) If the client and the server are HIP-enabled, the address sharing function does not need to insert a host-hint. If the client is not HIP-enabled, designing the device that performs address sharing to act as a UDP/TCP-HIP relay is not viable.

According to the above table and the analysis elaborated in Section 3:

- IP Option, IP-ID and Proxy Protocol proposals are broken;
- HIP is not largely deployed;
- The use of Port Set may contradict the port randomization [RFC6056] requirement identified in [RFC6269]. This solution can be used by a service provider for the delivery of its own service offerings relying on implicit identification.
- XFF is de facto standard deployed and supported in operational networks (e.g., HTTP Severs, Load-Balancers, etc.).
- From an application standpoint, the TCP Option is superior to XFF since it is not restricted to HTTP. Nevertheless XFF is compatible with the presence of address sharing and load-balancers in the communication path. To provide a similar functionality, the TCP Option may be extended to allow conveying a list of IP addresses to not loose the source IP address in the presence of load-balancers. Note that TCP Option requires the modification of the OS TCP/IP stack of remote servers; which can be seen as a blocking point.

As a conclusion of this analysis, the following recommendation is made:

[Hopefully to be completed]

3. Solutions Analysis

3.1. Define an IP Option

3.1.1. Description

This proposal aims to define an IP option [RFC0791] to convey a "host identifier". This identifier can be inserted by the address sharing function to uniquely distinguish a host among those sharing the same IP address. The option can convey an IPv4 address, the prefix part of an IPv6 address, etc.

Another way for using IP option has been described in Section 4.6 of [RFC3022].

3.1.2. Analysis

Unlike the solution presented in Section 3.2, this proposal can apply for any transport protocol. Nevertheless, it is widely known that routers (and other middle boxes) filter IP options. IP packets with IP options can be dropped by some IP nodes. Previous studies demonstrated that "IP Options are not an option" (Refer to [Not_An_Option], [Options]).

As a conclusion, using an IP option to convey a host-hint is not viable.

3.2. Define a TCP Option

3.2.1. Description

This proposal [I-D.wing-nat-reveal-option] defines a new TCP option called USER_HINT. This option encloses the TCP client's identifier (e.g., the lower 16 bits of their IPv4 address, their VLAN ID, VRF ID, subscriber ID). The address sharing device inserts this TCP option to the TCP SYN packet.

3.2.2. Analysis

The risk related to handling a new TCP option is low as measured in [Options].

[I-D.wing-nat-reveal-option] discusses the interference with other TCP options.

Using a new TCP option to convey the host-hint does not require any modification to the applications but it is applicable only for TCP-based applications. Applications relying on other transport protocols are therefore left unsolved.

Some downsides have been raised against defining a TCP option to reveal a host identity:

- Conveying an IP address in a TCP option may be seen as a violation of OSI layers but since IP addresses are already used for the checksum computation, this is not seen as a blocking point. Moreover, Updated version of [I-D.wing-nat-reveal-option] does not allow anymore to convey an IP address (the HOST_ID is encoded in 16bits).

- TCP option space is limited, and might be consumed by the TCP client. Earlier versions of [I-D.wing-nat-reveal-option] discuss two approaches to sending the HOST_ID: sending the HOST_ID in the TCP SYN (which consumes more bytes in the TCP header of the TCP SYN) and sending the HOST_ID in a TCP ACK (which consumes only two bytes in the TCP SYN). Content providers may find it more desirable to receive the HOST_ID in the TCP SYN, as that more closely preserves the host hint received in the source IP address as per current practices. It is more complicated to implement sending the HOST_ID in a TCP ACK, as it can introduce MTU issues if the ACK packet also contains TCP data, or a TCP segment is lost. The latest specification of the HOST_ID TCP Option, documented at [I-D.wing-nat-reveal-option], allows only to enclose the HOST_ID in the TCP SYN packet.

- When there are several NATs in the path, the original HOST_ID may be lost. In such case, the procedure may not be efficient.

- Interference with current usages such as X-Forwarded-For (see Section 3.4) should be elaborated to specify the behavior of servers when both options are used; in particular specify which information to use: the content of the TCP option or what is conveyed in the application headers.

- When load-balancers or proxies are in the path, this option does not allow to preserve the original source IP address and source. Preserving such information is required for logging purposes for instance.

3.3. Use the Identification Field of IP Header (IP-ID)

3.3.1. Description

IP-ID (Identification field of IP header) can be used to insert an information which uniquely distinguishes a host among those sharing the same IPv4 address. An address sharing function can re-write the IP-ID field to insert a value unique to the host (16 bits are sufficient to uniquely disambiguate hosts sharing the same IP address). Note that this field is not altered by some NATs; hence some side effects such as counting hosts behind a NAT as reported in [Count].

A variant of this approach relies upon the format of certain packets, such as TCP SYN, where the IP-ID can be modified to contain a 16 bit host-hint. Address sharing devices performing this function would require to indicate they are performing this function out of band, possibly using a special DNS record.

3.3.2. Analysis

This usage is not compliant with what is recommended in [I-D.ietf-intarea-ipv4-id-update].

3.4. Inject Application Headers

3.4.1. Description

Another option is to not require any change at the transport nor the IP levels but to convey at the application payload the required information which will be used to disambiguate hosts. This format and the related semantics depend on its application (e.g., HTTP, SIP, SMTP, etc.).

For HTTP, the X-Forwarded-For (XFF) header can be used to display the original IP address when an address sharing device is involved. Service Providers operating address sharing devices can enable the feature of injecting the XFF header which will enclose the original IPv4 address or the IPv6 prefix part. The address sharing device has to strip all included XFF headers before injecting their own. Servers may rely on the contents of this field to enforce some policies such as blacklisting misbehaving users. Note that XFF can also be logged by some servers (this is for instance supported by Apache).

3.4.2. Analysis

Not all applications impacted by the address sharing can support the ability to disclose the original IP address. Only a subset of protocols (e.g., HTTP) can rely on this solution.

For the HTTP case, to prevent users injecting invalid host-hints, an initiative has been launched to maintain a list of trusted ISPs using XFF: See for example the list available at: [Trusted_ISPs] of trusted ISPs as maintained by Wikipedia. If an address sharing device is on the trusted XFF ISPs list, users editing Wikipedia located behind the address sharing device will appear to be editing from their "original" IP address and not from the NATed IP address. If an offending activity is detected, individual hosts can be blacklisted instead of all hosts sharing the same IP address.

XFF header injection is a common practice of load balancers. When a load balancer is in the path, the original content of any included XFF header should not be stripped. Otherwise the information about the "origin" IP address will be lost.

When several address sharing devices are crossed, XFF header can

convey the list of IP addresses. The origin HOST_ID can be exposed to the target server.

XFF also introduces some implementation complexity if the HTTP packet is at or close to the MTU size.

It has been reported that some "poor" implementation may encounter some parsing issues when injecting XFF header.

For encrypted HTTP traffic, injecting XFF header may be broken.

3.5. PROXY Protocol

3.5.1. Description

The solution, referred to as Proxy Protocol [Proxy], does not require any application-specific knowledge. The rationale behind this solution is to prepend each connection with a line reporting the characteristics of the other side's connection as shown in the example below (excerpt from [Proxy]):

```
PROXY TCP4 198.51.100.1 198.51.100.11 56324 443\r\n
```

Upon receipt of a message conveying this line, the server removes the line. The line is parsed to retrieve the transported protocol. The content of this line is recorded in logs and used to enforce policies.

3.5.2. Analysis

This solution can be deployed in a controlled environment but it can not be deployed to all access services available in the Internet. If the remote server does not support the Proxy Protocol, the session will fail. Other complications will raise due to the presence of firewalls for instance.

As a consequence, this solution is broken and can not be recommended.

3.6. Enforce a Source-based Selection Algorithm at the Server Side (Port Set)

3.6.1. Description

This solution proposal does not require any action from the address sharing function to disclose a host identifier. Instead of assuming all the ports are associated with the same host, a random-based algorithm (or any port selection method) is run to generate the set of ports (including the source port of the received packet). The

length of the ports set to be generated by the server may be configurable (e.g., 8, 32, 64, 512, 1024, etc.). Instead of a random-based scheme, the server can use contiguous port ranges to form the port sets.

The server may reduce (or enlarge) the width of the ports set of the misbehaving action is (not) mitigated.

A variant of this proposal is to announce by off-line means the port set assignment policy of an operator. This announcement is not required for the delivery of internal services (i.e., offered by the service provider deploying the address sharing function) relying on implicit identification.

3.6.2. Analysis

In nominal mode, no coordination is required between the address sharing function and the server side but the efficiency of the method depends on the port set selection algorithm.

The method is more efficient if the provider that operates the address sharing device advertises its port assignment policy but this may contradicts the port randomization as identified in [RFC6269].

The method is deterministic for the delivery of services offered by the service provider offering also the IP connectivity service.

3.7. Host Identity Protocol (HIP)

3.7.1. Description

[RFC5201] specifies an architecture which introduces a new namespace to convey an identity information.

3.7.2. Analysis

This solution requires both the client and the server to support HIP [RFC5201]. Additional architectural considerations are to be taken into account such as the key exchanges, etc.

If the address sharing function is required to act as a UDP/TCP-HIP relay, this is not a viable option.

4. IANA Considerations

This document does not require any action from IANA.

5. Security Considerations

The same security concerns apply for the injection of an IP option, TCP option and application-related content (e.g., XFF) by the address sharing device. If the server trusts the content of the HOST_ID field, a third party user can be impacted by a misbehaving user to reveal a "faked" original IP address.

6. Acknowledgments

Many thanks to D. Wing and C. Jacquenet for their review, comments and inputs.

Thanks also to P. McCann, T. Tsou, Z. Dong, B. Briscoe, T. Taylor, M. Blanchet, D. Wing and A. Yourtchenko for the discussions in Prague.

Some of the issues related to defining a new TCP option have been raised by L. Eggert.

7. References

7.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate 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.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.

7.2. Informative References

[Count] "A technique for counting NATted hosts", <http://www.cs.columbia.edu/~smb/papers/fnat.pdf>.

[I-D.iab-privacy-workshop] Cooper, A., "Report from the Internet Privacy Workshop", draft-iab-privacy-workshop-00 (work in progress), June 2011.

[I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NAT (CGN)", draft-ietf-behave-lsn-requirements-03 (work in progress), August 2011.

[I-D.ietf-intarea-ipv4-id-update]
Touch, J., "Updated Specification of the IPv4 ID Field", draft-ietf-intarea-ipv4-id-update-02 (work in progress), March 2011.

[I-D.morris-privacy-considerations]
Aboba, B., Morris, J., Peterson, J., and H. Tschofenig, "Privacy Considerations for Internet Protocols", draft-morris-privacy-considerations-03 (work in progress), March 2011.

[I-D.wing-nat-reveal-option]
Yourtchenko, A. and D. Wing, "Revealing hosts sharing an IP address using TCP option", draft-wing-nat-reveal-option-02 (work in progress), June 2011.

[Not_An_Option]
R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica,, "IP options are not an option", 2005, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-24.html>.

[Options] Alberto Medina, Mark Allman, Sally Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", 2005, <http://conferences.sigcomm.org/imc/2004/papers/p336-medina.pdf>.

[Proxy] Tarreau, W., "The PROXY protocol", November 2010, <http://haproxy.1wt.eu/download/1.5/doc/proxy-protocol.txt>.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[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.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

[Trusted_ISPs]
"Trusted XFF list", <http://meta.wikimedia.org/wiki/XFF_project#Trusted_XFF_list>.

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange-ftgroup.com

Joe Touch
USC/ISI

Email: touch@isi.edu

Pierre Levis
France Telecom
Caen, 14000
France

Email: pierre.levis@orange-ftgroup.com

Reinaldo Penno
Juniper Networks
1194 N Mathilda Avenue
Sunnyvale, California 94089
USA

Email: rpenno@juniper.net

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 3, 2012

G. Chen
China Mobile

Oct 2011

NAT64 Operational Considerations
draft-chen-v6ops-nat64-cpe-03

Abstract

The document has summarized NAT64 usages on different modes, in which NAT64 may serve for a large-scale network or would give enterprise or residential service opportunities to be accessed by IPv6 remote subscribers. The document has described different operations for each usage and proposed operational considerations for each particular NAT64-mode.

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, 2012.

Copyright Notice

Table of Contents

1. Introduction

With fast developments of global Internet, the demands for IP address are rapidly increasing at present.This year, IANA announced that the global free pool of IPv4 depleted on 3 February. IPv6 is the only real option on the table. Operators have to accelerate the process of deploying IPv6 networks in order to address IP address strains.IPv6 deployment normally involves a step-wise approach where parts of the network should properly updated gradually. As IPv6 deployment progresses it may be simpler for operators and ICP/ISP to employ NAT64[RFC6146] functionalities at edge of IPv4 and IPv6 networks, since a significant part of network will still stay in IPv4 for long time. Especially, NAT64 could facilitate large ICP/ISP IPv6 transition process by eliminating upgrations of tremendous legacy IPv4 servers. Therefore, it's quite popular to deploy NAT64 at the front of IDC to shift the entire service to be IPv6-enable.

Depending on different usage, NAT64 could be deployed on different places. The document has summarized NAT64 usages on different modes. Considering the existing deployment approaches, the memo has proposed different operational consideration for each particular NAT64-mode.

2. NAT64-CGN Deployment

2.1. Deployment in IDC

NAT has widely used in data center environments whenever IDC have to make your IPv4-only content available to IPv6 clients.

Figure 1 illustrates the usage where an IPv6-only host would like to initiate communications with IDC in IPv4 domain through NAT64. The NAT64 would accept IPv6 incoming session and distribute them to multiple IPv4 servers.

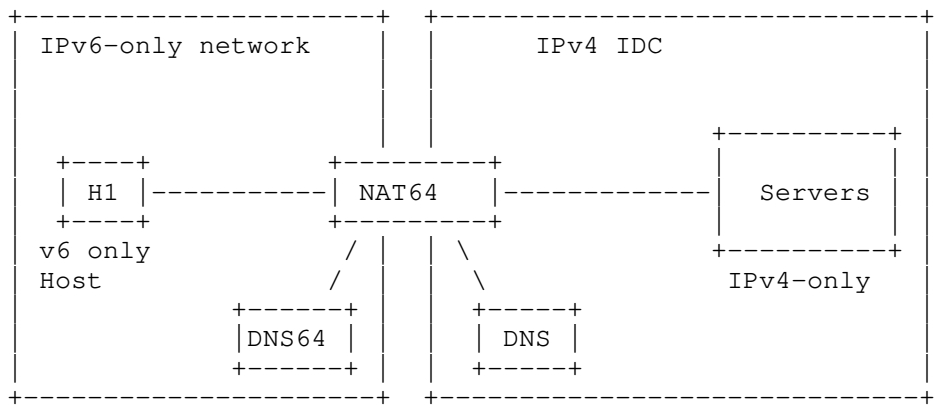


Figure 1: NAT64-CGN Mode Usage

NAT64 device in IDC may also take responsibilities of load balancer, which can accept incoming TCP/UDP sessions on a single virtual IPv6 interface or multiple IPv6 interfaces. Afterwards, it distributes them according to a specific algorithm it uses to multiple IPv4 servers. Ideally you could have a mix of IPv4 and IPv6 servers sitting behind the virtual IPv6 address.

Therein, NAT64 has to pick a new source IPv4 address and associated port number from local IPv4 address pool. DNS64 is a logical function that synthesizes DNS resource records(e.g., AAAA records containing IPv6 addresses) from DNS resource records actually contained in the DNS (e.g., A records containing IPv4 addresses).

2.2. Connecting with IPv4 Internet

NAT64 may also be used to connecting IPv6 users with IPv4 Internet. In this cases, NAT64 could colocated with BNG or Core Router to map legacy IPv4 servers into a NAT64 prefix and performs 6-to-4 address.

Therein, NAT64 would perform protocol translation mechanism and address translation mechanism. Protocol translation from an IPv4 packet header to an IPv6 packet header and vice versa is performed according to the IP/ICMP Translation Algorithm [RFC6145]. Address translation maps IPv6 transport addresses to IPv4 transport addresses and vice versa.

Following illustrates normal process for this usage.

- Step1: IPv6-only host performs an AAAA DNS query to DNS64 for the IPv6 address of the Pv4-only sever.

- Step2: DNS64 could not find the IPv6 address of the IPv4-only sever. So it tries to get the IPv4 address of the Pv4-only sever by sending A DNS query to DNS4.

- Step3: DNS4 return the A record to the DNS64.

- Step4: DNS64 map the IPv4 address to IPv6 address and send a synthetic AAAA record which is translated from A record to IPv6-only host.

- Step5: IPv6-only host send the IPv6 packet to the NAT64. NAT64 translates the IPv6 packet to IPv4 packet and send it to IPv4-only server.

2.3. NAT64-CGN Mode Requirements

According to above description for NAT64-CGN, the NAT64-CGN requirements are listed as following.

NAT64-CGN-R1: Each NAT64 device MUST have at least one unicast IPv6 prefix assigned to it, denoted Pref64::/n.

NAT64-CGN-R2:A NAT64 MUST have one or more unicast IPv4 addresses assigned to it.

NAT64-CGN-R3:Irrespective of the transport protocol used, the NAT64 MUST silently discard all incoming IPv6 packets containing a source address that contains the Pref64::/n.

NAT64-CGN-R4:The NAT64 MUST only process incoming IPv6 packets that contain a destination address that contains Pref64::/n. Likewise, the NAT64 MUST only process incoming IPv4 packets that contain a destination address that belongs to the IPv4 pool assigned to the NAT64.

NAT64-CGN-R5:NAT64 MUST support the algorithm for generating IPv6 representations of IPv4 addresses defined in RFC6052 as Address Translation Algorithms.

NAT64-CGN-R6:For incoming packets carrying TCP or UDP fragments with a non-zero checksum, NAT64 MAY elect to queue the fragments as they arrive and translate all fragments at the same time.

NAT64-CGN-R7: For incoming IPv4 packets carrying UDP packets with a zero checksum, if the NAT64 has enough resources, the NAT64 MUST

reassemble the packets and MUST calculate the checksum. If the NAT64 does not have enough resources, then it MUST silently discard the packets.

NAT64-CGN-R8: The NAT64 MAY require that the UDP, TCP, or ICMP header be completely contained within the fragment that contains fragment offset equal to zero.

NAT64-CGN-R9:The NAT64 MUST limit the amount of resources devoted to the storage of fragmented packets in order to protect from DoS attacks.

NAT64-CGN-R10:The NAT64 MUST make fragmentation process when MTU of incoming IPv4 traffic exceed maximum MTU on IPv6 side.

NAT64-CGN-R11:The NAT64 MAY let hosts and applications know IPv6 prefix used by the NAT64 and DNS64 so as to hosts have knowledge whether synthetic IPv6 address is targeted.

NAT64-CGN-R12:The NAT64 MAY decouple with DNS64 in order to establish communication with IPv4-only servers.

NAT64-CGN-R13:The NAT64 MAY take load-balancing functionalities incorporating with DNS64.

3. NAT64-CE Mode

NAT64-CE mode represents usages where there NAT64 is closed to customer edges, like enterprise network edge or residential network edge.

3.1. NAT64 at Enterprise Network Edge

Some enterprise would like to offers their employees with IPv6 access. However, the service may still stay in IPv4 domain. NAT64 useges in enterprise network could help shift all enterprise service to be IPv6 enable.

Figure 2 illustrates a network usage where an IPv6-only client attached to a dual-stack network, but the destination server is running on a private site where there is NAT64-CE numbered with public IPv6 addresses and private IPv4 addresses.

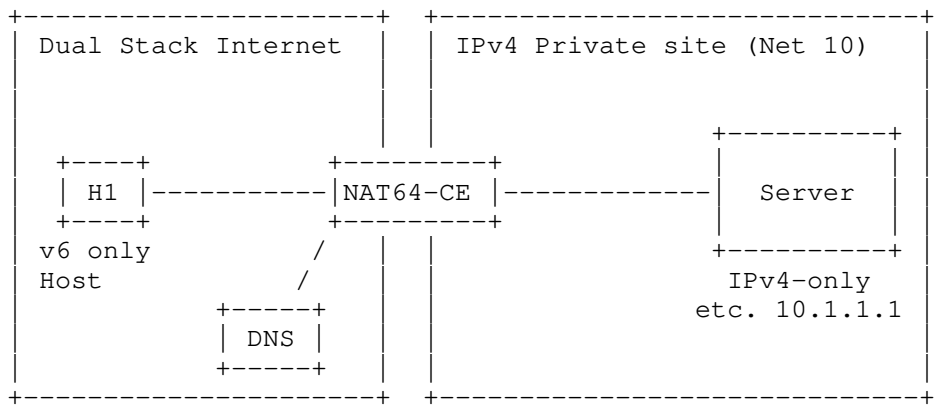


Figure 2: NAT64-CPE Mode Usage

3.2. NAT64 at Residential Network Edge

Residential servers are usually going beyond the operator's management. They may not able to IPv6-enable due to limitations of application supporting. In this case, ISP is still assigning private IPv4 address to servers. However, The nature of private IPv4 would block the end-to-end bi-directional communications. On the other hand, IPv6 will bring end-to-end benefits to operators. NAT64-CPE mode could let IPv6 users to access such IPv6-disable services in residential areas.

This scenario may appears in ISP network for several cases. As the instances, visitors go through distant network to take care of family affairs, like monitoring house security via residential camera, manipulating household appliances remotely prior to comeback home.

4. Security Considerations

Essentially, there are strong demands to have thorough security mechanism to prevent privacy invasion in NAT64-CPE scenario. The detailed considerations need to be further identified.

5. IANA Considerations

This memo includes no request to IANA.

6. Normative References

[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.

[RFC6204] Singh, H., Beebee, W., Donley, C., Stark, B., and O. Troan, "Basic Requirements for IPv6 Customer Edge Routers", RFC 6204, April 2011.

Author's Address

Gang Chen
China Mobile
53A,Xibianmennei Ave.,
Xuanwu District,
Beijing 100053
China

Email: chengang@chinamobile.com

Network working group
Internet Draft
Category: Standards Track
Expires: August 28, 2011

D. Cheng
Huawei Technologies

February 28, 2011

RADIUS Extensions for NAT Forwarding Port

draft-cheng-behave-nat-fwd-port-radius-ext-00

Status of this Memo

Copyright Notice

Abstract

This memo proposes two new RADIUS attributes with each to carry an Internal Port number and a Configured External Port number, both are associated with a specific NAT device and a specific user, and are configured on a RADIUS server such that when the user requests an Internet connection, the port mapping information can be conveyed to NAS that co-locates with the NAT device via RADIUS protocol, and is used during the NAT operation for IP flows to and from that user. The two attributes also include an IPv4 address or IPv6 address, respectively, as the pinhole internal IP address at the NAT device.

Table of Contents

1. Introduction

In most of the scenarios, the port mapping on a NAT device is dynamically created when the IP packets of an IP connection initiated by a user arrives. For some applications, the port mapping needs to be pre-defined allowing IP packets of applications from outside a NAT device to pass through and "port forwarded" to the correct user located behind the NAT device.

Port Control Protocol or PCP ([I-D.draft-ietf-pcp-base]), provides a mechanism to create pinholes from an external IP address to an internal IP address and port on a NAT device just to achieve the "port forwarding" purpose. PCP is a server-client protocol capable of creating or deleting a pinhole along with a rich set of features on a NAT device in dynamic fashion. In some deployment, all users need is a few, typically just one pre-configured port mapping for applications such

as web cam at home, and the lifetime of such a port mapping remains valid throughout the duration of the customer's Internet service connection time. In such an environment, it is possible to statically configure a port mapping on the RADIUS server for a user and let the RADIUS protocol to propagate the information to the associated NAT device.

In a broadband network, customer information is usually stored on a RADIUS server and at the time when a user initiates an Internet service request, the RADIUS server will populate the user's configuration information to the NAS, which is usually co-located with the BNG, after the connection request is granted. In many cases, the NAT function is also on the BNG, and therefore the port forwarding information can be configured on the RADIUS server as part of the user profile.

This memo proposes two new RADIUS attributes to carry Internal Port number and Configured External Port number, both are associated with a specific NAT device and a specific user, with an IPv4 address or IPv6 address as the pinhole internal address, respectively, and are configured on a RADIUS server such that when the user requests an Internet connection, the port mapping information can be conveyed to the NAS that co-locates with the NAT device via RADIUS protocol, and is used during the NAT operation for IP flows to and from that user.

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].

3. Operation

Port mapping information for NAT for a user (e.g., a CPE or host) is configured on a RADIUS server, along with other user information such as credentials. The port mapping information that is to be used during the NAT procedure is going to be populated from the RADIUS server to the NAT device using RADIUS protocol.

In Figure-1, a Network Access Server (NAS), co-located with a NAT device on a BNG, operates as a RADIUS client. The NAT device that resides on the BNG performs a single NAT (or firewall) function such as NAT44, NAT64, etc.

When the user sends a service request, the NAS on the BNG sends a RADIUS Access-Request message to the RADIUS server, requesting

authentication. Once the RADIUS server receives the request, it validates the sending client and if the request is approved, the RADIUS server replies with an Access-Accept message including a list of attribute-value pairs that describe the parameters to be used for this connection, including the port forwarding mapping specifically configured for the user.

When the RADIUS Access-Accept message arrived at the BNG, the port mapping information is used to create a pinhole on the NAT, along with the associated pinhole internal IP address, and also the external IP address, when it becomes available, for the specific user. A service granted message is then sent to the user, and after that point, IP packets from application initiated from network side (e.g., web cam) can be "port forwarded" by the NAT on the BNG to the user that is behind the NAT. IP packets belonging to the same flow but on opposite direction also traverse the same pinhole.

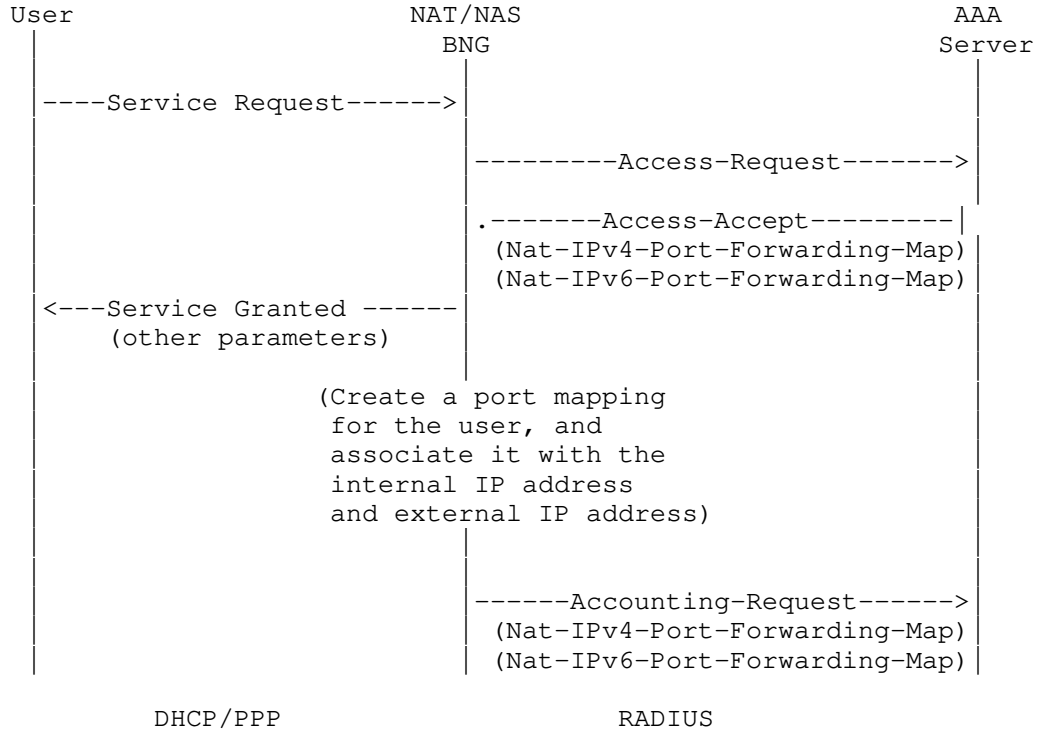


Figure 1: RADIUS Message Flow

When an IP packet travels from behind the NAT outwards (outbound), the NAT must change the source port number, i.e., the internal port to the configured external port, and when an IP packet travels from outside to the inside of the NAT (inbound), the NAT must change the target port number, i.e., the configured external port to the internal port.

Note that the service request that is initiated by a user can be associated with a PPP session or relevant DHCPv4/DHCPv6 message, with the same communication sequence between the RADIUS server and the NAS, and the installation of the port mapping on the NAT. Also, there may be different mechanisms as how an internal IP address and an external IP address (in the context of the NAT) assigned or determined, respectively, on the NAT for a specific user, but the forwarding port mapping information will remain the same as configured on the RADIUS server and is bonded to the specific user with one of its specific IP address.

A port mapping, once created on the NAT, will remain permanently in the duration of the user's Internet connection. When the connection is torn down, the mapping on the NAT must then be removed accordingly.

In the NAT444 scenario, in order to allow an IPv4 packet generated from outside of the BNG reaching the user, a forwarding port mapping is required on the NAT residing on the BNG as described above, but a separate forwarding port mapping is required on the user, typically a CPE, and in addition, the two sets of mapping need to be coordinated, so that an inbound IP packet, i.e., from outside of the BNG destined to the user, will successfully traverse two NATs before arriving at the user. The required mechanism for the NAT444 case is out of the scope of this document.

4. RADIUS Attributes

Two new RADIUS attributes are defined in this document, for IPv4 address and IPv6 address as the NAT pinhole internal address, respectively.

NAT-IPv4-Forwarding-Port-Map Attribute (Figure-2)
NAT-IPv6-Forwarding-Port-Map Attribute (Figure-3)

Description

Both of the two attributes contain a 16-bit Internal Port that identifies the source TCP/UDP port number of an IP packet sent by the user, or the destination port number of an IP packet destined

to the user, and in both cases, the IP packet travels behind the NAT device. Also they contain a 16-bit Configured External Port that identifies the source TCP/UDP port number of an IP packet sent by the user, or the destination port number of an IP packet destined to the user, and in both cases, the IP packet travels outside of the NAT device. In addition, the two attributes contain a 32-bit IPv4 address or 128-bit IPv6 address, respectively, as their respective NAT pinhole's internal IP address. Together, the port pair and IP address determine the port mapping rule for a specific IP flow that traverses a NAT device.

The attribute MAY appear in an Access-Accept packet, and may also appear in an Accounting-Request packet. In either case, the attribute MUST NOT appear more than once in a single packet.

Neither of these attributes MUST NOT appear in any other RADIUS packets.

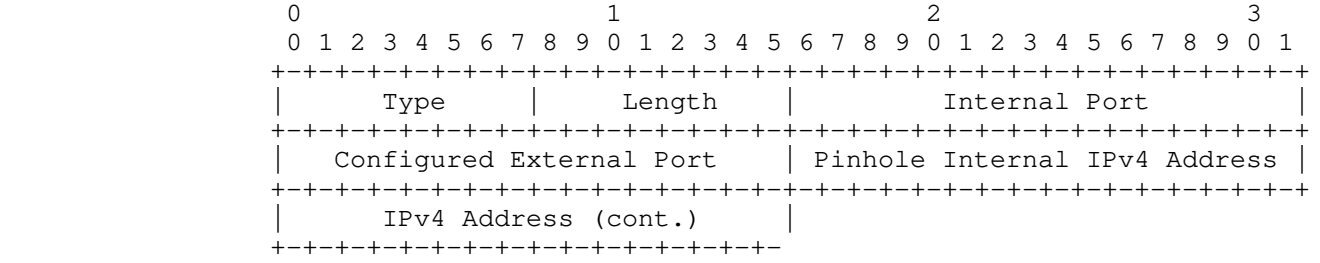


Figure-2 Nat-IPv4-Forwarding-Port-Map Attribute Format

These fields are described below:

Type

Type for NAT-IPv4-Forwarding-Port-Map (value is TBD)

Length

8 octets

Internal Port

Internal port for the pinhole

Configured External Port

External port for the pinhole

Pinhole IPv4 Address

The internal IPv4 address at the pinhole

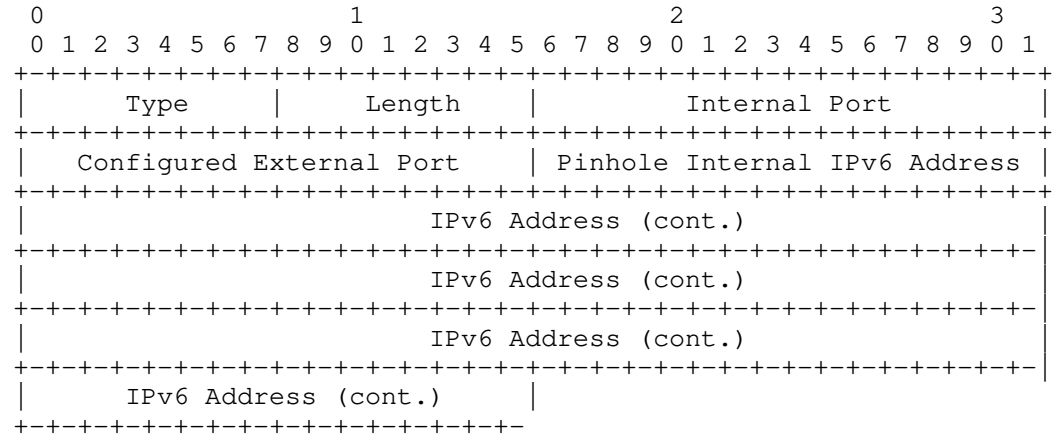


Figure-3 Nat-IPv6-Forwarding-Port-Map Attribute Format

These fields are described below:

Type

Type for NAT-IPv6-Forwarding-Port-Map (value is TBD)

Length

20 octets

Internal Port

Internal port for the pinhole

Configured External Port

External port for the pinhole

Pinhole IPv6 Address

The internal IPv6 address at the pinhole

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-IPv4-Forwarding-Port-Map
0-1	0-1	0	0	0-1	TBD	NAT-IPv6-Forwarding-Port-Map

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.

6. Security

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

7. IANA Considerations

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

NAT-IPv4-Forwarding-Port-Map

NAT-IPv6-Forwarding-Port-Map

8. Acknowledgements

Thanks to Dan Wing who provided some useful 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.draft-ietf-pcp-base] Wing, D., "Port Control Protocol (PCP), draft-ietf-pcp-base-05, work in progress.

10. Authors' Addresses

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

Behavior Engineering for Hindrance Avoidance
Internet-Draft
Intended status: Informational
Expires: July 27, 2011

R. Penno
Juniper Networks
T. Saxena
Cisco Systems
M. Boucadair
France Telecom
S. Sivakumar
Cisco Systems
January 23, 2011

Analysis of 64 Translation
draft-ietf-behave-64-analysis-01

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 27, 2011.

Copyright Notice

Table of Contents

1. Introduction

1.1. Definition

This document uses 64 proposal (or 64 for short) to refer to the mechanisms defined in the following documents:

- Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers [I-D.ietf-behave-v6v4-xlate-stateful]
- DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers [I-D.ietf-behave-dns64]
- IPv6 Addressing of IPv4/IPv6 Translators [RFC6052]
- Framework for IPv4/IPv6 Translation [I-D.ietf-behave-v6v4-framework]

1.2. Context

The current 64 proposal is widely seen as a major interconnection technique designed to enable 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. Issues listed in [RFC4966] are classified into three categories:

1. Problems impossible to solve;

2. Problems which can be solved.

3. Problems solved.

2.1. Problems Impossible to Solve

Problems discussed in [RFC4966], which are impossible to solve:

1. 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 (Section 2.2 of [RFC4966]).

   Analysis: This issue is not specific to 64 but to all NAT-based solutions.

2. Loss of information due to incompatible semantics between IPv4 and IPv6 versions of headers and protocols (Section 2.4 of [RFC4966]).

   Analysis: This issue is not specific to 64 but due to the design of IPv4 and IPv6.

3. Need for the NAT64-capable device to act as proxy for correspondent node when IPv6 node is mobile, with consequent restrictions on mobility (Section 2.7 of [RFC4966]).

   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.

2.2. Problems Which Can be Solved

Problems discussed in [RFC4966], which can be solved:

1. Disruption of all protocols that embed IP addresses (and/or ports) in packet payloads or apply integrity mechanisms using IP addresses (and ports) (Section 2.1 of [RFC4966]).

   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. Interaction with SCTP [RFC4960] and multihoming (Section 2.6 of [RFC4966]).

   Analysis: SCTP is out of scope of 64. Only TCP and UDP transport protocols are within the scope of 64.

3. Inability to handle multicast traffic (Section 2.8 of [RFC4966]).

   Analysis: This problem is not addressed by the current 64 specifications.

4. Scalability concerns together with introduction of a single point of failure and a security attack nexus (Section 3.2 of [RFC4966]).

   Analysis: This is not specific to NAT64 but to all stateful NAT flavors.

5. Creation of a DoS (Denial of Service) threat relating to exhaustion of memory and address/port pool resources on the translator (Section 3.4 of [RFC4966]).

   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.

6. Restricted validity of translated DNS records: a translated record may be forwarded to an application that cannot use it (Section 4.2 of [RFC4966]).

   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).

7. 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 (Section 4.5 of [RFC4966]).

   Analysis: This is not specific to NAT64 but to all NAT flavours.

8. Address selection issues when either the internal or external hosts implement both IPv4 and IPv6 (Section 4.1 of [RFC4966]).

   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].

2.3. Problems Solved

Problems, identified in [RFC4966], which are solved:

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. Need for additional state and/or packet reconstruction in dealing with packet fragmentation. Otherwise, implement no support for fragments. (Section 2.5 of [RFC4966])

   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).

3. Inappropriate translation of responses to A queries from IPv6 nodes (Section 4.3 of [RFC4966]).

   Analysis: DNS64 [I-D.ietf-behave-dns64] does not resolve A queries.

4. Address selection issues and resource consumption in a DNS-ALG with multi-addressed nodes (Section 4.4 of [RFC4966]).

   Analysis: Since the DNS-ALG is not there, there is no need to maintain temporary states in anticipation of connections. Note that static binding are required to allow for communications initiated from an IPv4 client to an IPv6 server.

5. Limitations on DNS security capabilities when using a DNS-ALG (Section 2.5 of [RFC4966]).

   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.

6. Creation of a DoS (Denial of Service) threat relating to exhaustion of memory and address/port pool resources on the translator (Section 3.4 of [RFC4966]).

   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.

7. Requirement for applications to use keepalive mechanisms to workaround connectivity issues caused by premature timeout for session table and BIB entries (Section 2.3 of [RFC4966]).

   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.

8. 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. (Section 3.3 of [RFC4966])

   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	--
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	Yes
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

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.

Table of Contents

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:

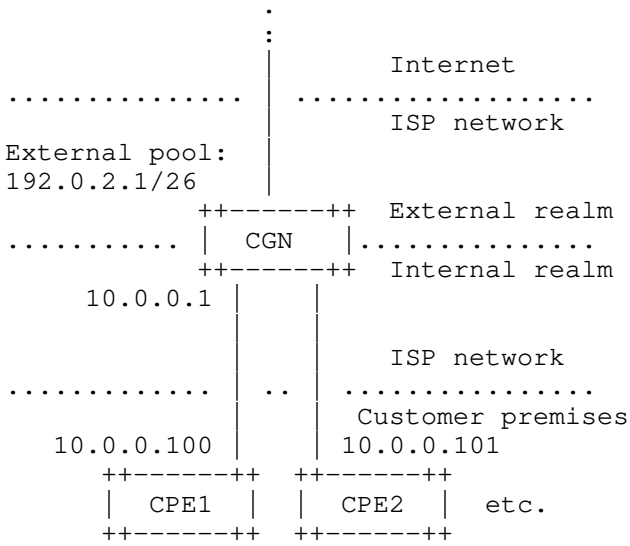
Carrier-Grade NAT (CGN): A NAT-based [RFC2663] logical function used to share the same IPv4 address among several subscribers. A CGN is not managed by the subscribers.

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.

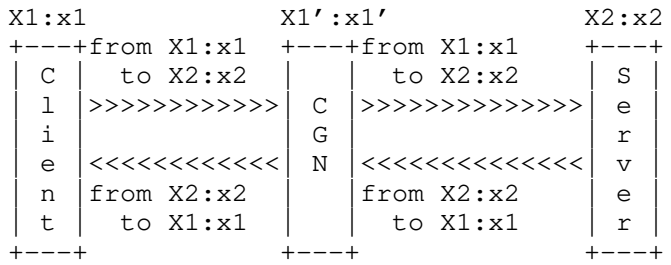


Figure 2: CGN pass-through

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):

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

Table of Contents

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".

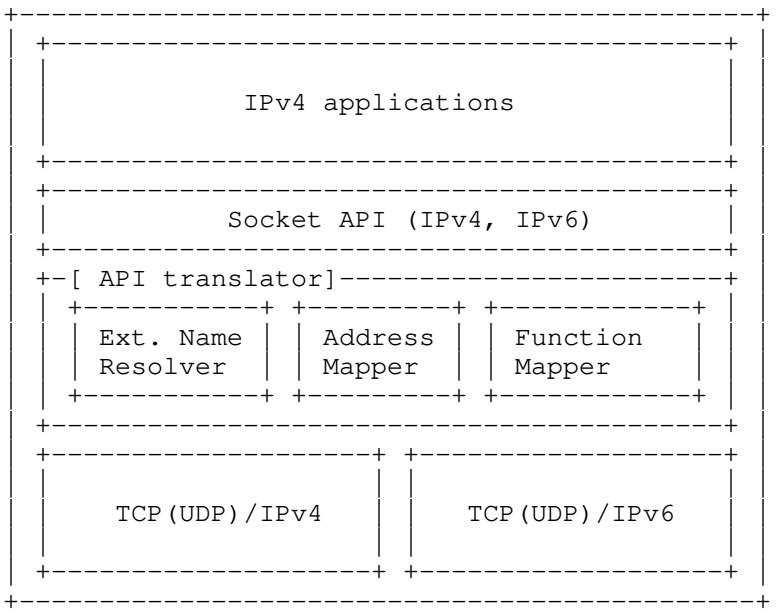


Figure 1: Architecture of a dual stack host using protocol translation at socket layer

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".

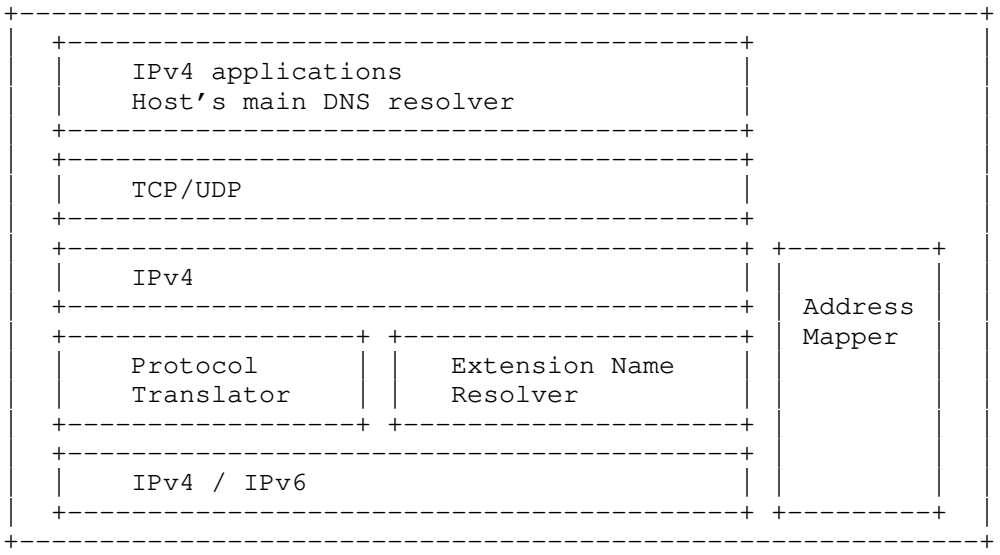


Figure 2: Architecture of a dual-stack host using protocol translation at the network layer

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 mathching 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.

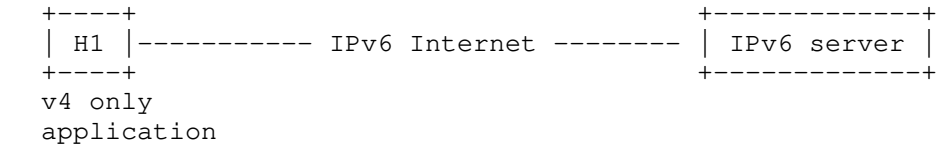


Figure 4: Network Scenario #1

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.

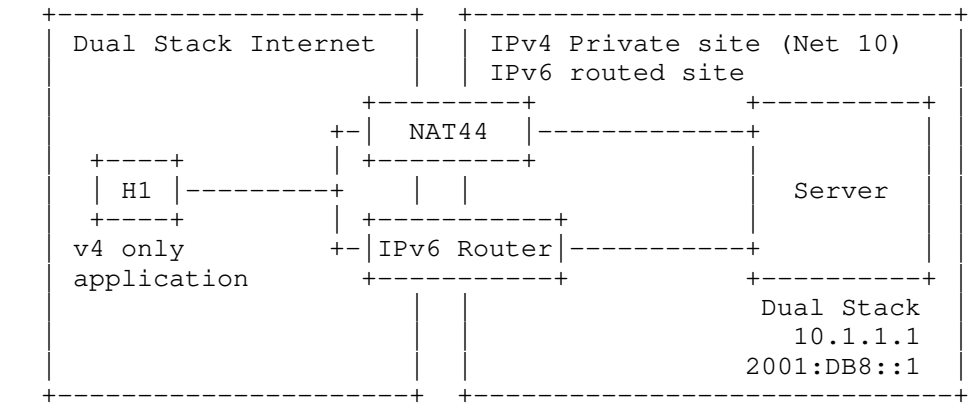
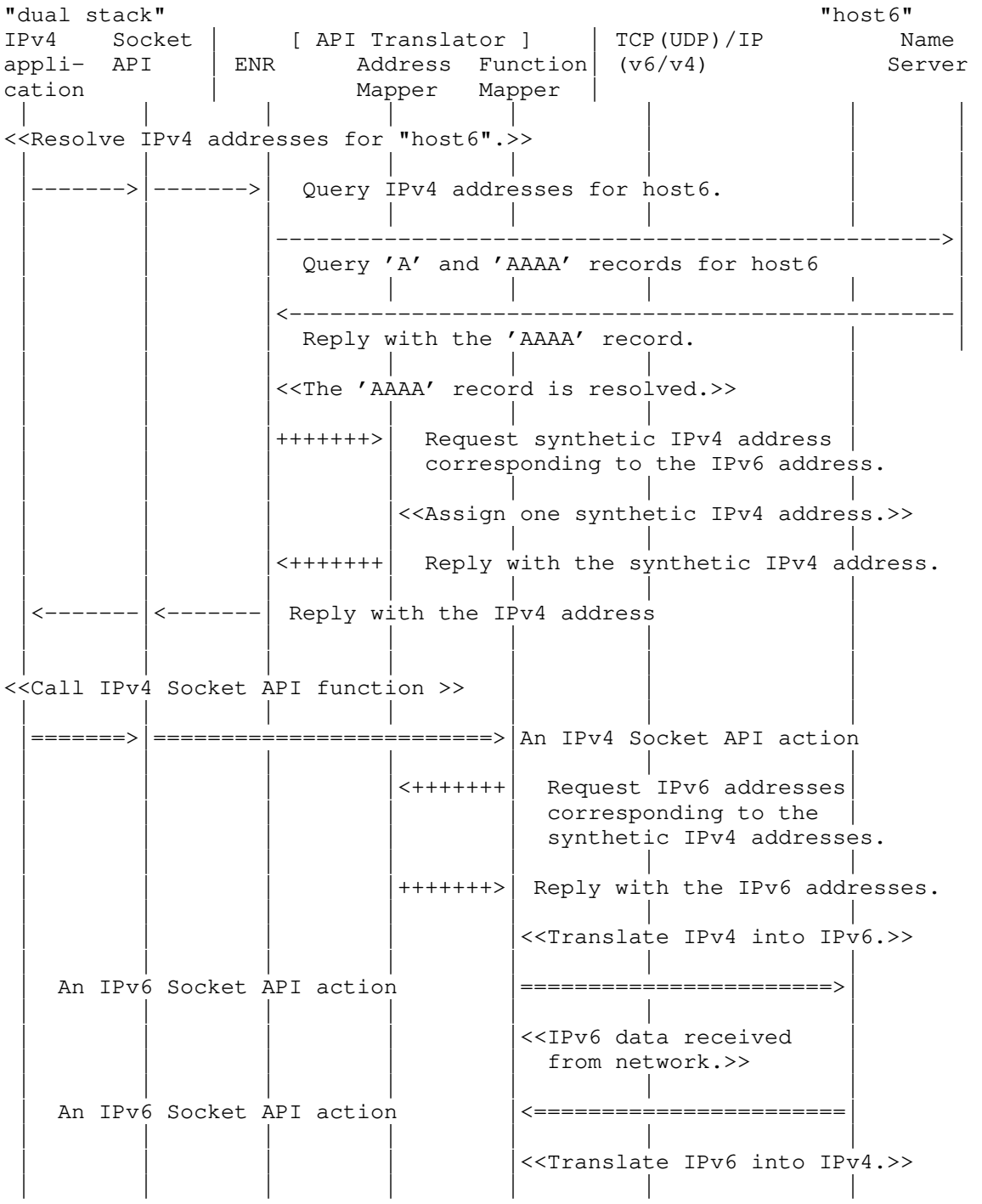


Figure 5: Network Scenario #2

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.


"dual stack"
IPv4 appli- cation
Socket API
ENR
[API Translator]
Address Mapper
Function Mapper
TCP(UDP)/IP (v6/v4)
"host6" Name Server
<<Resolve IPv4 addresses for "host6".>>
Query IPv4 addresses for host6.
Query 'A' and 'AAAA' records for host6
Reply with the 'AAAA' record.
<<The 'AAAA' record is resolved.>>
Request synthetic IPv4 address corresponding to the IPv6 address.
<<Assign one synthetic IPv4 address.>>
Reply with the synthetic IPv4 address.
Reply with the IPv4 address
<<Call IPv4 Socket API function >>
An IPv4 Socket API action
Request IPv6 addresses corresponding to the synthetic IPv4 addresses.
Reply with the IPv6 addresses.
<<Translate IPv4 into IPv6.>>
An IPv6 Socket API action
<<IPv6 data received from network.>>
An IPv6 Socket API action
<<Translate IPv6 into IPv4.>>

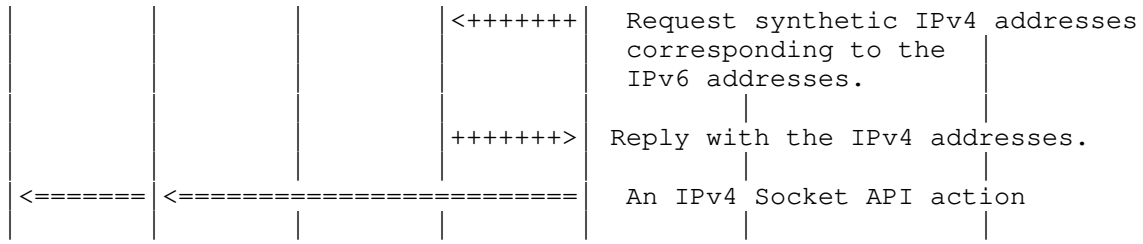
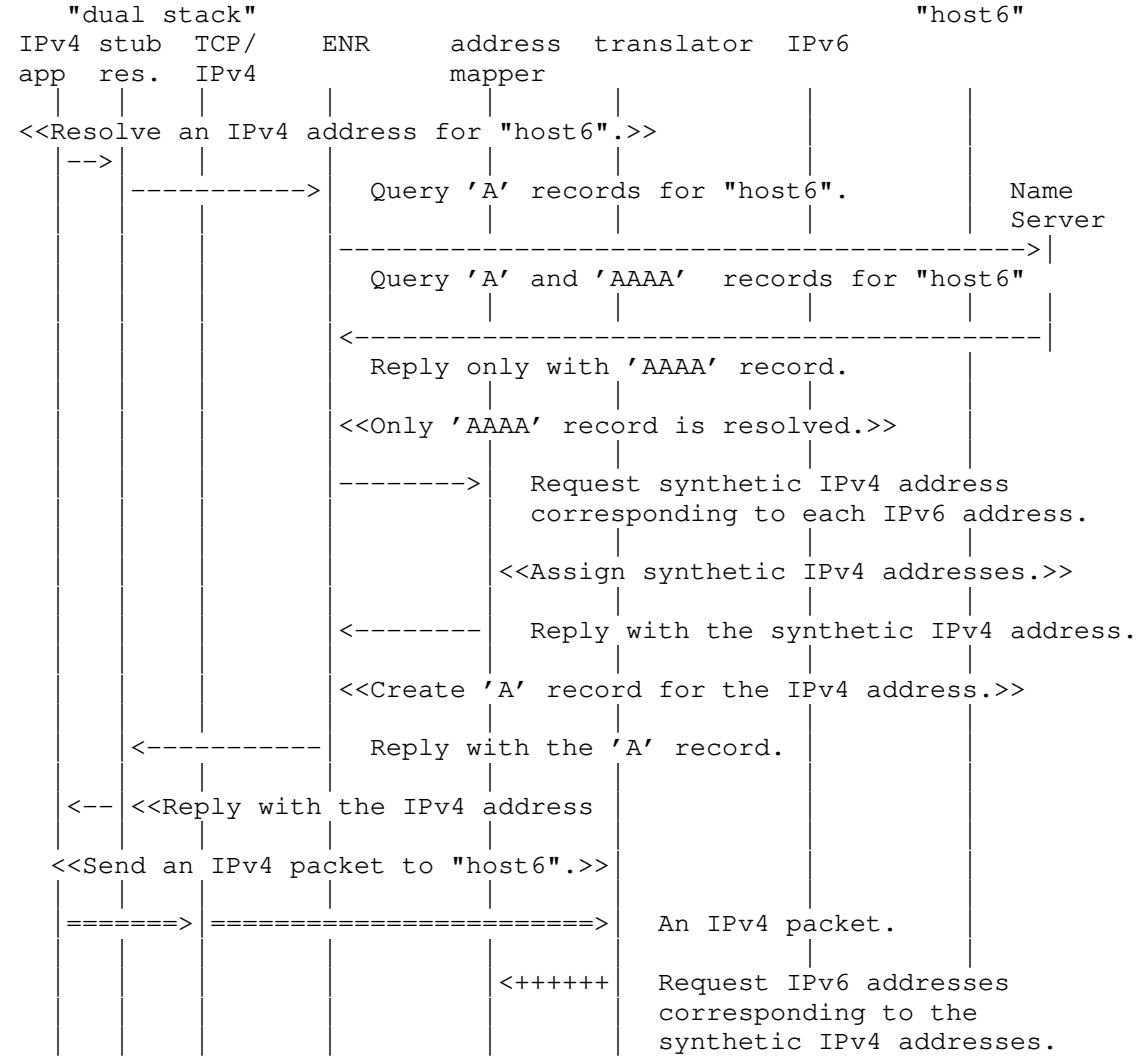


Figure 6: Example of BIH as API addition



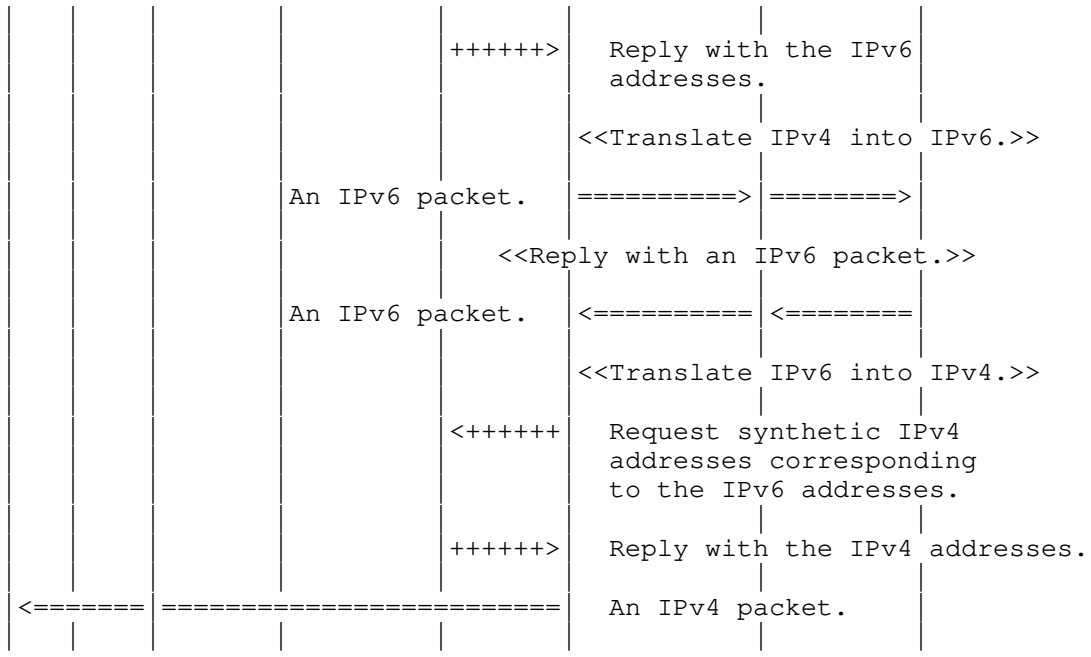


Figure 7: Example of BIH at the network layer

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 Jankiewizh, 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:

- getsocketopt()
- setsocketopt()

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

Network Working Group
Internet-Draft
Updates: 4008 (if approved)
Intended status: Standards Track
Expires: September 5, 2011

J. Dionne
Viagenie Inc.
M. Blanchet
Viagenie

March 4, 2011

NAT64 Management Information Base (MIB)
draft-jpdionne-behave-nat64-mib-00

Abstract

This memo describes the NAT64 Management Information Base. It reuses the NAT-MIB and add specific NAT64 variables.

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 September 5, 2011.

Table of Contents

1. Introduction

To enable IPv6-only nodes to access to IPv4-only nodes, a NAT64 [I-D.ietf-behave-v6v4-xlate-stateful] is used. This NAT64 requires a Management Information Base (MIB) for the purpose of managing the device.

The NAT-MIB [RFC4008] is designed to carry translation from any address family to any address family, therefore supports IPv6 to IPv4 translation such as NAT64.

Some specifics of NAT64 are not present in the NAT-MIB. This document describes the changes to fully support NAT64.

2. Modifications to NAT-MIB

A revision of [RFC4008] is proposed.

2.1. Pref64

The NAT64 uses a special prefix, named Pref64 [RFC6052], to carry IPv4 destination addresses into an IPv6 address. This prefix is required for the joint operation of NAT64 and DNS64[I-D.ietf-behave-dns64]. The operator can choose to use the Well-Known prefix (64:ff9b::/96) or a Network-Specific Prefix.

We propose to include this prefix to the NatAddrMapEntry. For other type of NATs, this value will be ignored and have a null value (::/0)

3. Configuring NAT64

This section presents considerations regarding NAT64 that doesn't require modification of the NAT-MIB.

3.1. NatTranslationEntity

NatTranslationEntity is: "An indication of a) the direction of a session for which an address map entry, address bind or port bind is applicable, and b) the entity (source or destination) within the session that is subject to translation."

For NAT64, both the source and destination is translated. Hence, two bits must be set:

"inboundSrcEndPoint" and "inboundDstEndPoint" if the operator chooses to configure the NAT64 address map entry on the IPv4 network interface.

or

"outboundSrcEndPoint" and "outboundDstEndPoint" if the operator chooses to configure the NAT64 address map entry on the IPv6 Internet interface.

3.2. Local and Private Terminology

Throughout [RFC4008] "private" and "public" are used interchangeably with "local" and "global" to refer to the networks the translator is operating on. "local" refer to the network where the NAT sessions are initiated. With NAT64, The IPv6 network corresponds to the "local" side and the "IPv4 Internet to the "global" side.

4. RFC4008 Proposed Changes

4.1. Changes to Section 3: Terminology

NAT Session:

"NAT sessions in the document are restricted to sessions based on TCP and UDP only." Add "ICMP". No changes needs to be made to the MIB definition ICMP can be specified as a protcocol session.

The "session" terminology also differs with [I-D.ietf-behave-v6v4-framework] definition: An "end-to-end session" in rfc4008 is a "session" in xlate.

4.2. Changes to Section 4: Overview

"The address map entry also identifies the end-point of the session that must be subject to translation.". For NAT64, this sentence implies the address map entry would contain pref64:: destination address. However, the real end-point is the ipv4 destination address.

4.2.1. Changes to Section 4.2: natAddrMapTable

Add a paragraph for pref64 variables. Refer to Section 4.3.2

Rephrase the last sentence to explain how to use NatTranslationEntity to describe NAT64.

4.2.2. Changes to Section 4.5: natSessionTable

"Session" terminology differs with draft-ietf-behave-v6v4-xlate-stateful's definition. A "end-to-end session" in rfc4008 is a "session" in xlate.

4.2.3. Changes to Section 4.7: Notifications

"natPacketDiscard notifies the end user/manager of packets being discarded due to lack of address mappings.". We must add: "or incompatibilities in IPv4/IPv6 translation.

4.2.4. Changes to Section 4.9: Configuration via the MIB

TBD: Add a point about how to configure the pref64 if its NAT64.

4.3. Changes to Section 5: Definitions

4.3.1. IMPORT

```
   Ipv6AddressPrefix,
           FROM IPV6-TC;
```

4.3.2. Update to NatAddrMapEntry

```
natAddrMapEntry OBJECT-TYPE
    SYNTAX      NatAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This entry represents an address map to be used for
             NAT and contributes to the dynamic and/or static
             address mapping tables of the NAT device."
    INDEX   { ifIndex, natAddrMapIndex }
    ::= { natAddrMapTable 1 }

NatAddrMapEntry ::= SEQUENCE {
    natAddrMapIndex                 NatAddrMapId,
    natAddrMapName                  SnmpAdminString,
    natAddrMapEntryType             NatAssociationType,
    natAddrMapTranslationEntity     NatTranslationEntity,
    natAddrMapLocalAddrType         InetAddressType,
    natAddrMapLocalAddrFrom         InetAddress,
    natAddrMapLocalAddrTo           InetAddress,
    natAddrMapLocalPortFrom         InetPortNumber,
    natAddrMapLocalPortTo           InetPortNumber,
    natAddrMapGlobalAddrType        InetAddressType,
    natAddrMapGlobalAddrFrom        InetAddress,
    natAddrMapGlobalAddrTo          InetAddress,
    natAddrMapGlobalPortFrom        InetPortNumber,
    natAddrMapGlobalPortTo          InetPortNumber,
    natAddrMapProtocol              NatProtocolMap,
    natAddrMapInTranslates          Counter64,
    natAddrMapOutTranslates         Counter64,
    natAddrMapDiscards              Counter64,
    natAddrMapAddrUsed              Gauge32,
    natAddrMapStorageType           StorageType,
    natAddrMapRowStatus             RowStatus
    natAddrMapPref64                Ipv6AddressPrefix
    natAddrMapPref64Length          INTEGER (0..128),
}
```

```
natAddrMapPref64 OBJECT-TYPE
    SYNTAX      Ipv6AddressPrefix
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the IPv6 address part of the Pref64
             for NAT64. For a NAT64, the default value is the Well-known
             prefix 64:ff9b::.  For other NAT types, the prefix is ::.
             The associated prefix length is found in
             natAddrMapPref64Length.
    ::= { natAddrMapEntry 22 }

natAddrMapPref64Length OBJECT-TYPE
    SYNTAX      INTEGER (0..128)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
      "The prefix length of natAddrMapPref64.  For NAT64, allowed values
       are: 32, 40, 48, 56, 64 and 96.".  For other NAT types, value is
       0."
    ::= { natAddrMapEntry 23 }
```

4.4. Complete updated NAT-MIB

Update is done in conformance to section 10 of [RFC2578].

TBD

5. Security Considerations

No additionnal security issues. See [RFC4008] section 7.

6. References

6.1. Normative References

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[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.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

6.2. Informative 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-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.

Authors' Addresses

Jean-Philippe Dionne
Viagenie Inc.
2875 boul. Laurier, suite D2-630
Quebec
Canada

Phone: +1-418-656-9254
Email: jean-philippe.dionne@viagenie.ca

Marc Blanchet
Viagenie
2875 boul. Laurier, suite D2-630
Quebec, QC G1V 2M2
Canada

Email: Marc.Blanchet@viagenie.ca
URI: http://viagenie.ca

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.

Table of Contents

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:

- 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.

- 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.

- Multicast translations [I-D.venaas-behave-mcast46][I-D.venaas-behave-v4v6mc-framework].

- 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.

- 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 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 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.

+ 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 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.

+ 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-NATPR 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 ENDS0 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 does 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 synthetization and hence using DNS also for delivering the synthetization information sounds logical. The fact that the synthetization 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 ENDS0 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

Behave WG
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2011

J. Korhonen
Nokia Siemens Networks
T. Savolainen, Ed.
Nokia
February 17, 2011

EDNS0 Option for Indicating AAAA Record Synthesis and Format
draft-korhonen-edns0-synthesis-flag-02.txt

Abstract

Advanced hosts and applications benefit of the knowledge of an IPv6 address, AAAA record, synthesis taking place in the network. This draft proposes new ENDS0 option for communicating the synthesis is taking place, used address format, and the IPv6 prefix and suffix used by the DNS64. The communicated information enables hosts to perform 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 21, 2011.

Table of Contents

1. Introduction

As the networks transition to IPv6, connectivity to IPv4-only domains have to be provided. NAT64 [I-D.ietf-behave-v6v4-xlate-stateful] and DNS64 [I-D.ietf-behave-dns64] technologies can be utilized to make IPv4-only peers look like being reachable over IPv6. The DNS64 utilizes IPv6 address synthesis to create local IPv6 presentations of peers having only IPv4 addresses. Applications utilizing DNS for resolving peers' IPv6 addresses can work seamlessly through protocol translation taking place at NAT64.

The DNS64 cannot serve applications not using DNS, such as those receiving IPv4 addresses as referrals. Such applications could nevertheless be able to work through NAT64, provided they are able to create locally valid IPv6 presentations of peers' IPv4 addresses.

This document describes a method for advanced applications to learn the information required to perform local IPv6 address synthesis.

The knowledge of IPv6 address synthesizing taking place may also be useful if DNS64 is present in dual-stack network access. In such cases hosts may choose to use IPv4 addresses instead of synthesized IPv6 addresses, and hence avoid traversal through NAT64.

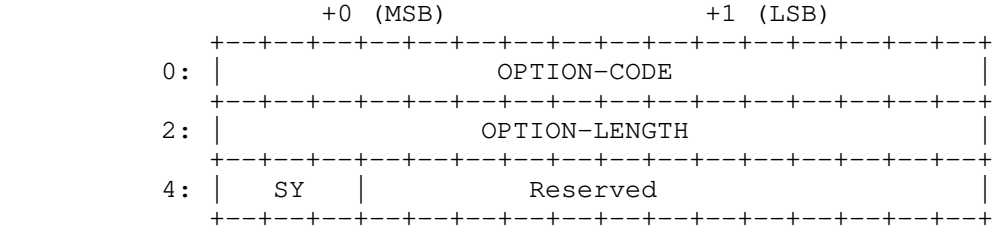
2. Requirements and Terminology

2.1. Requirements

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].

3. EDNS0 option for indicating address synthesis

The mechanism for informing AAAA record synthesis taking place and the used addressing format is communicated in an EDNS0 option in a DNS response. The option has three bits indicating the formats described in [RFC6052]. The option and bits are structured as follows:



OPTION-CODE (Assigned by IANA)

OPTION-LENGTH 2 (Length of payload in octets)

Possible values for SY-bits are:

000 reserved
001 prefix length /32
010 prefix length /40
011 prefix length /48
100 prefix length /56
101 prefix length /64
110 prefix length /96
111 address is not synthesized

Reserved Initialized to zero

Figure 1

The prefix length corresponds to the address formats documented in "IPv6 Addressing of IPv4/IPv6 Translators document" [RFC6052] section 2.2.

Absence of EDNS0 option means that either no synthesis took place or the DNS64 does not support this specification. Either way, when the EDNS0 option is missing, the host cannot conclude for certain whether the AAAA response was synthesized or not. The host may additionally utilize method described in [I-D.savolainen-heuristic-nat64-discovery].

4. Host behavior

If a host requires information for local IPv6 address synthesis or NAT64 avoidance, it shall send a DNS query for AAAA record of a well-known IPv4-only fully qualified domain name. This well-known name does not have to be in global DNS system. It is enough that DNS64 recognizes the name and replies to it.

The host may query for well-known IPv4-only name, for example, at the moment the host is configured an IPv6 address of a DNS server. This may also happen at the time first DNS query for AAAA record is initiated.

When sending AAAA query for the known name a host MUST set "Checking Disabled (CD)" bit to zero, as otherwise the DNS64 will not perform IPv6 address synthesis hence does not reveal the IPv6 prefix(es) used for protocol translation.

If a host receives negative reply, it learns there are no NAT64 in the network.

A DNS reply with one or more non-empty AAAA records indicates that the access network is utilizing IPv6 address synthesis. The host reads the flag values on the ENDS0 option to learn the used address format, and with that information fetches from the received IPv6 addresses the information used by the network for IPv6 address synthesis (prefix, suffix, u-bit). The host MUST look through all of the received AAAA records to collect all available prefixes. The prefixes may include Well-Known Prefix or one or more Network-Specific Prefixes.

In the case only one IPv6 prefix was present in the DNS response: a host shall use that IPv6 prefix for both local synthetization and for detecting synthesis done by the DNS64 entity on the network.

In the case multiple IPv6 prefixes were present in the DNS response: a host SHOULD use all received prefixes when determining whether other received IPv6 addresses are synthetic. However, for selecting prefix for the local IPv6 address synthesis host MUST use the following prioritization order, of which purpose is to avoid use of prefixes containing suffixes reserved for the future [RFC6052]:

1. Use NSP having /96 prefix

2. Use WKP prefix

3. Use longest available NSP prefix

In the case of NXDOMAIN or empty AAAA reply: the DNS64 is not available on the access network, network filtered the well-known AAAA query on purpose, or something went wrong in the DNS resolution. All unsuccessful cases result in unavailability of a host to perform local IPv6 address synthesis. The host MAY periodically resend AAAA query to check if DNS64 has become available or temporary problem cleared. The host MAY also continue monitoring DNS replies with IPv6 addresses constructed from WKP, in which case the host MAY use the

WKP as if it were learned during the query for well-known name.

The information required for local IPv6 address synthesis should be made available for applications to utilize.

Alternatively, the host may learn the required information for the local IPv6 address synthesis or the NAT64 avoidance along with any normal DNS query for an AAAA record. In that case all the above considerations and procedures apply, except for the fact that the fully qualified domain used for the DNS query may or may not be provisioned with an AAAA record. Therefore, if and when the EDNS0 option is absent in the reply, the host cannot reliably determine whether the returned IPv6 address is real or synthesized.

5. Security Considerations

No security considerations have been identified.

6. IANA Considerations

IANA should define a name and an IPv4 address for a well-known IPv4-only name.

IANA should allocate new OPTION-CODE for this option.

7. Acknowledgements

The authors would like to acknowledge Andrew Sullivan for presenting general idea of ENDS0 option and SY-bit in behave WG mailing list.

8. References

8.1. 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.

8.2. Informative References

[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.

Authors' Addresses

Jouni Korhonen
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

Behave WG
Internet-Draft
Intended status: Informational
Expires: August 19, 2011

T. Savolainen
Nokia
J. Korhonen
Nokia Siemens Networks
February 15, 2011

Discovery of a Network-Specific NAT64 Prefix using a Well-Known Name
draft-savolainen-heuristic-nat64-discovery-01.txt

Abstract

This document describes a method for detecting presence of DNS64 and for learning IPv6 prefix used for protocol translation on an access network without explicit support from the access network. The method depends on existence of a known IPv4-only domain name. The information learned enables applications and hosts to perform local IPv6 address synthesis and on dual-stack accesses avoid traversal through NAT64.

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 19, 2011.

Copyright Notice

Table of Contents

1. Introduction

As part of the transition to IPv6 NAT64 [I-D.ietf-behave-v6v4-xlate-stateful] and DNS64 [I-D.ietf-behave-dns64] technologies will be utilized by some access networks to provide IPv4 connectivity for IPv6-only hosts. The DNS64 utilizes IPv6 address synthesis to create local IPv6 presentations of peers having only IPv4 addresses, hence allowing DNS-using IPv6-only hosts to communicate with IPv4-only peers.

However, DNS64 cannot serve applications not using DNS, such as those receiving IPv4 address literals as referrals. Such applications could nevertheless be able to work through NAT64, provided they are able to create locally valid IPv6 presentations of peers' IPv4 addresses.

Additionally, DNS64 is not able to do IPv6 address synthesis for hosts running validating DNSSEC enabled resolvers, but instead the synthetization must be done by the hosts. In order to perform IPv6 synthesis hosts have to learn the IPv6 prefix(es) used on the access network for protocol translation.

This document describes a best effort method for advanced applications and hosts to learn the information required to perform local IPv6 address synthesis. An example application is a browser encountering an IPv4 address literal in an IPv6-only access network. Another example is a host running validating security aware DNS resolver.

The knowledge of IPv6 address synthetization taking place may also be useful if DNS64 and NAT64 are present in dual-stack enabled access network. In such cases hosts may choose to prefer IPv4 in order to avoid traversal through protocol translators.

The described method is intented for the scenarios where network assisted NAT64 and prefix discovery solutions are not available.

2. Requirements and Terminology

2.1. Requirements

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].

2.2. Terminology

Known Name: a fully qualified domain name known by the implementation to have only an A record. Implementation knows it by hard-coding or e.g. via some provisioning technology. The name is not known by everybody.

Well-Known IPv4-only Name: a fully qualified domain name well-known to have only A record.

3. Host behavior

A host requiring information about presence of NAT64 and the IPv6 prefix used for protocol translation shall send a DNS query for AAAA records of a known IPv4-only fully qualified domain name. This may happen, for example, at the moment the host is configured an IPv6 address of a DNS server. This may also happen at the time when first DNS query for AAAA record is initiated. The host may perform this check in both IPv6-only and dual-stack access networks.

When sending AAAA query for the known name a host MUST set "Checking Disabled (CD)" bit to zero, as otherwise the DNS64 will not perform IPv6 address synthesis hence does not reveal the IPv6 prefix(es) used for protocol translation.

A DNS reply with one or more non-empty AAAA records indicates that the access network is utilizing IPv6 address synthesis. A host MUST look through all of the received AAAA records to collect all available prefixes. The prefixes may include Well-Known Prefix or one or more Network-Specific Prefixes. In the case of NSPs the host SHALL search for the IPv4 address inside of the received IPv6 addresses to determine used address format.

An IPv4 address inside synthesized IPv6 address should be found at some of the locations described in [RFC6052]. If the searched IPv4 address is not found on any of the standard locations the network must be using different formatting. In such case the host may try to find out the IPv4 address at some other location.

The host should ensure a 32-bit IPv4 address value is present only once in an IPv6 address. In case another instance of the value is found inside the IPv6, the host shall repeat the search with another IPv4 address.

In the case only one IPv6 prefix was present in the DNS response: a host shall use that IPv6 prefix for both local synthetization and for detecting synthesis done by the DNS64 entity on the network.

In the case multiple IPv6 prefixes were present in the DNS response: a host SHOULD use all received prefixes when determining whether other received IPv6 addresses are synthetic. However, for selecting prefix for the local IPv6 address synthesis host MUST use the following prioritization order, of which purpose is to avoid use of prefixes containing suffixes reserved for the future [RFC6052]:

1. Use NSP having /96 prefix

2. Use WKP prefix

3. Use longest available NSP prefix

In the case of NXDOMAIN or empty AAAA reply: the DNS64 is not available on the access network, network filtered the well-known AAAA query on purpose, or something went wrong in the DNS resolution. All unsuccesful cases result in unavailability of a host to perform local IPv6 address synthesis. The host MAY periodically resend AAAA query to check if DNS64 has become available or temporary problem cleared. The host MAY also continue monitoring DNS replies with IPv6 addresses constructed from WKP, in which case the host MAY use the WKP as if it were learned during the query for well-known name.

3.1. Connectivity test

After the host has obtained a candidate prefix and format for the IPv6 address synthesis it may locally synthesize an IPv6 address, by using a publicly routable IPv4 address, and test connectivity with the resulting IPv6 address. The connectivity test may be conducted e.g. with ICMPv6 or with a transport layer protocol. The used public IPv4 address may be learned via separate A query.

This connectivity test ensures local address synthetization results in functional and protocol translatable IPv6 addresses.

3.2. IPv4 addresses of the known name

The IPv4 addresses of the known name should be such that they are unlikely to appear more than once within an IPv6 address and also as easy as possible to find from within the synthetic IPv6 address. Good addresses might be 127.127.127.127 as a primary and 192.168.127.254 as a secondary. The secondary address is needed in the case multiple instaces of primary address are present in a synthetic IPv6 address. The IPv4 addresses can, however, be publicly routable especially if also used for the connectivity test.

3.3. Non-standard IPv6 address formats

A node may need to perform more complex heuristics to cope with networks possibly using non-standard IPv6 address formats. Non-standard approaches might include for example:

1. Non-standard location: IPv4 address in one piece at non-standard location. Can be found by pattern matching.

2. Fragmented: IPv4 address in multiple pieces around the IPv6 address. May be found by pattern matching.

3. Obfuscated address: IPv4 address is obfuscated, for example xorred. May potentially be found especially if standard addess format is used, but as this is an indication of access network's unwillingness to support host based synthetization the host should not try to decipher the IPv6 prefix.

4. Hosting of an IPv4-only name(s)

The required IPv4-only name has to be hosted by someone. While IANA(?) might host one (?), it may be safest for device, operating system, and/or application vendors to host IPv4-only names for their own uses. The name should have two A records in order to manage in situations where the first IPv4 address appears more than once within synthetic IPv6 address. Another name may be needed for connectivity test purposes.

5. Required IPv4 addresses

A prefix detection without connectivity test does not require any routable IPv4 addresses. The connectivity test requires a routable IPv4 address on the server side.

6. Security Considerations

No security considerations have been identified.

7. IANA Considerations

IANA(?) should define a name and an IPv4 address for a Well-Known IPv4-only Name.

8. Acknowledgements

Authors would like to thank Dan Wing, Washam Fan, Cameron Byrne, and Christian Huitema for improvement ideas and comments.

9. 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.

Authors' Addresses

Teemu Savolainen
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com

Jouni Korhonen
Nokia Siemens Networks
Linnoitustie 6
FI-02600 Espoo
Finland

Email: jouni.nospam@gmail.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

Table of Contents

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:

- 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.

- 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].

- 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:

- to optimize the resources usage of deployed NAT64 devices (e.g., several ingress/egress NAT64 devices);
- to avoid congesting a single NAT64 device while other free resources are still available;
- 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

Tow 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 NAPT 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 polices. 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 Xinxi 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 Xinxi 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 Xinxi Rd.,
Hai-Dian District, Beijing 100085
P.R. China

Email: zhangdacheng@huawei.com

Xiaohu Xu
Huawei Technologies Co.,Ltd
KuiKe Building, No.9 Xinxi 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