Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

E. Abdo
M. Boucadair
J. Queiroz
France Telecom
July 16, 2012

HOST_ID TCP Options: Implementation & Preliminary Test Results
draft-abdo-hostid-tcpopt-implementation-03

Abstract

This memo documents the implementation of the HOST_ID TCP Options. It also discusses the preliminary results of the tests that have been conducted to assess the technical feasibility of the approach as well as its scalability. Several HOST_ID TCP options have been implemented and tested.

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 17, 2013.

Table of Contents

1. Introduction

To ensure IPv4 service continuity, service providers will need to deploy IPv4 address sharing techniques. Several issues are likely to be encountered (refer to [RFC6269] for a detailed survey of the issues) and they may affect the delivery of services that depends on the enforcement of policies based upon the source IPv4 address.

Some of these issues may be mitigated owing to the activation of advanced features. Among the solutions analyzed in [I-D.boucadair-intarea-nat-reveal-analysis], the use of a new TCP option to convey a HOST_ID seems to be a promising solution.

This memo documents some implementation and experimentation efforts that have been conducted to assess the viability of using HOST_ID TCP options at large scale. In particular, this document provides experimentation results related to the support of the HOST_ID TCP Options, the behavior of legacy TCP servers when receiving the HOST_ID TCP options. This draft also discusses the impact of using a HOST_ID TCP options on the time it takes to establish a connection; it also tries to evaluate the impact of the new TCP options on the performance of the CGN. Finally it presents the enforcement policies that could be applied by remote servers based upon the HOST_ID options contents.

2. Objectives

The implementation of several HOST_ID TCP options is primarily meant to:

- Assess the validity of the HOST_ID TCP option approach
- Evaluate the impact on the TCP stack to support the HOST_ID TCP options
- Improve filtering and logging capabilities based upon the contents of the HOST_ID TCP option. This means the enforcement of various policies based upon the content of the HOST_ID TCP option at the server side: Log, Deny, Accept, etc.
- Assess the behavior of legacy TCP servers when receiving a HOST_ID TCP option
- Assess the success ratio of TCP communications when a HOST_ID TCP option is received
- Assess the impact of injecting a HOST_ID TCP option on the time it takes to establish a connection
- Assess the performance impact on the CGN device that has been configured to inject the HOST_ID option

3. NAT Reveal TCP Options: Overview

The original idea of defining a TCP option is documented in [I-D.wing-nat-reveal-option] and denoted as HOST_ID_WING.

An additional TCP option is also considered and denoted as HOST_ID_BOUCADAIR. The main motivation is to cover also the load-balancer use case and provide richer functionality as Forwarded-For HTTP header than HOST_ID_WING can provide.

The following sub-sections provide an overview of these HOST_ID TCP options.

3.1. HOST_ID_WING TCP Option

HOST_ID_WING is defined in [I-D.wing-nat-reveal-option]. Figure 1 shows the format of this option.

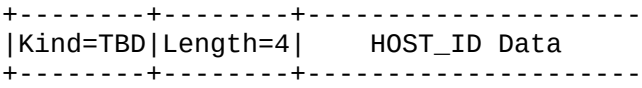


Figure 1: Format of HOST_ID_WING TCP Option

This option must be sent only upon the initial connection request, i.e., in SYN packets as shown in Figure 2

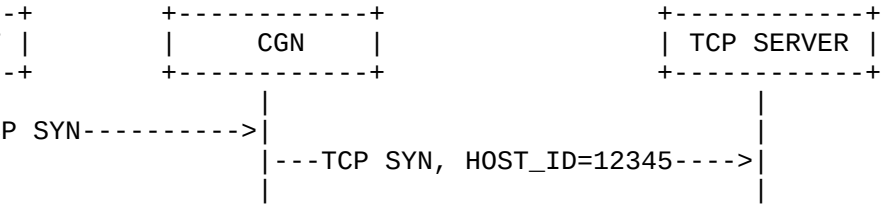


Figure 2: HOST_ID_WING TCP Option: Flow example

3.2. HOST_ID_BOUCADAIR TCP Option

As mentioned above, the HOST_ID_BOUCADAIR TCP Option is inspired from HOST_ID_WING and XFF.

The HOST_ID_BOUCADAIR option is a 10-byte long TCP option, where KIND, Length and lifetime-Origin fields fill one byte each, and HOST_ID data is 7-byte long as shown in Figure 3

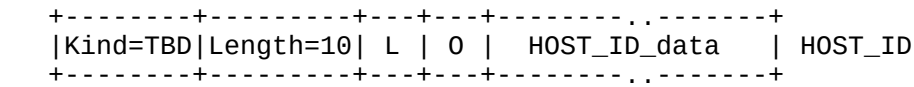


Figure 3: Format of HOST_ID_BOUCADAIR TCP option

- o L: Indicates the validity lifetime of the enclosed data (in the spirit of [RFC6250]). The following values are supported:
  0: Permanent;
  >0:Dynamic; this value indicates the validity time.
- o Origin: Indicates the origin of the data conveyed in the data field. The following values are supported:
  0: Internal Port
  1: Internal IPv4 address
  2: Internal Port: Internal IPv4 address
  3: IPv6 Prefix
  >3: No particular semantic
- o HOST_ID_data depends on the content of the Origin field; padding is required.

Two modes are described below: the SYN mode (Section 3.2.1) and the ACK mode. (Section 3.2.2).

If the ACK mode is used (Section 3.2.2), Figure 4 shows the HOST_ID_ENABLED option (2-bytes long) to be included in the SYN.

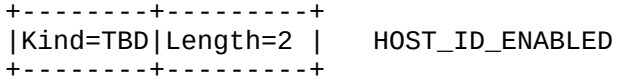


Figure 4: Format of HOST_ID_ENABLED

3.2.1. SYN Mode

This mode is similar to the mode described in Section 3.1. In this mode, HOST_ID_BOUCADAIR is sent in SYN packets.

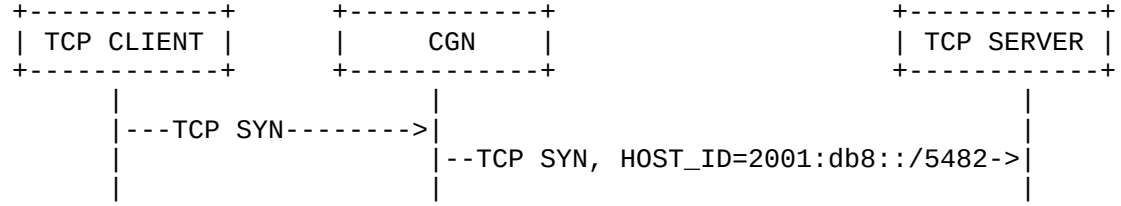


Figure 5: HOST_ID_BOUCADAIR: SYN Mode

3.2.2. ACK Mode

The ACK Mode is as follows (see Figure 6):

- Send HOST_ID_ENABLED (Figure 4) in SYN
- If the remote TCP server supports that option, it must return it in SYNACK
- Then the TCP Client sends an ACK in which the CGN injects HOST_ID_BOUCADAIR (Figure 3)

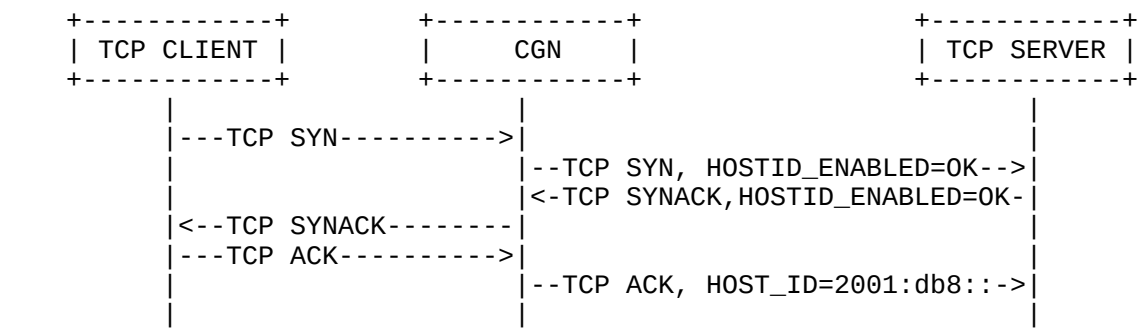


Figure 6: HOST_ID_BOUCADAIR: ACK Mode

4. Overview of the Linux Kernel Modifications

The objective of this phase is to support HOST_ID_WING, HOST_ID_BOUCADAIR and HOST_ID_ENABLED in the SYN mode.

In order to support the injection of the HOST_ID TCP options presented in Section 3, some modifications were applied to the Linux Kernel (more precisely to the TCP stack part of the Kernel). The header file tcp.h, file where are defined the TCP variables and functions, is updated to define the new HOST_ID options' KINDs (option numbers) and Lengths.

Major modifications have been made in the "tcp_output.c" file. This file is responsible for building and transmitting all TCP packets. For each HOST_ID TCP option, the required modifications to increase the header size and to inject KIND, Length and the corresponding HOST_ID data are implemented for the TCP SYN packets.

As we have three different HOST_ID options and as HOST_ID_BOUCADAIR can convey different information the configuration of the HOST_ID options have to be simple with minimal complexity. Since the manipulation of HOST_ID options impacts the Kernel TCP drivers, a suitable solution is to define new sysctl variables (system control variables) that allow the modification of Kernel parameters at

runtime, without having to reboot the machine so that it takes into account a new configuration.

Once modifications have taken place, the Kernel must be recompiled so that the new TCP options are taken into account.

Kernel modifications and recompilation have been done and tested successfully on Fedora and Debian Linux distributions, on different kernel versions.

The following configuration options are supported:

- Enable/Disable injecting the TCP Option
- Support HOST_ID WING, HOST_ID BOUCADAIR and HOST_ID_ENABLED
- When the HOST_ID TCP option is supported, the information to be injected is configurable:
  - Source IPv6 address or the first 56 bits of the address
  - Source IPv4 address
  - Source port number
  - Source IPv4 address and Source port
  - IPv6 address or the first 56 bits of the B4 when DS-Lite is activated

5. Testbed Setup & Configuration

The setup of three testbed configurations have been considered:

1. HOST_ID TCP option is injected by the host itself. No CGN is present in the forwarding path (Figure 7)
2. HOST_ID TCP option is injected by hosts deployed behind a HTTP proxy. No CGN is present in the forwarding path (Figure 8)
3. HOST_ID TCP option is injected by the DS-Lite AFTR element (Figure 9).

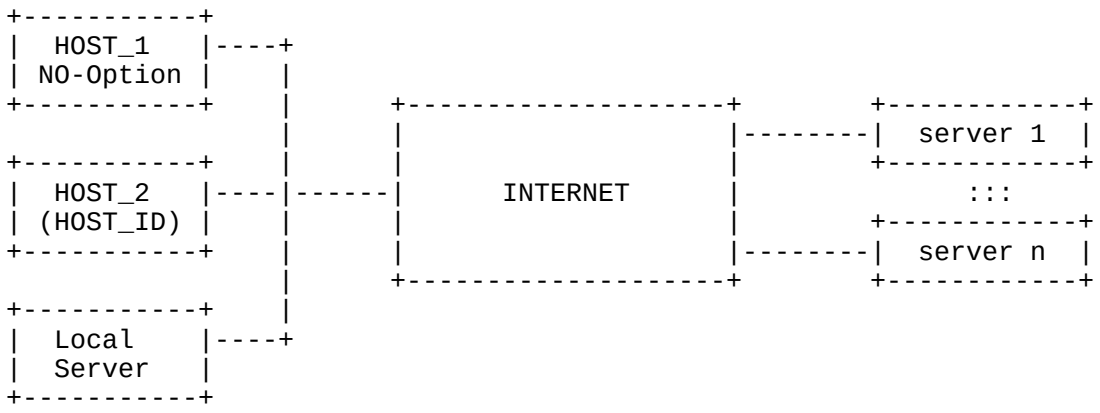


Figure 7: Testbed setup: No Proxy and no CGN

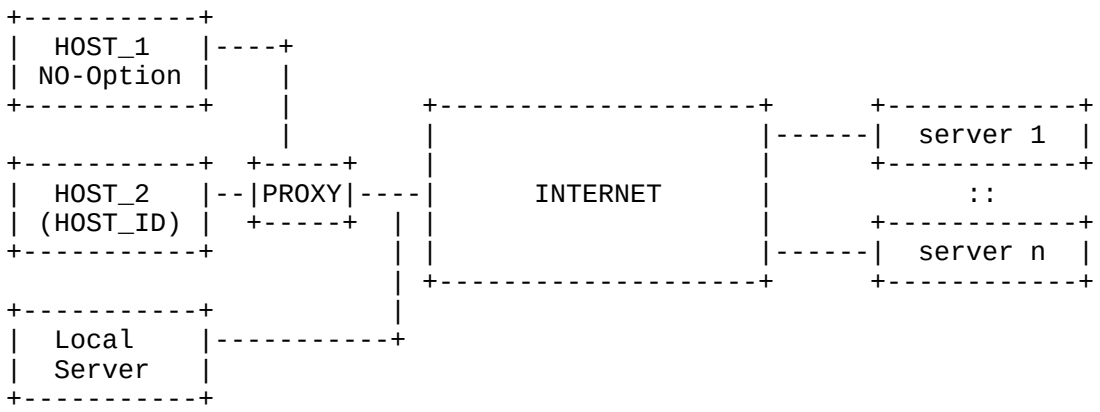


Figure 8: Testbed setup: HTTP Proxy

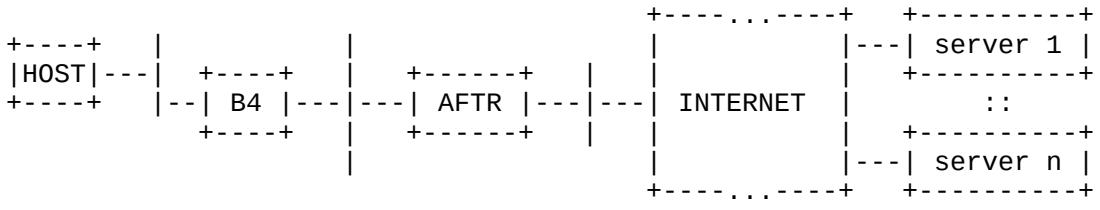


Figure 9: DS-Lite CGN Environment

Figure 7 and Figure 8 are used to assess the behavior of the top 100,000 sites when a HOST_ID option is enabled and to evaluate the impact of the option on both the session establishment delay and the success ratio.

On the other hand, the configuration shown in Figure 9 will be used to evaluate the impact on the CGN performances when HOST_ID TCP option is injected by the CGN.

5.1. Automated TCP Traffic Generator

A Python-encoded robot has been used as the traffic generator. The robot automates the retrieval of HTTP pages identified by URLs, and returns different connection information. The retrieval of pages is based upon Pycurl, a Python interface of libcurl. Libcurl is an URL transfer library that supports different protocols (e.g., HTTP, FTP).

The robot consists of two programs:

1. The first one takes an URL as a input parameter, performs the DNS lookup and then tries to connect to the corresponding machine. It returns either different time values and connection status or an error message with the source of the error in case of connection failure (e.g., DNS error). The TCP connection establishment time is calculated as the difference between the CONNECT_TIME and NAMELOOKUP_TIME where:
   * NAMELOOKUP_TIME is the time it took from the start until the name resolution is completed.
   * CONNECT_TIME is the time it took from the start until the connection to the remote host (or proxy) is completed.
2. The second program aims to increase efficiency and speed of the testing by using a multi-thread technique. It takes the number of threads and an input file listing URLs as parameters. This program prints URLs to an output file with the corresponding connection time. If something wrong happened so that the connection failed, the program returns an error message with the corresponding error type.

5.2. Testing Methodology and Procedure

The testing is done using two machines, one that supports the HOST_ID TCP options and the other that does not. The second machine is used as a reference for the measurements. Testing is performed in parallel on the two machines that are directly connected to the Internet. For each HOST_ID TCP option, the test is repeated many times. The cycle is repeated in different days. Then results are grouped into tables where averages are calculated. The comparison between the different HOST_ID options results is made by using the no-option testing results as a reference.

Testing was also performed behind a proxy (Figure 8) to evaluate the impact of embedding the HOST_ID TCP options on the connection establishment time when a proxy is in the path. When a proxy is

present, the connection delay is impacted (the delay is calculated for the connection between the host and the proxy).

Tests have been conducted from hosts:
1. Connected to an enterprise network
2. In a lab behind a firewall
3. Connected to two (2) commercial ISP networks

5.3. Check HOST_ID TCP Options are Correctly Injected

To check whether the HOST_ID TCP options are correctly injected, the local server in Figure 7 is configured to be reachable from Internet. Packets conveying the HOST_ID TCP options are sent from a host supporting the options. These packets are used without alteration by the local server.

This configuration confirms the packets sent to remote servers conveys HOST_ID TCP options.

5.4. Top Site List

The Alexa top sites list has been used to conduct the HTTP tests.

Anonymous FTP sites list from ftp-sites.org has been used to conduct the FTP tests.

6. Experimentation Results

Various combinations of the HOST_ID TCP options have been tested:
1. HOST_ID_WING
2. HOST_ID_WING has also been adapted to include 32 bits and 64 bits values. No particular impact on session establishment has been observed.
3. HOST_ID_BOUCADAIR (source port)
4. HOST_ID_BOUCADAIR (IPv4 address)
5. HOST_ID_BOUCADAIR (source port:IPv4 address)
6. HOST_ID_BOUCADAIR (IPv6 Prefix)
7. HOST_ID_ENABLED

Both the success ratio and the average time to establish the TCP session are reported below.

6.1. HTTP Experimentation Results

Tests have been conducted from hosts:

1. Connected to an enterprise network
2. Connected to two commercial ISP networks
3. In a lab behind a firewall

6.1.1. Configuration 1: Connected to an enterprise network

The results show that the success ratio for establishing TCP connection with legacy servers is almost the same for all the HOST_ID options as shown in Figure 10, Figure 11 and Figure 12.

6.1.1.1. Results

	NO-OPTION	O-WING	Failure Ratio
Top10	100,00000%	100,00000%	0,00000%
Top100	100,00000%	100,00000%	0,00000%
Top200	100,00000%	100,00000%	0,00000%
Top300	99,66667%	99,66667%	0,00000%
Top400	99,50000%	99,50000%	0,00000%
Top500	99,40000%	99,40000%	0,00000%
Top600	99,50000%	99,50000%	0,00000%
Top700	99,57143%	99,57143%	0,00000%
Top800	99,50000%	99,50000%	0,00000%
Top900	99,44444%	99,44444%	0,00000%
Top1000	99,50000%	99,50000%	0,00000%
Top2000	99,35000%	99,30000%	0,05000%
Top3000	99,10000%	99,06667%	0,03333%
Top4000	99,10000%	99,05000%	0,05000%
Top5000	99,14000%	99,10000%	0,04000%
Top6000	99,21667%	99,18333%	0,03333%
Top7000	99,25714%	99,21429%	0,04286%
Top8000	99,15000%	99,10000%	0,05000%
Top9000	99,16667%	99,12222%	0,04444%
Top10000	99,16000%	99,12000%	0,04000%
Top20000	98,50500%	98,44000%	0,06500%
Top30000	98,21667%	98,11667%	0,10000%
Top40000	98,10750%	98,00750%	0,10000%
Top50000	98,00000%	97,89800%	0,10200%
Top60000	97,95167%	97,85000%	0,10167%
Top70000	97,88857%	97,78857%	0,10000%
Top80000	97,84500%	97,74875%	0,09625%
Top90000	97,79444%	97,69889%	0,09556%
Top100000	97,75100%	97,64800%	0,10300%

Figure 10: Cumulated Success ratio (HOST_ID_WING)

	NO-OPTION	O-WING	Failure Ratio
1-100	100,00%	100,00%	0,00%
101-200	100,00%	100,00%	0,00%
201-300	99,00%	99,00%	0,00%
301-400	99,00%	99,00%	0,00%
401-500	99,00%	99,00%	0,00%
501-600	100,00%	100,00%	0,00%
601-700	100,00%	100,00%	0,00%
701-800	99,00%	99,00%	0,00%
801-900	99,00%	99,00%	0,00%
901-1000	100,00%	100,00%	0,00%
1-1000	99,50%	99,50%	0,00%
1001-2000	99,20%	99,10%	0,10%
2001-3000	98,60%	98,60%	0,00%
3001-4000	99,10%	99,00%	0,10%
4001-5000	99,30%	99,30%	0,00%
5001-6000	99,60%	99,60%	0,00%
6001-7000	99,50%	99,40%	0,10%
7001-8000	98,40%	98,30%	0,10%
8001-9000	99,30%	99,30%	0,00%
9001-10000	99,10%	99,10%	0,00%
10001-20000	97,85%	97,76%	0,90%
20001-30000	97,64%	97,47%	1,70%
30001-40000	97,78%	97,68%	1,00%
40001-50000	97,57%	97,46%	1,10%
50001-60000	97,71%	97,61%	1,00%
60001-70000	97,61%	97,52%	0,90%
70001-80000	97,44%	97,37%	0,70%
80001-90000	97,39%	97,30%	0,90%
90001-100000	97,36%	97,19%	1,70%

Figure 11: TopX000 Success Ratio (HOST_ID_WING)

	NO-OPTION	O-BOUCADAIR	Failure Ratio
1-100	100,00%	100,00%	0,00%
101-200	100,00%	100,00%	0,00%
201-300	99,00%	99,00%	0,00%
301-400	99,00%	99,00%	0,00%
401-500	99,00%	99,00%	0,00%
501-600	100,00%	100,00%	0,00%
601-700	100,00%	100,00%	0,00%
701-800	99,00%	99,00%	0,00%
801-900	99,00%	99,00%	0,00%
901-1000	100,00%	100,00%	0,00%
0-1000	99,50%	99,50%	0,00%
1001-2000	99,20%	99,10%	0,10%
2001-3000	98,60%	98,60%	0,00%
3001-4000	99,30%	99,30%	0,00%
5001-6000	99,60%	99,60%	0,00%
6001-7000	99,50%	99,40%	0,10%
7001-8000	98,40%	98,30%	0,10%
8001-9000	99,30%	99,20%	0,10%
9001-10000	99,10%	99,10%	0,00%
10001-20000	97,85%	97,76%	0,90%
20001-30000	97,64%	97,46%	1,80%
30001-40000	97,78%	97,66%	1,20%
40001-50000	97,57%	97,46%	1,10%
50001-60000	97,71%	97,61%	1,00%
60001-70000	97,61%	97,51%	1,00%
70001-80000	97,44%	97,36%	0,80%
80001-90000	97,39%	97,30%	0,90%
90001-100000	97,36%	97,19%	1,70%

Figure 12: TopX000 Success Ratio (HOST_ID_BOUCADAIR)

6.1.1.2. Analysis

- For the top 100,000 sites, connection failures occur for 2249 HTTP sites. These failures were reported as being caused by DNS issues (servers not mounted), connection timeouts (servers down...), connection resets by peers, connection problems and empty replies from servers. The 2249 failures occur, whether HOST_ID options are injected or not.
- When any HOST_ID TCP option is conveyed, 103 servers did not respond; however when no option is injected, all these servers responded normally.

- Same results were obtained for HOST_ID_WING and HOST_ID_ENABLED.
- Same results were obtained for all the HOST_ID_BOUCADAIR options (source port, IPv6 prefix, etc.).

When HOST_ID_BOUCADAIR is enabled, six (6) additional servers did not respond:

- Three (3) servers (www.teufel.de - www.1001fonts.com - www.sigur-ros.co.uk) did not respond to the SYN packets sent by the host.
- Three (3) servers (www.lawyers.com, www.lexis.com, www.nexis.com) responded with "strange" SYN/ACK packets with same TCP options length including a part of the HOST_ID options that was sent. This part of HOST_ID option caused an erroneous SYN/ACK packet received by the host: in fact the second byte of the HOST_ID part is considered as its length and this length does not really fit with the real length of the part. So the machine does not respond back to the server with an ACK packet. This is why we have no response for these servers.

When HOST_ID_WING or HOST_ID_ENABLED is enabled, also strange SYN/ACKs were received by the host but no errors in these packets (a long series of NOP options). This justifies the connection success for these 2 options.

The results show that including a HOST_ID TCP option does not systematically imply an extra delay for the establishment of the TCP session. Based on the average of session establishment with the top 100 000 sites, the following results have been obtained:

- delay(HOST_ID_WING) < delay(NO_OPTION): 42,55 %
- delay(HOST_ID_BOUCADAIR) < delay(NO_OPTION): 48,16 %
- delay(HOST_ID_ENABLED) < delay(NO_OPTION): 51,28 %

6.1.2. Configuration 2: In a lab behind a firewall

When a HTTP proxy is in the path, the injection of HOST_ID TCP option does not impact the success ratio. This is due to that the HTTP proxy strips the HOST_ID TCP options; these options are not leaked to remote Internet servers. The testing has been done by observing packets received to a server installed with a public IP address (no HOST_ID options were seen in the received SYN packets).

6.1.3. Configuration 3: Connected to two commercial ISP networks

The results obtained when testing was performed by connecting to two ISP networks confirmed the results obtained in the testing described in Section 6.1.1

6.1.4. Additional Results

In one of our testing for top 1000 sites, when padding was badly implemented for HOST_ID_BOUCADAIR (padding was implemented as a prefix so option's Length does not correspond to the real length because the padding was not counted), we got for configuration(1) in the lab and for one of the ISP the following results:

	No-Option	O-BOUCADAIR	Failure Ratio
Top10	100,00000%	100,00000%	0,00000%
Top100	100,00000%	100,00000%	0,00000%
Top200	100,00000%	100,00000%	0,00000%
Top300	100,00000%	99,66667%	0,33333%
Top400	99,75000%	99,00000%	0,75000%
Top500	99,80000%	99,00000%	0,80000%
Top600	99,83333%	98,66667%	1,16667%
Top700	99,85714%	98,14286%	1,71429%
Top800	99,75000%	98,00000%	1,75000%
Top900	99.66667%	97,33333%	2,33333%
Top1000	99,70000%	97,10000%	2,60000%

Cumulated Success ratio (HOST_ID_Boucadair with wrong padding)

The results for HOST_ID_WING for all three configurations are the same as Section 6 (this option was correctly coded). Results obtained for HOST_ID_BOUCADAIR are not the same.

For the configuration (2) behind a firewall, we did not face any rejection because of parsing the TCP options (the HOST_ID options were retrieved from the packet).

6.1.5. Analysis

Configuration (1) in Lab and for one of the two CPEs lead to the results because 2.6% of these 1000 servers perform parsing validation for the received options so when the bad HOST_ID_BOUCADAIR option is sent, 2.6% of the servers treat the received SYN packets as erroneous packets and discard them.

For the connection behind the second ISP, we didn't get a response for any of the servers. After investigation, the reason was that the Box validates the received packets before sending them to the Internet. The erroneous SYN packets holding badly encoded options (HOST_ID_BOUCADAIR in this case) were dropped and no connection was

established. On the other hand, the other box did not validate options length for received packets before sending them to the Internet.

6.2. FTP

Various combinations of the HOST_ID TCP options have been tested:

1. HOST_ID_WING
2. HOST_ID_BOUCADAIR (source port)
3. HOST_ID_BOUCADAIR (source port:IPv4 address)

A list of 5591 FTP servers has been used to conduct these testings. Among this list, only 2045 were reachable:

- Failure to reach 942 FTP servers due to connection timeout
- Failure to reach 1286 FTP servers due to DNS errors
- Failure to reach 717 FTP servers because access was denied
- Could not connect to 500 FTP servers
- Response reading failed for 81 servers
- Bad response from server for 20 servers

When HOST_ID TCP options are injected, 9 errors are observed (connection timeout).

Figure 13 and Figure 14 provide more data about the error distribution.

	NOB	HOST_ID	Failure Ratio
1-100	100%	100%	0,000%
101-200	100%	99%	1,000%
201-300	100%	99%	1,000%
301-400	100%	100%	0,000%
401-500	100%	100%	0,000%
501-600	100%	100%	0,000%
601-700	100%	100%	0,000%
701-800	100%	100%	0,000%
801-900	100%	99%	1,000%
901-1000	100%	99%	1,000%
1001-2000	100%	99,5%	0,500%
2000-2045	100%	100%	0,000%

Figure 13: Cumulated Success Ratio (FTP)

	NOB	HOST_ID	Failure Ratio
first 10	100,000%	100,000%	0,000%
first 100	100,000%	100,000%	0,000%
first 200	100,000%	99,500%	0,500%
first 300	100,000%	99,333%	0,667%
first 400	100,000%	99,500%	0,500%
first 500	100,000%	99,600%	0,400%
first 600	100,000%	99,667%	0,333%
first 700	100,000%	99,714%	0,286%
first 800	100,000%	99,750%	0,250%
first 900	100,000%	99,667%	0,333%
first 1000	100,000%	99,600%	0,400%
first 2000	100,000%	99,550%	0,450%
first 2045	100,000%	99,560%	0,440%

Figure 14: FirstXXX FTP Servers

The results show that including a HOST_ID TCP option does not systematically imply an extra delay for the establishment of the TCP session with remote FTP servers. Based upon the average of the session establishment with the 2045 FTP sites, the following results have been obtained:

- delay(HOST_ID_WING) < delay(NO_OPTION): 49,36585 %
- delay(HOST_ID_BOUCADAIR (source port:IPv4 address)) < delay(NO_OPTION): 48,41076%
- delay(HOST_ID_BOUCADAIR (source port)) < delay(NO_OPTION): 48,43902 %

6.3. SSH

The secure shell service has been tested between a host and a SSH server connected to the same network.

SSH connections have been successfully established with the server for all the HOST_ID TCP options. Same results were obtained using configuration (1) and configuration (2).

6.4. Telnet

Telnet sessions have been successfully initiated for all HOST_ID TCP options with a server (the CGN used in Figure 9).

7. AFTR Module Modifications

This section highlights the support the HOST_ID functionalities in the AFTR element of the DS-Lite model (Figure 9) and presents the testing results in order to conclude about the HOST_ID TCP options impacts on the performance of the CGN.

We used ISC AFTR implementation.

7.1. Specification

All privately-addressed IPv4 packets sent from DS-Lite serviced hosts go through an AFTR device where an isc_aftr daemon program is responsible for establishing the tunnel, configuring network interfaces and processing received packets.

The aftr.c source code controls all functionalities to be included or modified on packets received by the CGN, e.g., patching TCP MSS values, fix MTU, etc.

In order to activate/deactivate such functionalities, the corresponding parameters can be configured in a specific configuration file called "aftr.conf". In this file, other parameters are configured, e.g., the IPv6 addresses assigned to the tunnel endpoint and the global IPv4 address pool maintained by the CGN.

To support the injection of HOST_ID TCP options, "aftr.c" must be updated to inject, retrieve or verify the HOST_ID options depending on the HOST_ID parameters defined in "aftr.conf" file. Four HOST_ID parameters are defined in the configuration file:

1. hostid: to enable the injection, retrieval, matching... of HOST_ID options
2. hostid_wing: to enable injection/verification of HOST_ID_WING - to disable injection or to remove HOST_ID_WING
3. hostid_boucadair: to enable injection/verification of HOST_ID_BOUCADAIR - to disable injection or to remove HOST_ID_BOUCADAIR
4. hostid_enabled: to enable or disable HOST_ID_ENABLED injection

hostid, hostid_wing and hostid_enabled can be simply enabled or disabled. hostid_boucadair can be disabled or enabled with the corresponding Origin as HOST_ID data can be:

- Source Port Number
- Source IPv4 Address
- Source IPv4 Address + Source Port Number

- o 56 bits of Tunnel Softwire IPv6 Source Address.

Based on different HOST_ID parameters, the "aftr.c" code has been modified to control HOST_ID options; the AFTR is able to:

- o Inject the enabled HOST_ID TCP option if it is not already present in the packet
- o Retrieve an existing HOST_ID TCP option if this option is not enabled
- o Check an existing HOST_ID option's content if it is enabled; if the content's verification failed, the AFTR replaces the HOST_ID contents with the suitable information

The implementation takes into consideration the SYN mode for all the HOST_ID options (even for HOST_ID_enabled). The Support of HOST_ID_BOUCADAIR in the ACK mode needs implementation on the server's side and since both Enabled and Boucadair's options have been tested and no impact observed; the ACK mode should not imply any complication in implementation or impact on the performance.

7.2. Verification

The verification of HOST_ID implementation in the CGN has taken place using the testbed setup shown in Figure 9. The host used in this testing is a modified Linux machine that can inject HOST_ID options. The objective of the testing is to verify the different functionalities implemented in the AFTR. Verification has occurred using a local server where all the received packets were observed to make sure that the content of the HOST_ID fields is consistent with the enabled option.

The testing consists in observing the SYN packets (as SYN mode is supported) sent by the host and in comparing these packets to those received by the server. Different combinations of HOST_ID options sent by the host and HOST_ID configured options at the CGN level have been used.

The results show that once the host sends packets without any HOST_ID option injected, the SYN packets received by the server contain the correct option that has been enabled by the CGN (if any). Once HOST_ID_WING or HOST_ID_BOUCADAIR are injected by the host, if the hostid parameter in aftr.conf is enabled, the enabled (in "aftr.conf") HOST_ID option will be injected if not already present, or else its content will be verified and corrected (if wrong); the other disabled option will be discarded if it has already been sent by the host.

One additional case has been tested when both Wing's and Boucadair's HOST_ID options are sent by the host, the contents of the enabled

option are checked and corrected (if wrong), the other option is retrieved from the packet. The two options are dropped from the packet if they are both disabled.

The testing has been repeated for all the HOST_ID options sent by the host and enabled by the CGN. Verification also occurred for HOST_ID_ENABLED option.

7.3. CGN Performance Testing

To conclude about the impact of using HOST_ID, a commercial testing product has been used. This tool supports multiple application protocols such as HTTP and FTP for both IPv4 and IPv6 (including encapsulation). The DS-Lite model can be built directly from a port of this product: IPv4 packets are directly encapsulated in an IPv6 tunnel; the client's port emulates hosts and B4 elements at the same time. This port is directly connected to the AFTR tunnel endpoint. The AFTR's IPv4 interface is connected to the testing product server side where servers are assigned IPv4 addresses.

The testbed setup of this testing is shown in Figure 15:

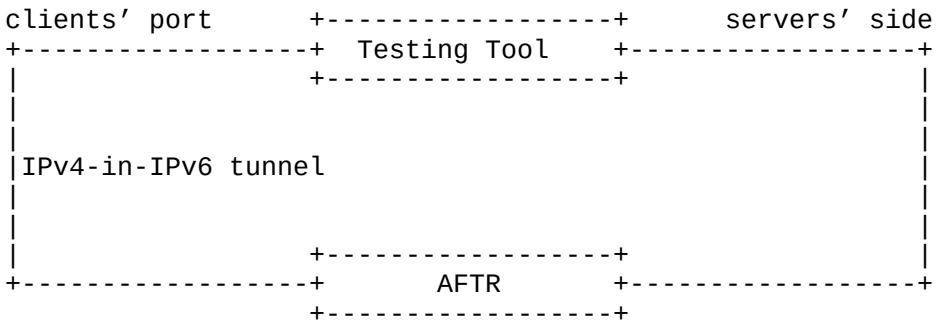


Figure 15: Platform Testbed

7.3.1. Configuration

At the IP level, the testing client port was configured with IPv6 addresses representing the B4. The testing tool also supports the DS-Lite "level" where the number of clients connected to each B4 and their addresses are configured. The AFTR address is defined at this level.

In the current testing, the total number of B4 elements is 5000 behind; One client is connected to each B4 (in total, 5000 clients are configured). However, the number of active users varies from 10 to 100, 500, 1000 and 10,000 during each testing simulation.

From the server standpoint, five servers have been assigned IPv4 addresses. These servers support HTTP and FTP traffic. For each HOST_ID TCP option, the testing was repeated for a different number of active users (N=10, 100, 500, 1000 and 10,000) and for HTTP and FTP traffic.

The HOST_ID options are injected by the CGN.

7.3.2. HTTP Testing

The testing duration was about 50 seconds during which the number of active users varies as a function of time: during the first 10s, the number of active users reaches the maximum and remains the same for the next 20 s. Then it decreases to zero during the next 20s.

Hereafter are provided some testing statistics providing some details about connections' success ratio, latency and other information that can be useful to evaluate the impact of HOST_ID on the CGN.

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1378	1267	1363	1369
TCP SYN SENT	1378	1267	1363	1369
Success Ratio	100	100	100	100
TCP Retries	193	193	197	177
TCP timeouts	140	136	152	111
HTTP connect' latencies t=20s	0,11	0,21	0,20	0,1
t=40s	0,40	0,50	0,50	0,45
t=60s	0,60	0,60	0,50	0,6
HTTP throughput received	46,47	45,31	45,88	46,12
TCP Connections Established/s	20,29	19,88	20,06	20,18

Figure 16: Results HTTP (N=10)

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1662	1739	1813	1679
TCP SYN SENT	1718	1770	1819	1729
Success Ratio	96	98	99	97
TCP Retries	1577	1569	1783	1576
TCP timeouts	798	806	934	808
HTTP connect' latencies t=20s	1,70	2,00	1,90	1,80
t=30s	3,30	2,40	2,25	3,30
t=40s	4,20	3,70	3,75	4,00
t=50s	5,00	4,80	4,50	5,00
HTTP throughput received	47,56	46,65	48,59	48,06
TCP Connections Established/s	20,94	20,53	21,35	21,19

Figure 17: Results HTTP (N=100)

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1956	1923	1944	1873
TCP SYN SENT	2088	2095	2137	1986
Success Ratio	93	91	90	94
TCP Retries	2734	2576	2453	2773
TCP timeouts	1261	1110	995	1213
HTTP connect' latencies t=20s	2,00	1,80	1,50	2,30
t=40s	4,00	3,30	2,80	4,30
t=50s	6,50	6,90	6,00	8,00
HTTP throughput received	70,19	65,00	69,81	67,13
TCP Connections Established/s	30,69	28,41	30,50	29,38

Figure 18: Results HTTP (N=1000)

	No-Opt	O-WING	O-BOUCADAIR4	O-ENABLED
TCP connection established	1576	2000	1796	1998
TCP SYN SENT	2088	2304	2009	2262
Success Ratio	87	86	89	88
TCP Retries	3018	3101	3013	3148
TCP timeouts	1167	1298	1213	1417
HTTP connect' latencies t=20s	2,20	3,00	2,20	2,50
t=40s	3,70	3,00	3,30	3,00
t=60s	7,80	5,00	7,00	5,60
t=70s	9,60	6,00	8,70	7,00
HTTP throughput received	45,00	54,52	51,45	57,20
TCP Connections Established/s	19,98	24,05	22,45	25,04

Figure 19: Results HTTP (N=10000)

7.3.2.1. Analysis of results

The results clearly show that there is no impact of any HOST_ID option on session establishment success ratio, which is quite similar to the success ratio when packets do not hold options or when HOST_ID options are not used. Also, the number of established connections does not decrease when any HOST_ID option is injected, so the CGN performance is not impacted by the fact of adding the HOST_ID options.

Another important factor to study is the latency that can be caused by HOST_ID injection. As the results show, the HTTP connection latency does not increase when HOST_ID is present if we compare the latency measured at different times for the different options.

As a result, we clearly see that the average throughput measured at servers is identical, whether HOST_ID options are used or not (given that the number of session established is quite the same).

Another consequence is that the TCP connection establishment rate at servers is not decreasing when a HOST_ID option is taken into account.

7.3.2.2. Conclusion

The results that have been obtained show that the performance of the CGN is not impacted by HOST_ID option injection even when the number of active users is high (10,000 is not negligible for a CGN run on an ordinary Linux machine): neither the session success ratio, nor the connection latency are impacted by the presence of the HOST_ID in SYN

packets.

7.3.3. FTP

The same testing was also run for FTP traffic. No particular impact on the performance of the CGN has been observed.

8. IPTABLES: Modifications to Enforce Policies at the Server Side

8.1. Overview

iptables module has been updated to:
- Log the content of TCP header with HOST_ID
- Drop packets holding a HOST_ID option
- Match any HOST_ID value
- Drop packets holding a specific HOST_ID value
- Strip any existing HOST_ID option

To support the above functionalities, modification should take into consideration stripping and matching options as described below:

1. To strip the content of any existing HOST_ID option, the shared library "libxt_TCPOPTSTRIP.so" is modified: the HOST_ID_WING and HOST_ID_BOUCADAIR Kinds' numbers were defined in the corresponding source file (libxt_TCPOPTSTRIP.c) with the corresponding names to enforce the iptables stripping rule. After enforcing these changes, the shared library must be created to replace the existing one and to allow applying the rule of stripping of the HOST_ID options. Once modifications have taken place, the following command should be used to strip the HOST_ID options:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing, hostid_boucadair
```

2. In order to allow blocking, logging or applying any rule based upon the HOST_ID_WING or HOST_ID_BOUCADAIR values or range of values, a HOST_ID shared library must be created to:
   * Match HOST_ID options values entered in corresponding iptables rules,
   * Print the HOST_ID rules on screen,
   * Save values,
   * Check the values (or range values) entered by user if they respect the limit values of these options.

In addition to the shared library: a specific Kernel module must be built to apply HOST_ID matching rules on the packets passing through the network interfaces. This module compares the HOST_ID options' values held by packets with the HOST_ID values specified in the iptables rule table: when a packet matches the HOST_ID's range, the corresponding rule will be applied for this packet. The HOST_ID_WING matching value is 2 bytes long corresponding to HOST_ID_WING data.
The HOST_ID_BOUCADAIR matching value is 8 bytes long corresponding to Lifetime + Origin field (1 byte) and HOST_ID_WING data (7 bytes).

8.2. Validation

After having updated the iptables package with the suitable HOST_ID libraries and module, different HOST_ID policies should be applied and tested on the server side. The testing has been done using a simple configuration as shown below (Figure 20).

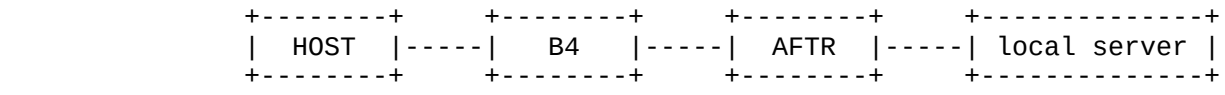


Figure 20: Platform configuration: HOST_ID enforcing policies

In the current testing, the AFTR supports HOST_ID options injection and iptables is modified at the local server. Logging recommendations consists of logging the IPv4 address and the HOST_ID option for each connection. Because HOST_ID is sent only in SYN packets (in the current implementation), only SYN packets will be logged to a specific file called iptables.log: the rsyslog.d must be updated with the corresponding command to log iptables messages into the specific file. Then rsyslog must be reloaded to apply changes.

8.3. Stripping HOST_ID Options

To strip a certain HOST_ID option, TCPOPTSTRIP rule must be called. Verification consists in logging and then checking the SYN packets and more precisely the corresponding TCP options, e.g., the following rules must be applied to strip HOST_ID_WING:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing
iptables -A INPUT -j LOG --log-tcp-options -p tcp --syn
```

The first rule applies for the mangle table. This table allows

stripping HOST_ID_WING whose role is to remove option Wing's fields and replaces them by NOP options (NOP=No Operation=0x01). The second rule enables the logging of SYN packets with the corresponding TCP options.

After applying these rules (to strip and log HOST_ID_WING) on the local server, we tried to access the server's HTTP pages from the host. The test is repeated several times and a different HOST_ID option is enabled by the AFTR each time.

Then the "iptables.log" file is checked: only one SYN packet is logged with 4 bytes stripped out in the TCP option part. All IPv4 packets going through the AFTR are also logged to be compared with the server's logged stripped packets.

The comparison of the SYN packets logged by the server with the SYN packets sent by the AFTR clearly shows that the stripped option is HOST_ID_WING (all the header fields have been verified to ensure packet matching): the 4 bytes corresponding to the HOST_ID_WING option are replaced with NOP options (each one of the 4 bytes is equal to '1' = NOP).

The same testing was repeated with HOST_ID_BOUCADAIR. The testing shows that the 10 bytes corresponding to this option were successfully stripped.

8.4. Logging a Specific HOST_ID Option Value

The remote server should be able to track connections coming from different clients; it should log packets headers including the HOST_ID TCP option information. This can be enforced using the following command:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing
```

Now, to log packets matching a certain HOST_ID value or range of values, the following rule must be applied:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_wing value[:value]
 -j LOG -log-tcp-options
```

This command matches the HOST_ID_WING values held by SYN packets with the specific value [or the specific range of values] determined by

the rule.

The testing configuration in Figure 20 was used. The HOST_ID_WING data are implemented as being the last 16 bits of the IPv4 private source address. When the HOST_ID_WING option is injected by the CGN, if the data field value corresponds to the iptables value (or range of values), the packet header is logged. Otherwise, if the HOST_ID_WING data is said out of range or the packet does not hold the HOST_ID_WING option, the packet is not logged.

The same testing was repeated to match HOST_ID_BOUCADAIR data information:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_boucadair value
 [:value] -j LOG -log-tcp-options
```

To verify the logging of a specific Boucadair's value, the Boucadair's options holding source IP address (Origin=2) or IPv6 prefix (Origin=4) were tested successfully; these data values are fixed since they depend on the host's address. The two other options that include source port numbers (variable) cannot be tested by value because the port number varies for each connection.

The iptables rules to log HOST_ID_BOUCADAIR range values have been verified successfully for all four HOST_ID_BOUCADAIR options.

8.5. Dropping a specific HOST_ID Option Value

The same testing methodology described in the previous section was repeated to drop packets matching HOST_ID value (or a range of values); e.g. to drop SYN packets matching a particular HOST_ID_WING value:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_wing value[:value]
 -j DROP
```

In this testing, the HOST_ID_WING option is enabled at the CGN level. After applying the previous rule where Wing's specified value corresponds to the HOST_ID_WING data value (last 16 bits of the host's IPv4 source address), the hosts tries to access HTTP pages of the local server. It sends SYN packets but the server does not respond. Because this packet matches the iptables matching value, the corresponding rule is applied to the SYN packets: a SYN packet is dropped so the host does not receive any packet in return.

When the host is still trying to retrieve pages by sending SYN packets, the command 'iptables -F' will flush all iptables rules. Once applied, this command will let the host retrieve the required pages and the connection is therefore established successfully.

The same testing was repeated for HOST_ID_BOUCADAIR options. SYN packets matching the corresponding rule value or range of values were dropped. Once iptables rules are flushed, connection is established normally.

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

Security considerations discussed in [I-D.wing-nat-reveal-option] should be taken into account.

11. Acknowledgments

Many thanks to M. Meulle, P. Ng Tung and L. Valeyre for their help and review. Special thanks to C. Jacquenet for his careful review.

12. References

12.1. Normative References

[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-03 (work in progress), December 2011.

[RFC6250] Thaler, D., "Evolution of the IP Model", RFC 6250, May 2011.

12.2. Informative References

[I-D.boucadair-intarea-nat-reveal-analysis]
Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments", draft-boucadair-intarea-nat-reveal-analysis-04 (work in

progress), September 2011.

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

Authors' Addresses

Elie Abdo
France Telecom
Issy-les-Moulineaux

Email: elie.abdo@orange.com

Mohamed Boucadair
France Telecom

Email: mohamed.boucadair@orange.com

Jaqueline Queiroz
France Telecom
Issy-les-Moulineaux

Email: jaqueline.queiroz@orange.com

behave
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2018

C. Bao
X. Li
CERNET Center/Tsinghua University

October 17, 2017

Extended IPv6 Addressing for Encoding Port Range
draft-bcx-behave-address-fmt-extension-11

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on April 20, 2018.

Table of Contents

1. Introduction

This document discusses an extension of the address format defined in [RFC6052]. In Section 2.2, the IPv4-embedded IPv6 address format is defined which composed of a variable length prefix, the embedded IPv4 address, and a variable length suffix, as presented in the following diagram, in which PL designates the prefix length:

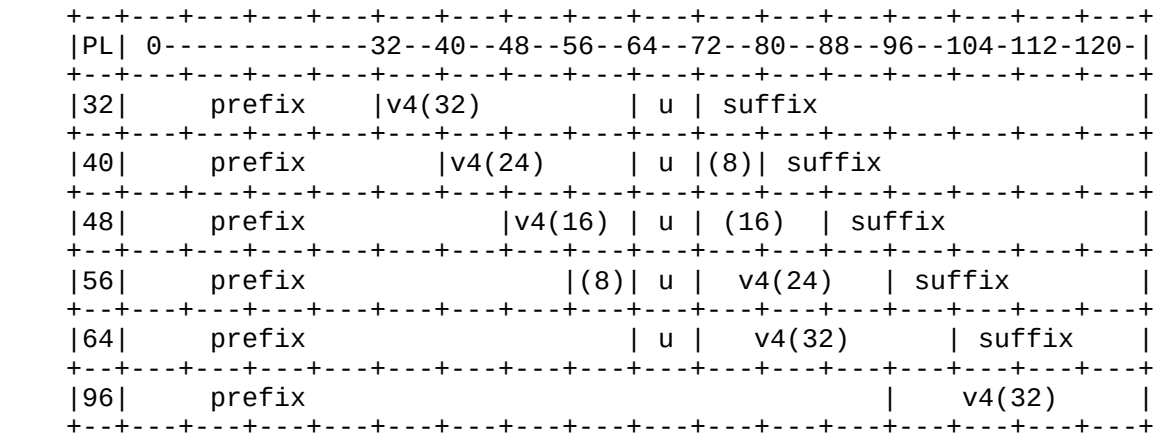


Figure 1: Address Format

In [RFC6052] Section 3.5, it states:

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

This document defines such a suffix encoding scheme and the corresponding port mapping algorithm.

1.1. Applicability Scope

The address format extension presented in this document is used for IPv4/IPv6 stateless translation and dual IPv4/IPv6 stateless translation without prefix delegation [I-D.xli-behave-divi]. The address format used for dual IPv4/IPv6 stateless translation and encapsulation with prefix delegation should refer to [RFC7599] [RFC7597].

1.2. Conventions

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

2. Port Mapping Algorithm

2.1. Mathematical representation of the Algorithm

There exist many port mapping algorithms and each one may have advantages and disadvantages, as well as has its best application scenario. Since different PSID MUST have non-overlapped port range, the two extreme cases are: (1) the port number is not continue for each PSID, but uniformly distributed cross the whole port range (0-65535); (2) the port number is continue in a single range for each PSID. The port mapping algorithm proposed here is called generalized modulus algorithm and it is flexible, meets these two cases and simple.

For given sharing ratio (R) and the maximum number of continue ports (M), the generalized modulus algorithm is defined as

1. The port number (P) of a given PSID (K) is composed of

   P = R*M*j + M*K + i

   Where
   o PSID: K=0 to R-1
   o Port range index: j = (1024/M)/R to ((65536/M)/R)-1, if the well-known port numbers (0-1023) are excluded.
   o Port continue index: i=0 to M-1

2. The PSID (K) of a given port number (P) is determined by

   K = (floor(P/M)) % R

   Where
   o % is modular operator
   o floor(arg) is a function returns the largest integer not greater than arg

3. The well-known port number (0-1023) can be used, if additional port mapping rule is defined.

2.2. Bit Representation of the Algorithm

Given sharing ratio (R=2^k), the maximum number of continue ports (M=2^m), for any PSID (K) available ports (P) can be represented as:

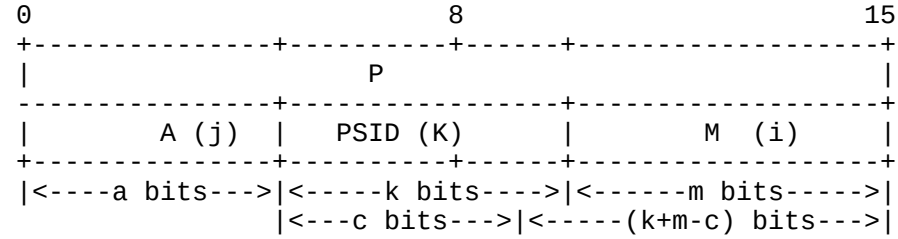


Figure 2: Bit representation

Where j and i are the same indexes defined in the port mapping algorithm.

For any port number, the PSID can be obtained by bit mask operation and therefore the generalized modulus algorithm does not introduce the computational complexity.

Note that in above figure there is a PSID prefix length (c). Based on this definition, PSID is also in CIDR style and more ports can be assigned to a single CE when PSID prefix length (c < k).

When m=0, the generalized modulus algorithm becomes modulus operation. When a=0, the generalized modulus algorithm becomes division operation.

2.3. Example of the Algorithm

2.3.1. PSID with fixed prefix length

For example, for R=128 (k=7), M=4 (m=2)

	Port range-1	Port rang-2	Port
PSID=0	1024, 1025, 1026, 1027,	1536, 1537, 1538, 1539,	2048
PSID=1	1028, 1029, 1030, 1031,	1540, 1541, 1542, 1543,	....
PSID=2	1032, 1033, 1034, 1035,	1544, 1545, 1546, 1547,	....
PSID=3	1036, 1037, 1038, 1039,	1548, 1549, 1550, 1551,	....
...			
PSID=127	1532, 1533, 1534, 1535,	2044, 2045, 2046, 2047,	....

Figure 3: Example 1

2.3.2. PSID with variable prefix length

For example, different PSIDs have different prefix length (c)

Host	PSID prefix	Number of ports
Host0	000/2	2x8192
Host1	010/3	1x8192
Host2	011/3	1x8192
Host3	100/1	4x8192

Figure 4: Example 2

2.4. Features of the Algorithm

The generalized modulus operation has the following features:

1. There is no waste of the port number, except the well-known ports.

2. The algorithm is flexible, the control parameters are sharing ratio (R), the continue port range (M) and PSID prefix length (c).

3. It does not introduce algorithm complexity.

4. It allows service providers to define their own address sharing ratio, the theoretical value is from 1:1 to 1:65536 and a more practical value is from 1:1 to 1:4096.

5. It supports deployments using differentiated port ranges.

6. It supports differentiated port ranges within a single shared IPv4 address.

7. It support excluding the well known ports 0-1023.

8. It supports assigning well known ports to a CE.

9. It supports legacy RTP/RTCP compatibility.

3. Extended IPv4-translatable IPv6 Address

3.1. Address Format

Based on the port mapping algorithm, the extended address format is shown in the following figure.

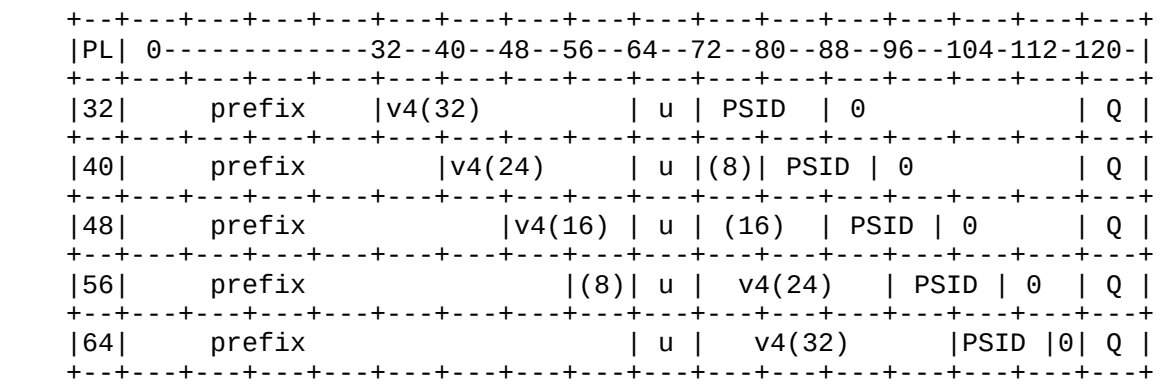


Figure 5: Address Format

Where PL designates the prefix length.

The PSID is placed right after the IPv4 address, since the combination of the IPv4 address and the PSID represents the more specifics in CIDR style which is sharing an IPv4 address with others.

The PSID prefix length (Q=c) is encoded in the last octet (bits 120-127) to indicate the number of ports can be used. When Q=0, the extended address format will become the address format defined in [RFC6052]. The relations between Q, the sharing ratio (R), the maximum continue port range (M) and the number of ports can be shown in the following figure.

Q	Ratio	Maximum M	# of Ports
0	1:1	65,536	65,536
1	1:2	32,786	32,786
2	1:4	16,384	16,384
3	1:8	8,192	8,192
4	1:16	4,096	4,096
5	1:32	2,048	2,048
6	1:64	1,024	1,024
7	1:128	512	512
8	1:256	256	256
9	1:512	128	128
10	1:1,024	64	64
11	1:2,048	32	32
12	1:4,096	16	16

Figure 6: Port range

Since newly defined IPv6 addresses with suffix are more specifics compared with the original address format defined in [RFC6052], the routing considerations in that document are also applied here. Furthermore, the port range is embedded in the extended IPv4-translatable IPv6 addresses and bound to the PSID therefore the packets containing extended IPv4-translatable IPv6 addresses as the destination can be routed to different IPv6 nodes.

3.2. Considerations of Using a Shorter Prefix length

Since IPv4 address plus variable length PSID represents the more specifics, the prefix length (PL) defined in [RFC6052] can be shorter. In these cases, the interface identifier (IID: second 64 bits) will not contains PSID and therefore can be used for regular prefix delegation, as shown in the following figure.

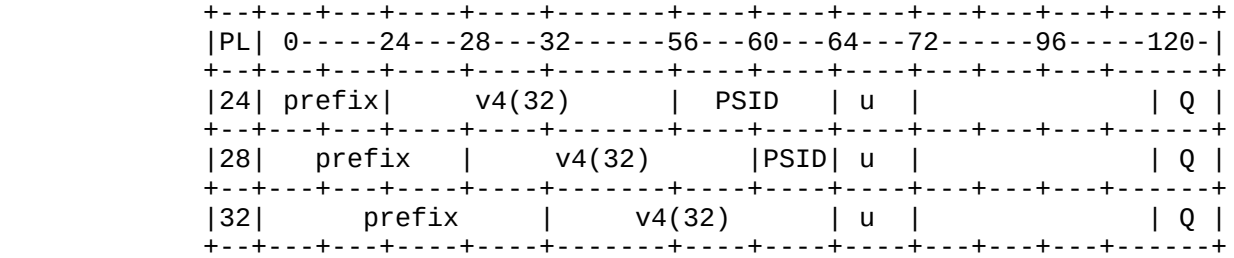


Figure 7: Shorter PL

Note that PL can take any value. For example,

- o PL=24: Q=8, R=256
- o PL=25: Q=7, R=128
- o PL=26: Q=6, R=64
- o PL=27: Q=5, R=32
- o PL=28: Q=4, R=16
- o PL=29: Q=3, R=8
- o PL=30: Q=2, R=4
- o PL=31: Q=1, R=2
- o PL=32: Q=0, R=1

However, there will be a waste of the IPv6 address space in order to represent the IPv4-converted addresses.

3.3. Mapping Extended IPv4-translatable IPv6 Address to RFC1918 Space

Based on the algorithm defined in this document, a public IPv4 address and PSID can be mapped to extended IPv4-translatable IPv6 address and vise versa.

On the other hand, it is also possible to map the extended IPv4-translatable IPv6 address to [RFC1918] address space. In this case, one public IPv4 address can be mapped to several RFC1918 addresses and used by IPv4 or dual stack hosts.

For public IPv4 address a.b.c.d,

- o If R <= 256, the corresponding RFC1918 address is 10.c.d.PSID (PSID has 8 bits)
- o Otherwise, the corresponding RFC1918 address is 10.d.[PSID] (PSID has 16 bits)

4. DHCP Options Extensions

Based on the address format and the port mapping algorithm defined in this document, the IPv6 host needs to get the corresponding parameters via DHCPv6 [RFC3315][RFC3633] or others signaling scheme. These parameters are:

1. The IPv6 prefix
2. The IPv6 prefix length
3. The IPv4 prefix
4. The IPv4 prefix length

5. The sharing ratio (R)

6. The maximum number of continue ports (M)

7. The PSID (K)

8. The PSID length (c)

5. Comparisons with MAP

There are common parts and differences between this document and the address format defined in [RFC7597] [RFC7599].

1. The address format extension defined in this document is used for single and dual stateless translation without prefix delegation, while MAP is used for encapsulation and dual stateless translation with prefix delegation.

2. The address format extension defined in this document uses same IPv6 prefix for the source address from a CE to any destination (IPv4-translatable address) and the destination address from a CE to the outside IPv4 Internet (IPv4-converted address), while MAP uses different IPv6 prefixes, due to the requirements of prefix delegation.

3. The address format extension defined in this document uses same IPv6 prefix for all CEs, so there is no need to define prefix encoding scheme (e.g. CE index, or EA-bits), while MAP defines the prefix encoding scheme, due to the requirements of prefix delegation.

4. Due to the nature of using same IPv6 prefix for both IPv4-translatable address and IPv4-converted address, there is no referral problem and mesh scenarios can be supported without additional mapping rules, while MAP does require additional mapping rule for supporting mesh scenario.

5. The address format extension defined in this document and MAP share the same suffix coding scheme (IPv4 address + PSID).

6. The AFT and MAP share the same port mapping algorithm (generalized modulus algorithm).

6. IANA Considerations

This memo adds no new IANA considerations.

7. Security Considerations

There is no special security consideration.

8. Acknowledgements

Thanks to Wojciech Dec, Remi Despres, Ole Troan, Rajiv Asati, Chongfeng Xie, Qiong Sun, Fred Baker, Wing Dan, M. Boucadair and all the friends for the fruitful discussions during the unification of the port mapping algorithm of [I-D.mdt-softwire-mapping-address-and-port]. Also thanks to Wentao Shang, Guoliang Han, Yu Zhai, Haotao Zhang and Weicai Wang for their contributions.

9. References

9.1. Normative References

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <https://www.rfc-editor.org/info/rfc3633>.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010,

<https://www.rfc-editor.org/info/rfc6052>.

9.2. Informative References

[I-D.mdt-softwire-mapping-address-and-port]
Bao, C., Troan, O., Matsushima, S., Murakami, T., and X. Li, "Mapping of Address and Port (MAP)", draft-mdt-softwire-mapping-address-and-port-03 (work in progress), January 2012.

[I-D.xli-behave-divi]
Bao, C., Li, X., Zhai, Y., and W. Shang, "dIVI: Dual-Stateless IPv4/IPv6 Translation", draft-xli-behave-divi-07 (work in progress), January 2017.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, DOI 10.17487/RFC6144, April 2011, <https://www.rfc-editor.org/info/rfc6144>.

[RFC6346] Bush, R., Ed., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, DOI 10.17487/RFC6346, August 2011, <https://www.rfc-editor.org/info/rfc6346>.

[RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", RFC 7597, DOI 10.17487/RFC7597, July 2015, <https://www.rfc-editor.org/info/rfc7597>.

[RFC7599] Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <https://www.rfc-editor.org/info/rfc7599>.

Authors' Addresses

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

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

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

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

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.

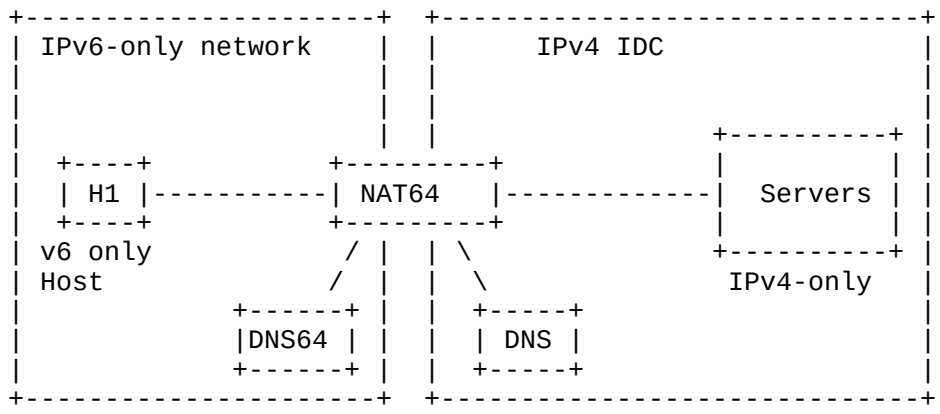


Figure 1: NAT64-CGN Mode Usage

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.

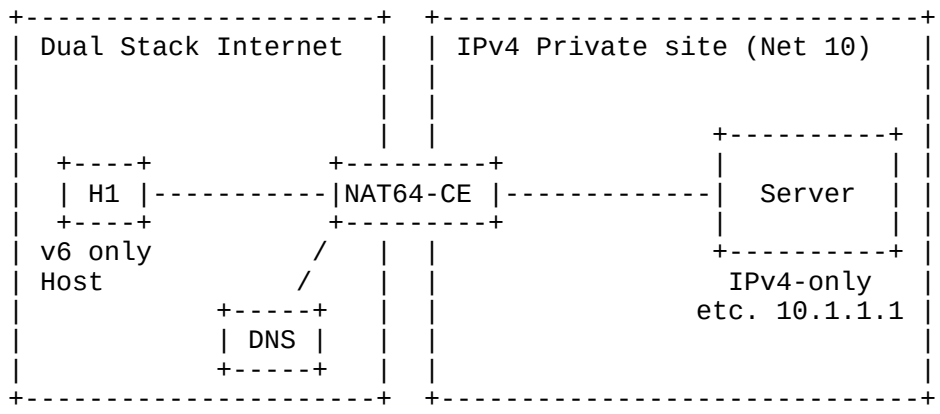


Figure 2: NAT64-CPE Mode Usage

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
Intended status: Standards Track
Expires: June 20, 2014

D. Cheng
Huawei Technologies
J. Korhonen
Broadcom
M. Boucadair
France Telecom
S. Sivakumar
Cisco Systems

December 17, 2013

RADIUS Extensions for Port Set Configuration and Reporting
draft-cheng-behave-cgn-cfg-radius-ext-07

Abstract

This document defines new RADIUS attributes that can be used by a device implementing port ranges to communicate with a RADIUS server to configure and/or report TCP/UDP port sets and ICMP identifiers mapping behavior for specific hosts. This mechanism can be used in various deployment scenarios such as CGN, NAT64, Provider WiFi Gateway, etc.

This document does not make any assumption about the deployment context.

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 June 20, 2014.

Table of Contents

1. Introduction

In a broadband network, customer information is usually stored on a RADIUS server [RFC2865] and at the time when a user initiates an IP connection request, the RADIUS server will populate the user's configuration information to the Network Access Server (NAS), which is usually co-located with the Border Network Gateway (BNG), after

the connection request is granted. The Carrier Grade NAT (CGN) function may also implemented on the BNG, and therefore CGN TCP/UDP port (or ICMP identifier) mapping behavior can be configured on the RADIUS server as part of the user profile, and populated to the NAS in the same manner. In addition, during the operation, the CGN can also convey port/identifier mapping behavior specific to a user to the RADIUS server, as part of the normal RADIUS accounting process.

The CGN device that communicates with a RADIUS server using RADIUS extensions defined in this document may perform NAT44 [RFC3022], NAT64 [RFC6146], or Dual-Stack Lite AFTR [RFC6333] function.

For the CGN example, when IP packets traverse a CGN, it would perform TCP/UDP source port mapping or ICMP identifier mapping as required. A TCP/ UDP source port or ICMP identifier, along with source IP address, destination IP address, destination port and protocol identifier if applicable, uniquely identify a session. Since the number space of TCP/UDP ports and ICMP identifiers in CGN's external realm is shared among multiple users assigned with the same IPv4 address, the total number of a user's simultaneous IP sessions is likely to subject to port quota.

The attributes defined in this document may also be used to report the assigned port set in some deployment such as Provider Wi-Fi [I-D.gundavelli-v6ops-community-wifi-svcs]. For example, a visiting host can be managed by a CPE which will need to report the assigned port set to the service platform. This is required for identification purposes (see WT-146 for example).

This document proposes three new RADIUS attributes as RADIUS protocol's extensions, and they are used for separate purposes as follows:

- A session limit is configured on a RADIUS server based on service agreement with a subscriber, and this parameter imposes the limit of total number of TCP/UDP ports and/or ICMP identifiers that the subscriber can use. Alternately, a separate session limit may be configured to limit the number of TCP ports, UDP ports, or the sum of the two, and ICMP identifiers, respectively, that the user can use. The session limit is carried by a new RADIUS attribute Port-Session-Limit, which is included in a RADIUS Access-Accept message sent by the RADIUS server to port-based device. This new RADIUS attribute can also be included in a RADIUS CoA message sent by the RADIUS server to the port-based device in order to change the session limit previously configured.

- A port-based device may allocate or de-allocate a set of TCP/UDP ports or ICMP identifiers for a specific subscriber. When it does

so, the associated session range along with the shared IPv4 address can be conveyed to the RADIUS server as part of the accounting process. These parameters are carried by a new RADIUS attribute Port-Session-Range, which is included in a RADIUS Accounting- Request message sent by the port-based device to the RADIUS server.

- A user may require the port-based device to perform port forwarding function, i.e., a port mapping is pre-configured on the port-based so that inbound IP packets sent by some applications from the port-based external realm can pass through that device and reach the user. The port mapping information includes the port-based device internal port, external port, and may also include the associated internal IPv4 or IPv6 address, and is carried by a new RADIUS attribute Port- Forwarding-Map, which is included in a RADIUS Access-Accept message sent by the RADIUS server to the port-based device. This new RADIUS attribute can also be included in a RADIUS CoA message sent by the RADIUS server to the port-based device in order to change the forwarding port mapping previously configured.

2. Terminology

Some terms that are used in this document are listed as follows:

- Session Limit - This is the maximum number of TCP ports, or UDP ports, or the total of the two, or ICMP identifiers, or the total of the three, that a device supporting port ranges can use when performing mapping on TCP/ UDP ports or ICMP identifiers for a specific user.

- Session Range - This specifies a set of TCP/UDP port numbers or ICMP identifiers, indicated by the port/identifier with the smallest numerical number and the port/identifier with the largest numerical number, inclusively.

- Internal IP Address - The IP address that is used as a source IP address in an outbound IP packet sent toward a device supporting port ranges in the internal realm. In IPv4 case, it is typically a private address [RFC1918].

- External IP Address - The IP address that is used as a source IP address in an outbound IP packet after traversing a device supporting port ranges in the external realm. In IPv4 case, it is typically a global and routable IP address.

- Internal Port - The internal port is a UDP or TCP port, or an ICMP identifier, which is allocated by a host or application behind a

device supporting port ranges for an outbound IP packet in the internal realm.

- o External Port - The external port is a UDP or TCP port, or an ICMP identifier, which is allocated by a device supporting port ranges upon receiving an outbound IP packet in the internal realm, and is used to replace the internal port that is allocated by a user or application.

- o External realm - The networking segment where IPv4 public addresses are used in respective of the device supporting port ranges.

- o Internal realm - The networking segment that is behind a device supporting port ranges and where IPv4 private addresses are used.

- o Mapping - This term in this document associates with a device supporting port ranges for a relationship between an internal IP address, internal port and the protocol, and an external IP address, external port, and the protocol.

- o Port-based device - A device that is capable of providing IP address and TCP/UDP port mapping services and in particular, with the granularity of one or more subsets within the 16-bit TCP/UDP port number range. A typical example of this device can be a CGN, CPE, Provider Wi-Fi Gateway, etc.

Note the terms "internal IP address", "internal port", "internal realm", "external IP address", "external port", "external realm", and "mapping" and their semantics are the same as in [RFC6887], and [RFC6888].

3. RADIUS Attributes

[Discussion: should these attributes be allocated from the extended RADIUS attribute code space?]

[Discussion: Should we define a dedicated attribute (port_set_policies) to configure the following policies: (1) enforce port randomization, (2) include/exclude the WKP in the port assignment, (3) preserve parity, (4) quota for explicit port mapping, (5) DSCP marking policy, (6) Port hold down timer, (7) port hold down pool, etc. Perhaps we don’t need to cover all these parameters.]

3.1. Port-Session-Limit Attribute

This attribute is of type complex [RFC6158] and specifies the limit of TCP ports, or UDP ports, or the sum of the two, or ICMP identifiers, or the sum of the three, which is configured on a device supporting port ranges corresponding to a specific subscriber.

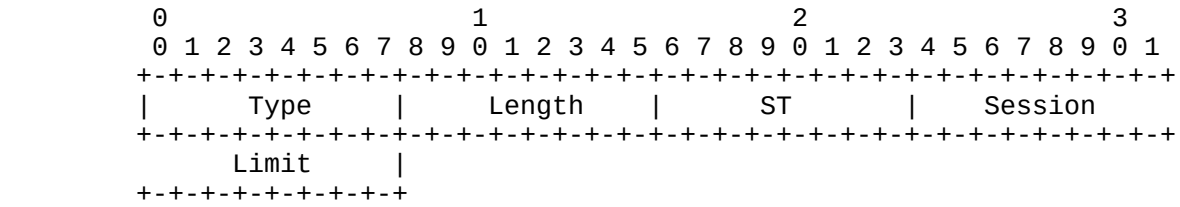
The Port-Session-Limit MAY appear in an Access-Accept packet, it MAY also appear in an Access-Request packet as a hint by the device supporting port ranges, which is co-allocated with the NAS, to the RADIUS server as a preference, although the server is not required to honor such a hint.

The Port-Session-Limit MAY appear in an CoA-Request packet.

The Port-Session-Limit MAY appear in an Accounting-Request packet.

The Port-Session-Limit MUST NOT appear in any other RADIUS packets.

The format of the Port-Session-Limit RADIUS attribute format is shown below. The fields are transmitted from left to right.



Type:

TBA1 for Port-Session-Limit.

Length:

5 octets. This field indicates the total length in octets of this attribute including the Type and the Length field.

ST (Session Type):

This one octet field contains an enumerated value that indicates the applicability of the Session Limit as follows:

0:

The limit as specified is applied to each transport protocol (TCP/UDP) and ICMP Identifiers as a whole.

1:

The limit as specified is applied to TCP and UDP ports.

2:

The limit as specified is applied to TCP ports.

3:

The limit as specified is applied to UDP ports.

4:

The limit as specified is applied to ICMP Identifiers.

5-255:

These values are undefined.

Session Limit:

This field contains the maximum number that is assigned to the transport sessions depending on the value in the Session Type (ST) field, that the specific user can use.

3.2. Port-Session-Range Attribute

This attribute is of type complex [RFC6158] and contains a range of numbers for TCP ports or UDP ports, or both, or for ICMP Identifiers, which has been allocated or de-allocated by a device supporting port ranges for a given subscriber, along with an external IPv4 address that is associated with any TCP/UDP port or ICMP identifier in the range.

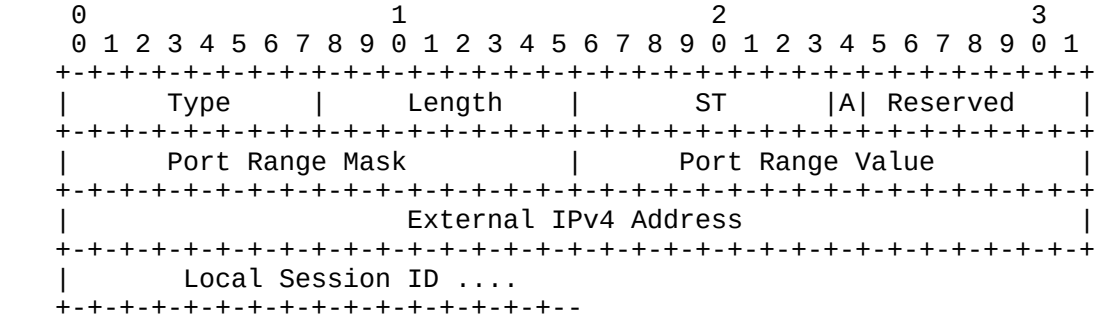
In some CGN deployment scenarios as described such as L2NAT [I-D.miles-behave-l2nat], DS-Extra-Lite [RFC6619] and Lightweight 4over6 [I-D.ietf-softwire-lw4over6], parameters at a customer premise such as MAC address, interface ID, VLAN ID, PPP session ID, IPv6 prefix, VRF ID, etc., may also be required to pass to the RADIUS server as part of the accounting record.

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

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

The port range follows the encoding specified in [RFC6431]; as such both contiguous and non-contiguous port sets are supported.

The format of the Port-Session-Range RADIUS attribute format is shown below. The fields are transmitted from left to right.



Type:

TBA2 for Port-Session-Range.

Length:

12 octets plus the length of optional field Local Session ID. This field indicates the total length in octets of this attribute including the Type and the Length field.

ST (Session Type):

This one octet field contains an enumerated value that indicates the semantics of the session range. The values follow the Session Type encoding defined in Section 3.1 except that the following values are not valid in scope of this attribute:

0:

The limit as specified is applied to the sum of TCP ports, UDP ports, and ICMP Identifiers as a whole.

A-bit Flag:

This field is set to 0 or 1, indicates that the session range has been allocated or de-allocated, respectively, by the device supporting port ranges.

Reserved:

This field MUST be set to zero by the sender and ignored by the receiver.

Port Range Mask:

The Port Range Mask indicates the position of the bits that are used to build the Port Range Value. By default, no PRM value is assigned. The 1 values in the Port Range Mask indicate by their position the significant bits of the Port Range Value. Refer to [RFC6431] for more details.

Port Range Value:

The PRV indicates the value of the significant bits of the Port Mask. By default, no PRV is assigned. Refer to [RFC6431] for more details.

External IPv4 Address:

This is an optional field. If present, this field contains the IPv4 address assigned to the associated subscriber to be used in the external realm. If set to 0/0, the allocation address policy is local to the device supporting port ranges.

Local Session ID:

This is an optional field and if presents, it contains a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/ prefix, etc. The length of this field equals to the total attribute length minus 12 octets. If this field is not present, the port range policies must be enforced to all subscribers using a local subscriber identifier.

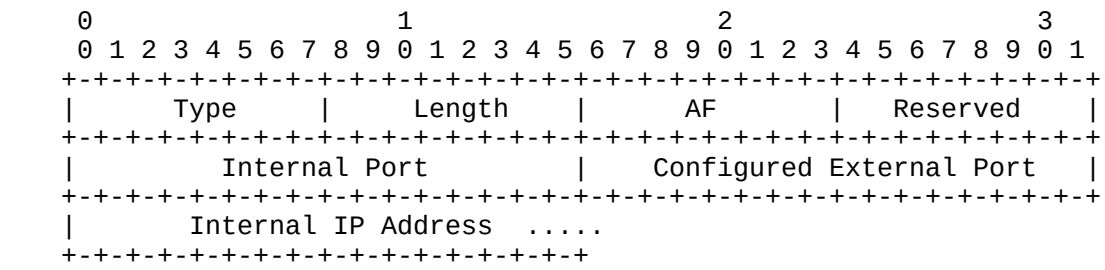
3.3. Port-Forwarding-Map Attribute

This attribute is type of complex [RFC6158] and contains 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 attribute may contain a 32-bit IPv4 address or a 128-bit IPv6 address, respectively, as their respective NAT mappings 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.

The attribute MUST NOT appear in any other RADIUS packets.

The format of the Port-Forwarding-Map RADIUS attribute format is shown below. The fields are transmitted from left to right.



Type:

TBA3 for Port-Forwarding-Map.

Length:

This field indicates the total length in octets of this attribute including the Type and the Length field. Depending on the value of the AF field, the length could be 8, 12 or 24 octets.

AF (Address Family):

This one octet field contains a value that indicates address family of the internal IP address at the mapping as follows:

0:

There is no internal address attached.

1:

The internal address is an IPv4 address.

2:

The internal address is an IPv6 address.

3-255:

Unused.

[Discussion: should we use IANA assigned protocol numbers here?]

Reserved:

This field is set to zero by the sender and ignored by the receiver.

Internal Port:

This field contains the internal port for the CGN mapping.

Configured External Port:

This field contains the external port for the CGN mapping.

Internal IP Address:

This field may or may not present, and when it does, contains the internal IPv4 or IPv6 address for the CGN mapping.

4. Applications, Use Cases and Examples

This section describes some applications and use cases to illustrate the use of the RADIUS port set attributes.

4.1. Managing CGN Port Behavior using RADIUS

In a broadband network, customer information is usually stored on a RADIUS server, and the BNG hosts the NAS. The communication between the NAS and the RADIUS server is triggered by a subscriber when the user signs in to the Internet service, where either PPP or DHCP/DHCPv6 is used. When a user signs in, the NAS sends a RADIUS Access-Request message to the RADIUS server. The RADIUS server validates the request, and if the validation succeeds, it in turn sends back a RADIUS Access-Accept message. The Access-Accept message carries configuration information specific to that user, back to the NAS, where some of the information would pass on to the requesting user via PPP or DHCP/DHCPv6.

A CGN function in a broadband network would most likely reside on a BNG. In that case, parameters for CGN port/identifier mapping behavior for users can be configured on the RADIUS server. When a user signs in to the Internet service, the associated parameters can be conveyed to the NAS, and proper configuration is accomplished on the CGN device for that user.

Also, CGN operation status such as CGN port/identifier allocation and de-allocation for a specific user on the BNG can also be transmitted back to the RADIUS server for accounting purpose using the RADIUS protocol.

RADIUS protocol has already been widely deployed in broadband networks to manage BNG, thus the functionality described in this specification introduces little overhead to the existing network operation.

In the following sub-sections, we describe how to manage CGN behavior using RADIUS protocol, with required RADIUS extensions proposed in Section 3.

4.1.1. Configure CGN Session Limit

In the face of IPv4 address shortage, there are currently proposals to multiplex multiple subscribers' connections over a smaller number of shared IPv4 addresses, such as Carrier Grade NAT [RFC6888], Dual-Stack Lite [RFC6333], NAT64 [RFC6146], etc. As a result, a single IPv4 public address may be shared by hundreds or even thousands of subscribers. As indicated in [RFC6269], it is therefore necessary to impose limits on the total number of ports available to an individual subscriber to ensure that the shared resource, i.e., the IPv4 address remains available in some capacity to all the subscribers using it, and port limiting is also documented in [RFC6888] as a requirement.

There are two practical granularities to impose such a limit. One is to define a session limit that is imposed to the total number of TCP and UDP ports, plus the number of ICMP identifiers, for a specific subscriber. Alternatively, a session limit can be specified for the sum of TCP ports and UDP ports, or a separate session limit for TCP ports and UDP ports, respectively, and another session limit for ICMP identifiers.

The per-subscriber based session limit(s) is configured on a RADIUS server, along with other user information such as credentials. The value of these session limit(s) is based on service agreement and its specification is out of the scope of this document.

When a subscriber signs in to the Internet service successfully, the session limit(s) for the subscriber is passed to the BNG based NAS, where CGN also locates, using a new RADIUS attribute called Port-Session-Limit (defined in Section 3.1), along with other configuration parameters. While some parameters are passed to the subscriber, the session limit(s) is recorded on the CGN device for imposing the usage of TCP/UDP ports and ICMP identifiers for that subscriber.

Figure 1 illustrates how RADIUS protocol is used to configure the maximum number of TCP/UDP ports for a given subscriber on a NAT44 device.

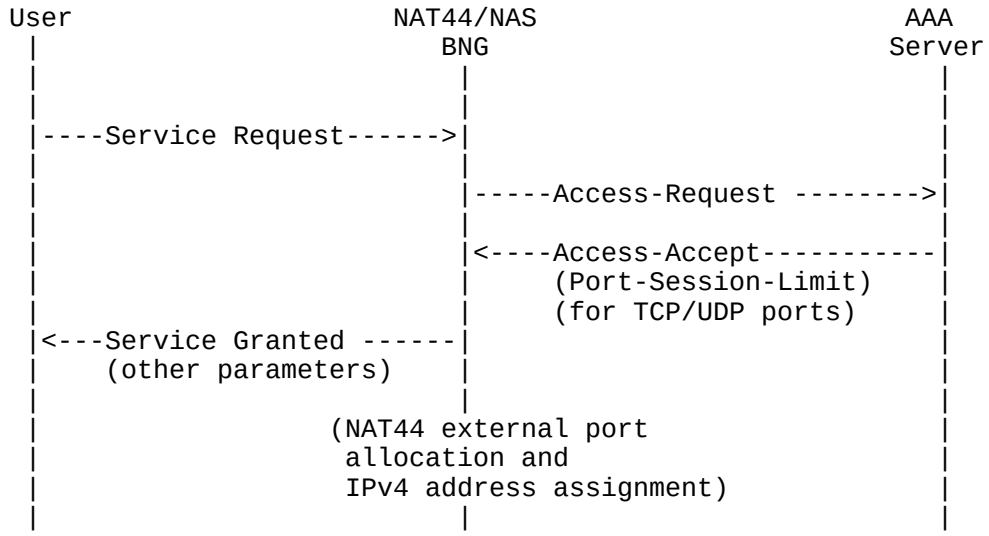


Figure 1: RADIUS Message Flow for Configuring NAT44 Port Limit

The session limit(s) created on a CGN device for a specific user using RADIUS extension may be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS attribute. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new session limit replaces the previous one.

Figure 2 illustrates how RADIUS protocol is used to increase the TCP/UDP port limit from 1024 to 2048 on a NAT44 device for a specific user.

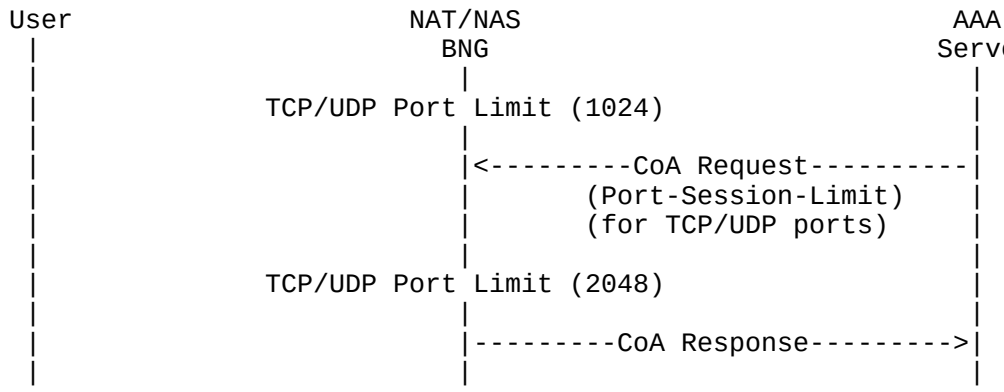


Figure 2: RADIUS Message Flow for changing a user's NAT44 port limit

4.1.2. Report CGN Session Allocation or De-allocation

Upon obtaining the session limit(s) for a subscriber, the CGN device needs to allocate a TCP/UDP port or an ICMP identifiers for the subscriber when receiving a new IP flow sent from that subscriber.

As one practice, a CGN may allocate a bulk of TCP/UDP ports or ICMP identifiers once at a time for a specific user, instead of one port/identifier at a time, and within each session bulk, the ports/identifiers may be randomly distributed or in consecutive fashion. When a CGN device allocates bulk of TCP/UDP ports and ICMP identifiers, the information can be easily conveyed to the RADIUS server by a new RADIUS attribute called the CGN-Session-Range (defined in Section 3.2). The CGN device may allocate one or more TCP/UDP port ranges or ICMP identifier ranges, or generally called session ranges, where each range contains a set of numbers representing TCP/UDP ports or ICMP identifiers, and the total number of sessions must be less or equal to the associated session limit defined for that subscriber. A CGN device may choose to allocate a small session range, and allocate more at a later time as needed; such practice is good because its randomization in nature.

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

When a subscriber initiates an IP flow, the CGN device randomly selects a TCP/UDP port or ICMP identifier from the associated and pre-allocated session range for that subscriber to replace the original source TCP/UDP port or ICMP identifier, along with the replacement of the source IP address by the shared IPv4 address.

A CGN device may decide to "free" a previously assigned set of TCP/ UDP ports or ICMP identifiers that have been allocated for a specific subscriber but not currently in use, and with that, the CGN device must send the information of the de-allocated session range along with the shared IPv4 address to the RADIUS server.

Figure 3 illustrates how RADIUS protocol is used to report a set of ports allocated and de-allocated, respectively, by a NAT44 device for a specific user to the RADIUS server.

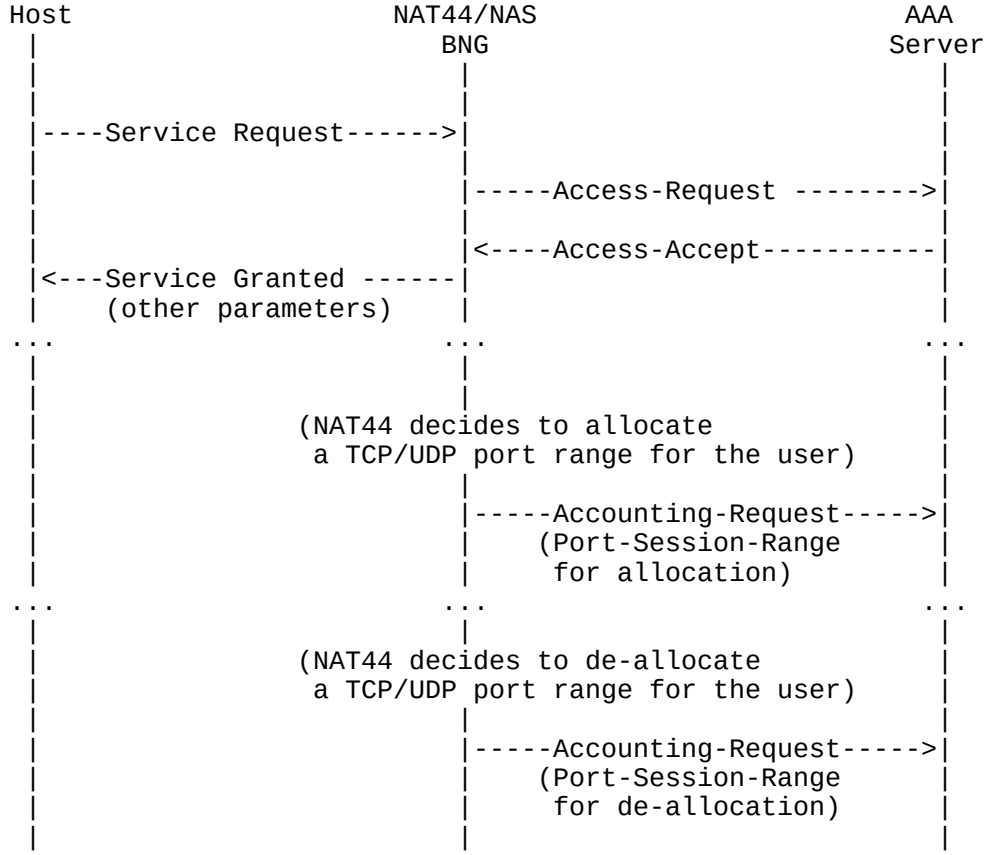


Figure 3: RADIUS Message Flow for reporting NAT44 allocation/de-allocation of a port set

4.1.3. Configure CGN Forwarding Port Mapping

In most 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 CGN device to pass through and "port forwarded" to the correct user located behind the CGN device.

Port Control Protocol [RFC6887], provides a mechanism to create a mapping from an external IP address and port to an internal IP address and port on a CGN device just to achieve the "port forwarding" purpose. PCP is a server-client protocol capable of creating or deleting a mapping along with a rich set of features on a CGN 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 CGN device.

Figure 4 illustrates how RADIUS protocol is used to configure a forwarding port mapping on a NAT44 device by using RADIUS protocol.

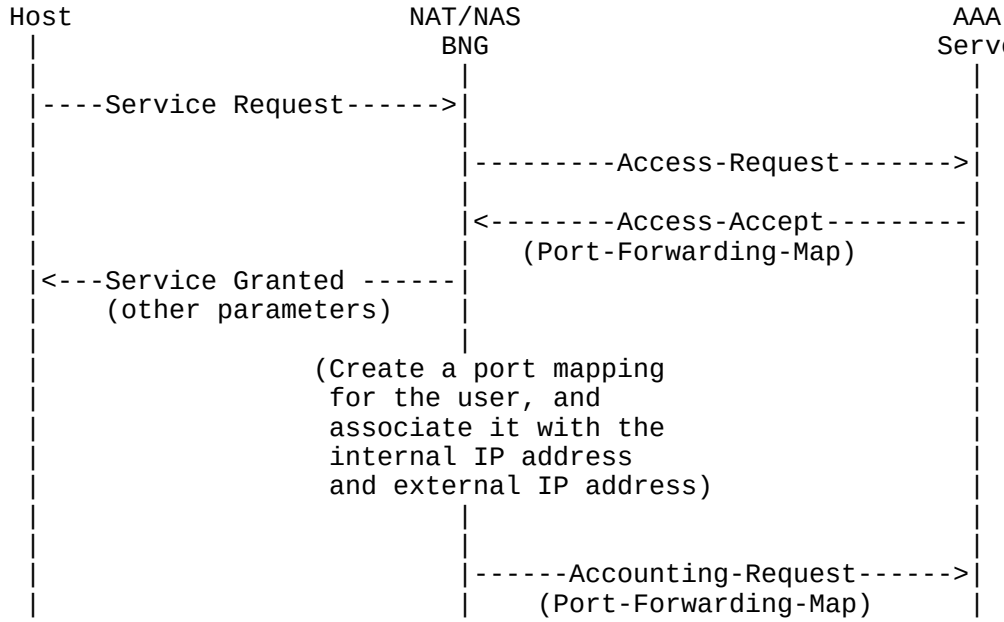


Figure 4: RADIUS Message Flow for configuring a forwarding port mapping

A port forwarding mapping that is created on a CGN device using RADIUS extension as described above may also be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS associate. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new port forwarding mapping then replaces the previous one.

Figure 5 illustrates how RADIUS protocol is used to change an existing port mapping from (a:X) to (a:Y), where "a" is an internal port, and "X" and "Y" are external ports, respectively, for a specific user with a specific IP address

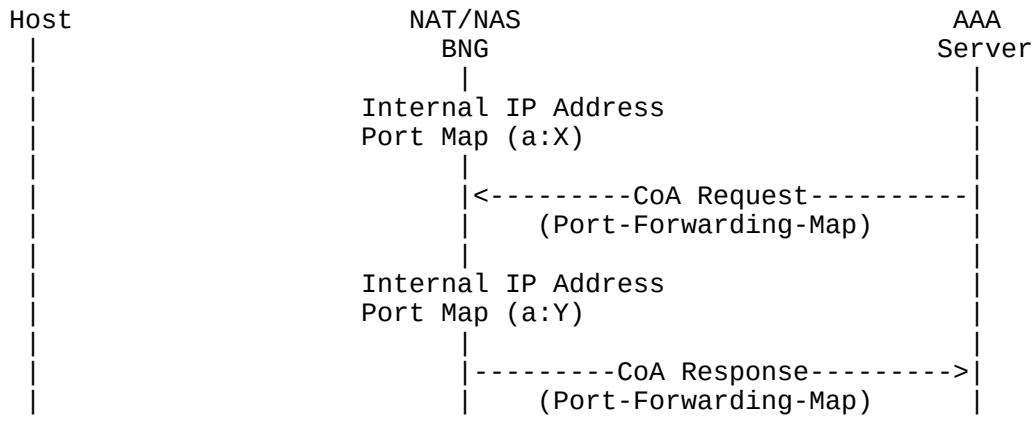


Figure 5: RADIUS Message Flow for changing a user's forwarding port mapping

4.1.4. An Example

An Internet Service Provider (ISP) assigns TCP/UDP 500 ports for the subscriber Joe. This number is the limit that can be used for TCP/UDP ports on a NAT44 device for Joe, and is configured on a RADIUS server. Also, Joe asks for a pre-defined port forwarding mapping on the NAT44 device for his web cam applications (external port 5000 maps to internal port 80).

When Joe successfully connects to the Internet service, the RADIUS server conveys the TCP/UDP port limit (1000) and the forwarding port mapping (external port 5000 to internal port 80) to the NAT44 device, using Port-Session-Limit attribute and Port-Forwarding-Map attribute, respectively, carried by an Access-Accept message to the BNG where NAS and CGN co-located.

Upon receiving the first outbound IP packet sent from Joe's laptop, the NAT44 device decides to allocate a small port pool that contains 40 consecutive ports, from 3500 to 3540, inclusively, and also assign a shared IPv4 address 192.0.2.15, for Joe. The NAT44 device also randomly selects one port from the allocated range (say 3519) and use that port to replace the original source port in outbound IP packets.

For accounting purpose, the NAT44 device passes this port range (3500-3540) and the shared IPv4 address 192.0.2.15 together to the RADIUS server using Port-Session-Range attribute carried by an Accounting-Request message.

When Joe works on more applications with more outbound IP sessions and the port pool (3500-3540) is close to exhaust, the NAT44 device

allocates a second port pool (8500-8800) in a similar fashion, and also passes the new port range (8500-8800) and IPv4 address 192.0.2.15 together to the RADIUS server using Port-Session-Range attribute carried by an Accounting-Request message. Note when the CGN allocates more ports, it needs to assure that the total number of ports allocated for Joe is within the limit.

Joe decides to upgrade his service agreement with more TCP/UDP ports allowed (up to 1000 ports). The ISP updates the information in Joe's profile on the RADIUS server, which then sends a CoA-Request message that carries the Port-Session-Limit attribute with 1000 ports to the NAT44 device; the NAT44 device in turn sends back a CoA-ACK message. With that, Joe enjoys more available TCP/UDP ports for his applications.

When Joe travels, most of the IP sessions are closed with their associated TCP/UDP ports released on the NAT44 device, which then sends the relevant information back to the RADIUS server using Port-Session-Range attribute carried by Accounting-Request message.

Throughout Joe's connection with his ISP Internet service, applications can communicate with his web cam at home from external realm directly traversing the pre-configured mapping on the CGN device.

When Joe disconnects from his Internet service, the CGN device will de-allocate all TCP/UDP ports as well as the port-forwarding mapping, and send the relevant information to the RADIUS server.

4.2. Report Assigned Port Set for a Visiting UE

Figure 6 illustrates an example of the flow exchange which occurs when a visiting UE connects to a CPE offering Wi-Fi service.

For identification purposes (see [RFC6967]), once the CPE assigns a port set, it issues a RADIUS message to report the assigned port set.

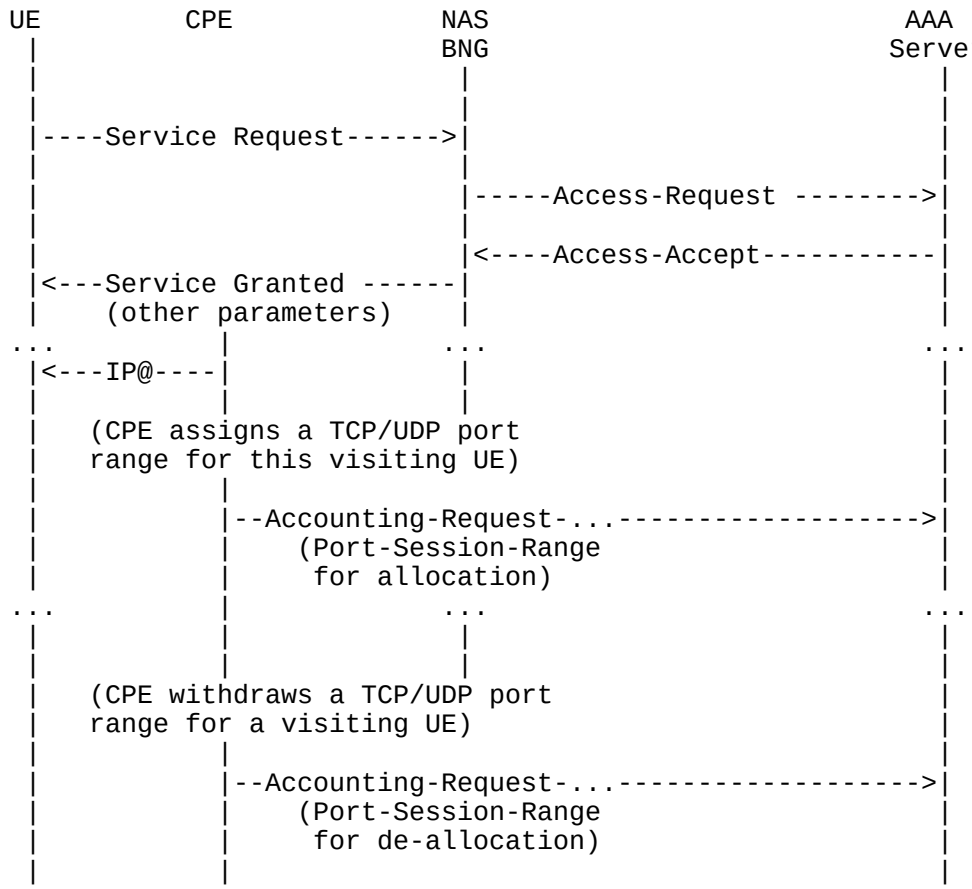


Figure 6: RADIUS Message Flow for reporting CPE allocation/de-allocation of a port set to a visiting UE

5. Table of Attributes

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

Request	Accept	Reject	Challenge	Acct. Request	#	Attribute
0-1	0-1	0	0	0-1	TBA1	Port-Session-Limit
0	0	0	0	0-1	TBA2	Port-Session-Range
0-1	0-1	0	0	0-1	TBA3	Port-Forwarding-Map

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

0 This attribute MUST NOT be present in packet.
0+ Zero or more instances of this attribute MAY be present in packet.
0-1 Zero or one instance of this attribute MAY be present in packet.

6. Security Considerations

This document does not introduce any security issue than what has been identified in [RFC2865].

7. IANA Considerations

7.1. RADIUS Attributes

This document requires new code point assignment for the three new RADIUS attributes as follows:

- Port-Session-Limit
- Port-Session-Range
- Port-Forwarding-Map

7.2. Name Spaces

This document establishes a new name space for Session Type (see Section 3.1 for the initial reservation of values. The allocation of future values is according to RFC Required policy [RFC5226].

8. Acknowledgements

Many thanks to Dan Wing, Roberta Maglione, Daniel Derksen, and David Thaler for their useful comments and suggestions.

9. References

9.1. 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.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

9.2. Informative References

[I-D.gundavelli-v6ops-community-wifi-svcs]
Gundavelli, S., Grayson, M., Seite, P., and Y. Lee, "Service Provider Wi-Fi Services Over Residential Architectures", draft-gundavelli-v6ops-community-wifi-svcs-06 (work in progress), April 2013.

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-03 (work in progress), November 2013.

[I-D.miles-behave-l2nat]
Miles, D. and M. Townsley, "Layer2-Aware NAT", draft-miles-behave-l2nat-00 (work in progress), March 2009.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 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.

[RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

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

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

[RFC6431] Boucadair, M., Levis, P., Bajko, G., Savolainen, T., and T. Tsou, "Huawei Port Range Configuration Options for PPP IP Control Protocol (IPCP)", RFC 6431, November 2011.

[RFC6619] Arkko, J., Eggert, L., and M. Townsley, "Scalable Operation of Address Translators with Per-Interface Bindings", RFC 6619, June 2012.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

[RFC6888] Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, April 2013.

[RFC6967] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Potential Solutions for Revealing a Host Identifier (HOST_ID) in Shared Address Deployments", RFC 6967, June 2013.

Authors' Addresses

Dean Cheng
Huawei Technologies
2330 Central Expressway
Santa Clara, California 95050
USA

Email: dean.cheng@huawei.com

Jouni Korhonen
Broadcom
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

Mohamed Boucadair
France Telecom
Rennes
France

Email: mohamed.boucadair@orange.com

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, North Carolina
USA

Email: ssenthil@cisco.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.

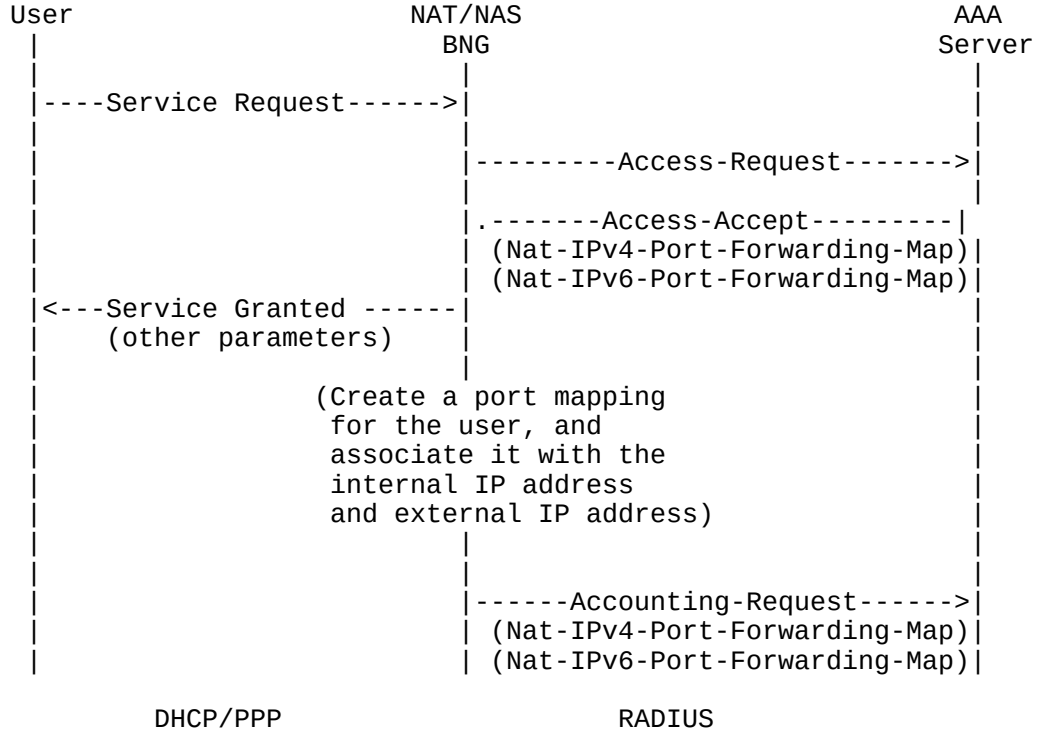


Figure 1: RADIUS Message Flow

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.

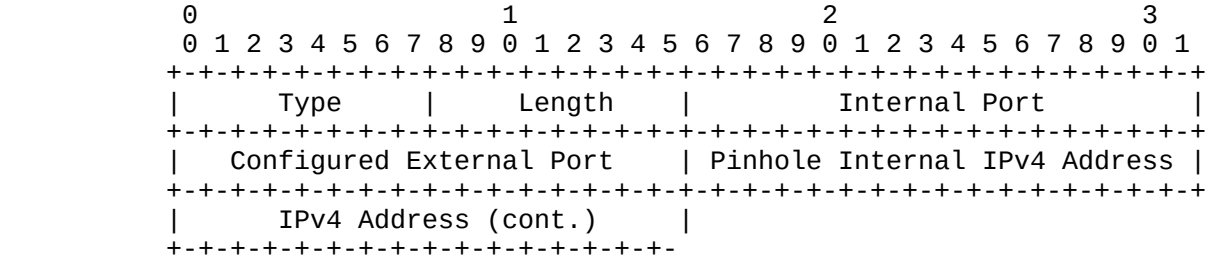


Figure-2 Nat-IPv4-Forwarding-Port-Map Attribute Format

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

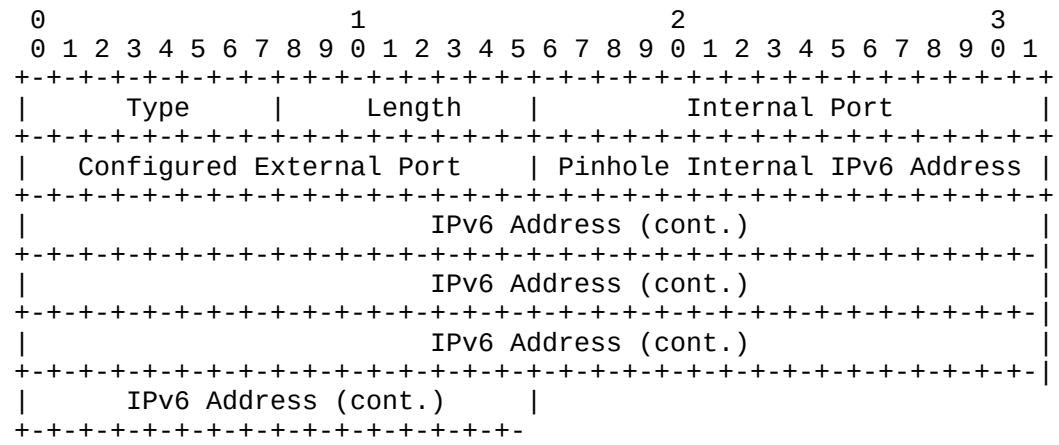


Figure-3 Nat-IPv6-Forwarding-Port-Map Attribute Format

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 14, 2015

C. Donley
CableLabs
C. Grundemann
Internet Society
V. Sarawat
K. Sundaresan
CableLabs
O. Vautrin
Juniper Networks

December 11, 2014

Deterministic Address Mapping to Reduce Logging in Carrier Grade NAT Deployments
draft-donley-behave-deterministic-cgn-09

Abstract

In some instances, Service Providers have a legal logging requirement to be able to map a subscriber's inside address with the address used on the public Internet (e.g. for abuse response). Unfortunately, many Carrier Grade NAT logging solutions require active logging of dynamic translations. Carrier Grade NAT port assignments are often per-connection, but could optionally use port ranges. Research indicates that per-connection logging is not scalable in many residential broadband services. This document suggests a way to manage Carrier Grade NAT translations in such a way as to significantly reduce the amount of logging required while providing traceability for abuse response. IPv6 is, of course, the preferred solution. While deployment is in progress, service providers are forced by business imperatives to maintain support for IPv4. This note addresses the IPv4 part of the network when a Carrier Grade NAT solution is in use.

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 June 14, 2015.

Table of Contents

1. Introduction

It is becoming increasingly difficult to obtain new IPv4 address assignments from Regional/Local Internet Registries due to depleting supplies of unallocated IPv4 address space. To meet the growing demand for Internet connectivity from new subscribers, devices, and service types, some operators will be forced to share a single public IPv4 address among multiple subscribers using techniques such as Carrier Grade Network Address Translation (CGN) [RFC6264] (e.g., NAT444 [I-D.shirasaki-nat444], DS-Lite [RFC6333], NAT64 [RFC6146] etc.). However, address sharing poses additional challenges to operators when considering how they manage service entitlement, public safety requests, or attack/abuse/fraud reports [RFC6269]. In order to identify a specific user associated with an IP address in response to such a request or for service entitlement, an operator will need to map a subscriber's internal source IP address and source port with the global public IP address and source port provided by the CGN for every connection initiated by the user.

CGN connection logging satisfies the need to identify attackers and respond to abuse/public safety requests, but it imposes significant operational challenges to operators. In lab testing, we have observed CGN log messages to be approximately 150 bytes long for NAT444 [I-D.shirasaki-nat444], and 175 bytes for DS-Lite [RFC6333] (individual log messages vary somewhat in size). Although we are not aware of definitive studies of connection rates per subscriber, reports from several operators in the US sets the average number of connections per household at approximately 33,000 connections per day. If each connection is individually logged, this translates to a data volume of approximately 5 MB per subscriber per day, or about 150 MB per subscriber per month; however, specific data volumes may vary across different operators based on myriad factors. Based on available data, a 1-million subscriber service provider will generate approximately 150 terabytes of log data per month, or 1.8 petabytes per year. Note that many Service Providers compress log data after collection; compression factors of 2:1 or 3:1 are common.

The volume of log data poses a problem for both operators and the public safety community. On the operator side, it requires a significant infrastructure investment by operators implementing CGN. It also requires updated operational practices to maintain the logging infrastructure, and requires approximately 23 Mbps of bandwidth between the CGN devices and the logging infrastructure per 50,000 users. On the public safety side, it increases the time required for an operator to search the logs in response to an abuse report, and could delay investigations. Accordingly, an international group of operators and public safety officials

approached the authors to identify a way to reduce this impact while improving abuse response.

The volume of CGN logging can be reduced by assigning port ranges instead of individual ports. Using this method, only the assignment of a new port range is logged. This may massively reduce logging volume. The log reduction may vary depending on the length of the assigned port range, whether the port range is static or dynamic, etc. This has been acknowledged in [RFC6269], which recommends source port logging at the server and/or destination logging at the CGN and [I-D.sivakumar-behave-nat-logging], which describes information to be logged at a NAT.

However, the existing solutions still poses an impact on operators and public safety officials for logging and searching. Instead, CGNs could be designed and/or configured to deterministically map internal addresses to {external address + port range} in such a way as to be able to algorithmically calculate the mapping. Only inputs and configuration of the algorithm need to be logged. This approach reduces both logging volume and subscriber identification times. In some cases, when full deterministic allocation is used, this approach can eliminate the need for translation logging.

This document describes a method for such CGN address mapping, combined with block port reservations, that significantly reduces the burden on operators while offering the ability to map a subscriber's inside IP address with an outside address and external port number observed on the Internet.

The activation of the proposed port range allocation scheme is compliant with BEHAVE requirements such as the support of APP.

2. Deterministic Port Ranges

While a subscriber uses thousands of connections per day, most subscribers use far fewer resources at any given time. When the compression ratio (see Appendix B of RFC6269 [RFC6269]) is low (e.g., the ratio of the number of subscribers to the number of public IPv4 addresses allocated to a CGN is closer to 10:1 than 1000:1), each subscriber could expect to have access to thousands of TCP/UDP ports at any given time. Thus, as an alternative to logging each connection, CGNs could deterministically map customer private addresses (received on the customer-facing interface of the CGN, a.k.a., internal side) to public addresses extended with port ranges (used on the Internet-facing interface of the CGN, a.k.a., external side). This algorithm allows an operator to identify a subscriber internal IP address when provided the public side IP and port number without having to examine the CGN translation logs. This prevents an

operator from having to transport and store massive amounts of session data from the CGN and then process it to identify a subscriber.

The algorithmic mapping can be expressed as:

(External IP Address, Port Range) = function 1 (Internal IP Address)

Internal IP Address = function 2 (External IP Address, Port Number)

The CGN SHOULD provide a method for administrators to test both mapping functions (e.g., enter an External IP Address + Port Number and receive the corresponding Internal IP Address).

Deterministic Port Range allocation requires configuration of the following variables:

- Inside IPv4/IPv6 address range (I);
- Outside IPv4 address range (O);
- Compression ratio (e.g. inside IP addresses I/outside IP addresses O) (C);
- Dynamic address pool factor (D), to be added to the compression ratio in order to create an overflow address pool;
- Maximum ports per user (M);
- Address assignment algorithm (A) (see below); and
- Reserved TCP/UDP port list (R)

Note: The inside address range (I) will be an IPv4 range in NAT444 operation (NAT444 [I-D.shirasaki-nat444]) and an IPv6 range in DS-Lite operation (DS-Lite [RFC6333]).

A subscriber is identified by an internal IPv4 address (e.g., NAT44) or an IPv6 prefix (e.g., DS-Lite or NAT64).

The algorithm may be generalized to L2-aware NAT [I-D.miles-behave-l2nat] but this requires the configuration of the Internal interface identifiers (e.g., MAC addresses).

The algorithm is not designed to retrieve an internal host among those sharing the same internal IP address (e.g., in a DS-Lite context, only an IPv6 address/prefix can be retrieved using the

algorithm while the internal IPv4 address used for the encapsulated IPv4 datagram is lost).

Several address assignment algorithms are possible. Using predefined algorithms, such as those that follow, simplifies the process of reversing the algorithm when needed. However, the CGN MAY support additional algorithms. Also, the CGN is not required to support all algorithms described below. Subscribers could be restricted to ports from a single IPv4 address, or could be allocated ports across all addresses in a pool, for example. The following algorithms and corresponding values of A are as follow:

0: Sequential (e.g. the first block goes to address 1, the second block to address 2, etc.)

1: Staggered (e.g. for every n between 0 and ((65536-R)/(C+D))-1 , address 1 receives ports n*C+R, address 2 receives ports (1+n)*C+R, etc.)

2: Round robin (e.g. the subscriber receives the same port number across a pool of external IP addresses. If the subscriber is to be assigned more ports than there are in the external IP pool, the subscriber receives the next highest port across the IP pool, and so on. Thus, if there are 10 IP addresses in a pool and a subscriber is assigned 1000 ports, the subscriber would receive a range such as ports 2000-2099 across all 10 external IP addresses).

3: Interlaced horizontally (e.g. each address receives every Cth port spread across a pool of external IP addresses).

4: Cryptographically random port assignment (Section 2.2 of RFC6431 [RFC6431]). If this algorithm is used, the Service Provider needs to retain the keying material and specific cryptographic function to support reversibility.

5: Vendor-specific. Other vendor-specific algorithms may also be supported.

The assigned range of ports MAY also be used when translating ICMP requests (when re-writing the Identifier field).

The CGN then reserves ports as follows:

1. The CGN removes reserved ports (R) from the port candidate list (e.g., 0-1023 for TCP and UDP). At a minimum, the CGN SHOULD remove system ports (RFC6335) [RFC6335] from the port candidate list reserved for deterministic assignment.

2. The CGN calculates the total compression ratio (C+D), and allocates 1/(C+D) of the available ports to each internal IP address. Specific port allocation is determined by the algorithm (A) configured on the CGN. Any remaining ports are allocated to the dynamic pool.

   Note: Setting D to 0 disables the dynamic pool. This option eliminates the need for per-subscriber logging at the expense of limiting the number of concurrent connections that 'power users' can initiate.

3. When a subscriber initiates a connection, the CGN creates a translation mapping between the subscriber's inside local IP address/port and the CGN outside global IP address/port. The CGN MUST use one of the ports allocated in step 2 for the translation as long as such ports are available. The CGN SHOULD allocate ports randomly within the port range assigned by the deterministic algorithm. This is to increase subscriber privacy. The CGN MUST use the preallocated port range from step 2 for Port Control Protocol (PCP, [RFC6887]) reservations as long as such ports are available. While the CGN maintains its mapping table, it need not generate a log entry for translation mappings created in this step.

4. If D>0, the CGN will have a pool of ports left for dynamic assignment. If a subscriber uses more than the range of ports allocated in step 2 (but fewer than the configured maximum ports M), the CGN assigns a block of ports from the dynamic assignment range for such a connection or for PCP reservations. The CGN MUST log dynamically assigned port blocks to facilitate subscriber-to-address mapping. The CGN SHOULD manage dynamic ports as described in [I-D.tsou-behave-natx4-log-reduction].

5. Configuration of reserved ports (e.g., system ports) is left to operator configuration.

Thus, the CGN will maintain translation mapping information for all connections within its internal translation tables; however, it only needs to externally log translations for dynamically-assigned ports.

2.1. IPv4 Port Utilization Efficiency

For Service Providers requiring an aggressive address sharing ratio, the use of the algorithmic mapping may impact the efficiency of the address sharing. A dynamic port range allocation assignment is more suitable in those cases.

2.2. Planning & Dimensioning

Unlike dynamic approaches, the use of the algorithmic mapping requires more effort from operational teams to tweak the algorithm (e.g., size of the port range, address sharing ratio, etc.). Dedicated alarms SHOULD be configured when some port utilization thresholds are fired so that the configuration can be refined.

The use of algorithmic mapping also affects geolocation. Changes to the inside and outside address ranges (e.g. due to growth, address allocation planning, etc.) would require external geolocation providers to recalibrate their mappings.

2.3. Deterministic CGN Example

To illustrate the use of deterministic NAT, let’s consider a simple example. The operator configures an inside address range (I) of 198.51.100.0/28 [RFC6598] and outside address (O) of 192.0.2.1. The dynamic address pool factor (D) is set to ’2’. Thus, the total compression ratio is 1:(14+2) = 1:16. Only the system ports (e.g. ports < 1024) are reserved (R) . This configuration causes the CGN to preallocate ((65536-1024)/16 =) 4032 TCP and 4032 UDP ports per inside IPv4 address. For the purposes of this example, let’s assume that they are allocated sequentially, where 198.51.100.1 maps to 192.0.2.1 ports 1024-5055, 198.51.100.2 maps to 192.0.2.1 ports 5056-9087, etc. The dynamic port range thus contains ports 57472-65535 (port allocation illustrated in the table below). Finally, the maximum ports/subscriber is set to 5040.

Inside Address / Pool	Outside Address & Port
Reserved	192.0.2.1:0-1023
198.51.100.1	192.0.2.1:1024-5055
198.51.100.2	192.0.2.1:5056-9087
198.51.100.3	192.0.2.1:9088-13119
198.51.100.4	192.0.2.1:13120-17151
198.51.100.5	192.0.2.1:17152-21183
198.51.100.6	192.0.2.1:21184-25215
198.51.100.7	192.0.2.1:25216-29247
198.51.100.8	192.0.2.1:29248-33279
198.51.100.9	192.0.2.1:33280-37311
198.51.100.10	192.0.2.1:37312-41343
198.51.100.11	192.0.2.1:41344-45375
198.51.100.12	192.0.2.1:45376-49407
198.51.100.13	192.0.2.1:49408-53439
198.51.100.14	192.0.2.1:53440-57471
Dynamic	192.0.2.1:57472-65535

When subscriber 1 using 198.51.100.1 initiates a low volume of connections (e.g. < 4032 concurrent connections), the CGN maps the outgoing source address/port to the preallocated range. These translation mappings are not logged.

Subscriber 2 concurrently uses more than the allocated 4032 ports (e.g. for peer-to-peer, mapping, video streaming, or other connection-intensive traffic types), the CGN allocates up to an additional 1008 ports using bulk port reservations. In this example, subscriber 2 uses outside ports 5056-9087, and then 100-port blocks between 58000-58999. Connections using ports 5056-9087 are not logged, while 10 log entries are created for ports 58000-58099, 58100-58199, 58200-58299, ..., 58900-58999.

In order to identify a subscriber behind a CGN (regardless of port allocation method), public safety agencies need to collect source address and port information from content provider log files. Thus, content providers are advised to log source address, source port, and timestamp for all log entries, per [RFC6302]. If a public safety agency collects such information from a content provider and reports abuse from 192.0.2.1, port 2001, the operator can reverse the mapping algorithm to determine that the internal IP address subscriber 1 has been assigned generated the traffic without consulting CGN logs (by correlating the internal IP address with DHCP/PPP lease connection records). If a second abuse report comes in for 192.0.2.1, port 58204, the operator will determine that port 58204 is within the dynamic pool range, consult the log file, correlate with connection

records, and determine that subscriber 2 generated the traffic (assuming that the public safety timestamp matches the operator timestamp. As noted in RFC6292 [RFC6292], accurate time-keeping (e.g., use of NTP or Simple NTP) is vital).

In this example, there are no log entries for the majority of subscribers, who only use pre-allocated ports. Only minimal logging would be needed for those few subscribers who exceed their pre-allocated ports and obtain extra bulk port assignments from the dynamic pool. Logging data for those users will include inside address, outside address, outside port range, and timestamp.

Note that in a production environment, operators are encouraged to consider [RFC6598] for assigning inside addresses.

3. Additional Logging Considerations

In order to be able to identify a subscriber based on observed external IPv4 address, port, and timestamp, an operator needs to know how the CGN was configured with regards to internal and external IP addresses, dynamic address pool factor, maximum ports per user, and reserved port range at any given time. Therefore, the CGN MUST generate a record any time such variables are changed. The CGN SHOULD generate a log message any time such variables are changed. The CGN MAY keep such a record in the form of a router configuration file. If the CGN does not generate a log message, it would be up to the operator to maintain version control of router config changes. Also, the CGN SHOULD generate such a log message once per day to facilitate quick identification of the relevant configuration in the event of an abuse notification.

Such a log message MUST, at minimum, include the timestamp, inside prefix I, inside mask, outside prefix O, outside mask, D, M, A, and reserved port list R; for example:

[Wed Oct 11 14:32:52
2000]:198.51.100.0:28:192.0.2.0:32:2:5040:0:1-1023,5004,5060.

3.1. Failover Considerations

Due to the deterministic nature of algorithmically-assigned translations, no additional logging is required during failover conditions provided that inside address ranges are unique within a given failover domain. Even when directed to a different CGN server, translations within the deterministic port range on either the primary or secondary server can be algorithmically reversed, provided the algorithm is known. Thus, if 198.51.100.1 port 3456 maps to 192.0.2.1 port 1000 on CGN 1 and 198.51.100.1 port 1000 on Failover

CGN 2, an operator can identify the subscriber based on outside source address and port information.

Similarly, assignments made from the dynamic overflow pool need to be logged as described above, whether translations are performed on the primary or failover CGN.

4. Impact on the IPv6 Transition

The solution described in this document is applicable to Carrier Grade NAT transition technologies (e.g. NAT444, DS-Lite, and NAT64). As discussed in [RFC7021], the authors acknowledge that native IPv6 will offer subscribers a better experience than CGN. However, many CPE devices only support IPv4. Likewise, as of October 2014, only approximately 5.2% of the top 1 million websites were available using IPv6. Accordingly, Deterministic CGN should in no way be understood as making CGN a replacement for IPv6 service; however, until such time as IPv6 content and devices are widely available, Deterministic CGN will provide operators with the ability to quickly respond to public safety requests without requiring excessive infrastructure, operations, and bandwidth to support per-connection logging.

5. Privacy Considerations

The algorithm described above makes it easier for Service Providers and public safety officials to identify the IP address of a subscriber through a CGN system. This is the equivalent level of privacy users could expect when they are assigned a public IP address and their traffic is not translated. However, this algorithm could be used by other actors on the Internet to map multiple transactions to a single subscriber, particularly if ports are distributed sequentially. While still preserving traceability, subscriber privacy can be increased by using one of the other values of the Address Assignment Algorithm (A), which would require interested parties to know more about the Service Provider’s CGN configuration to be able to tie multiple connections to a particular subscriber.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

The security considerations applicable to NAT operation for various protocols as documented in, for example, RFC 4787 [RFC4787] and RFC 5382 [RFC5382] also apply to this document.

Note that with the possible exception of cryptographically-based port allocations, attackers could reverse-engineer algorithmically-derived port allocations to either target a specific subscriber or to spoof traffic to make it appear to have been generated by a specific subscriber. However, this is exactly the same level of security that the subscriber would experience in the absence of CGN. CGN is not intended to provide additional security by obscurity.

8. Acknowledgements

The authors would like to thank the following people for their suggestions and feedback: Bobby Flaim, Lee Howard, Wes George, Jean-Francois Tremblay, Mohammed Boucadair, Alain Durand, David Miles, Andy Anchev, Victor Kuarsingh, Miguel Cros Cecilia, Fred Baker, Brian Carpenter, and Reinaldo Penno.

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.

[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.

[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.

9.2. Informative References

[I-D.miles-behave-l2nat] Miles, D. and M. Townsley, "Layer2-Aware NAT", draft-miles-behave-l2nat-00 (work in progress), March 2009.

[I-D.shirasaki-nat444]
Yamagata, I., Shirasaki, Y., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444", draft-shirasaki-nat444-06 (work in progress), July 2012.

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

[I-D.tsou-behave-natx4-log-reduction]
Tsou, T., Li, W., Taylor, T., and J. Huang, "Port Management To Reduce Logging In Large-Scale NATs", draft-tsou-behave-natx4-log-reduction-04 (work in progress), July 2013.

[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.

[RFC6292] Hoffman, P., "Requirements for a Working Group Charter Tool", RFC 6292, 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.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6431] Boucadair, M., Levis, P., Bajko, G., Savolainen, T., and T. Tsou, "Huawei Port Range Configuration Options for PPP IP Control Protocol (IPCP)", RFC 6431, November 2011.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, April 2012.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

[RFC7021] Donley, C., Howard, L., Kuarsingh, V., Berg, J., and J. Doshi, "Assessing the Impact of Carrier-Grade NAT on Network Applications", RFC 7021, September 2013.

Authors' Addresses

Chris Donley
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: c.donley@cablelabs.com

Chris Grundemann
Internet Society
Denver, CO
US

Email: cgrundemann@gmail.com

Vikas Sarawat
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: v.sarawat@cablelabs.com

Karthik Sundaresan
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: k.sundaresan@cablelabs.com

Olivier Vautrin
Juniper Networks
1194 N Mathilda Avenue
Sunnyvale, CA 94089
US

Email: olivier@juniper.net

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

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

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

Abstract

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

Table of Contents

1. Introduction

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

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

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

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

1. Video/Audio streaming, e.g. Silverlight-based applications, Netflix, YouTube, Pandora 2.

2. Peer-to-peer applications, e.g. video gaming, uTorrent

3. On line gaming, e.g. Xbox

4. Large file transfers using File Transfer Protocol (FTP)

5. Session Initiation Protocol (SIP) calls via X-Lite, Skype

6. Social Networking, e.g. Facebook, Webkinz

7. Video chat, e.g. Skype

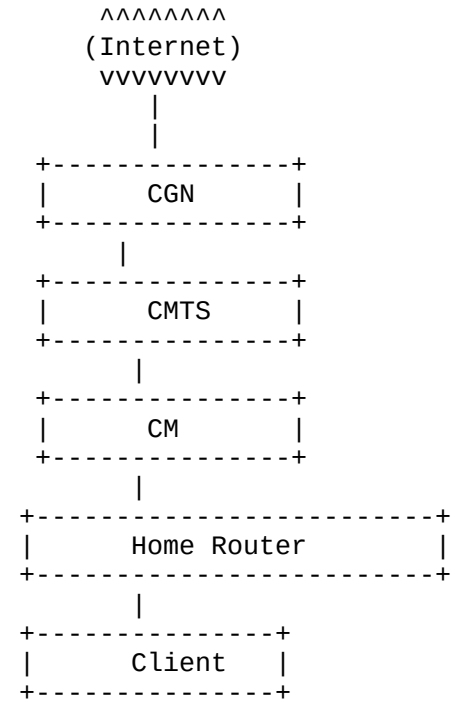
8. Web conferencing

2. Testing Scope

2.1. Test Cases

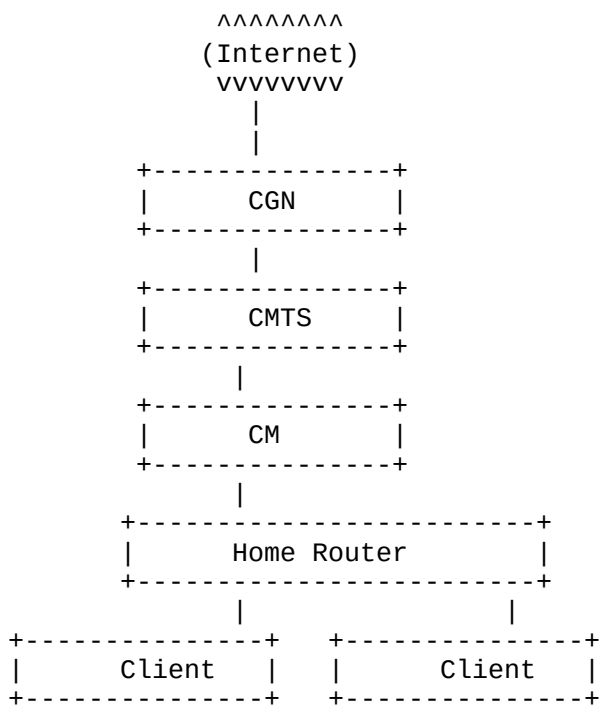
The diagrams below depict the general network architecture used for testing NAT444 and Dual Stack-Lite co-existence technologies at CableLabs.

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



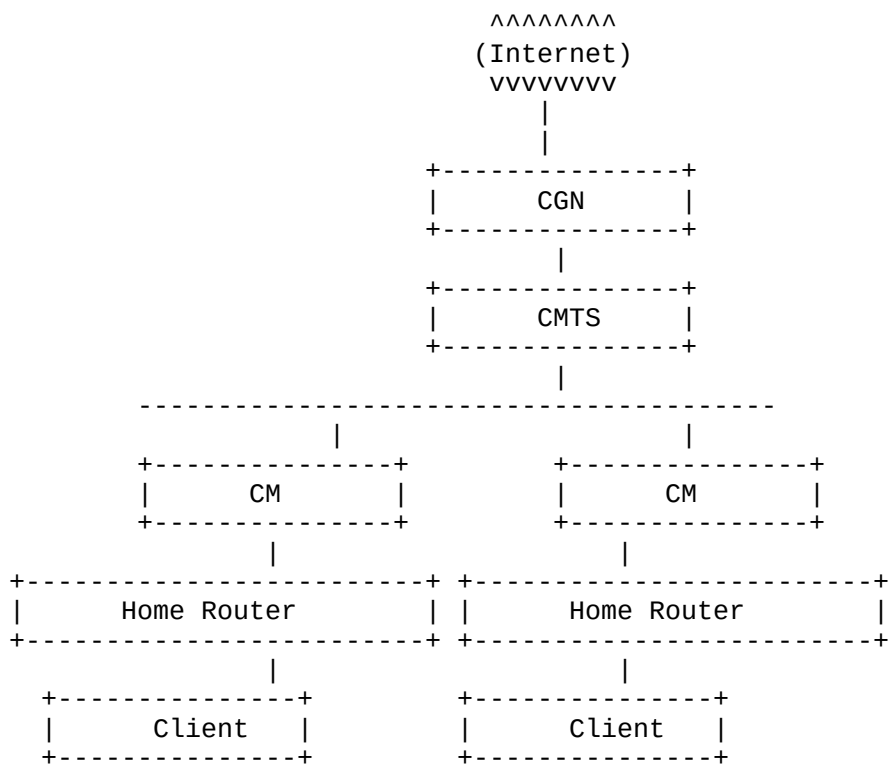
This is a typical case for a client accessing content on the Internet. For this case, we focused on basic web browsing, voice and video chat, instant messaging, video streaming (using YouTube, Google Videos , etc.), Torrent leeching and seeding, FTP, and gaming.

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



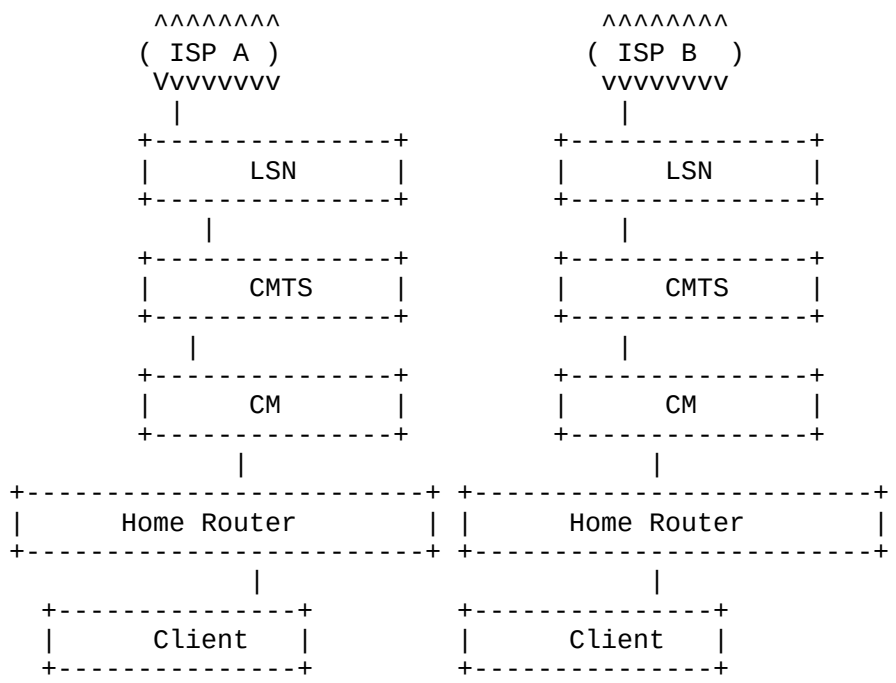
This is similar to Case 1, except that two clients are behind the same LSN and in the same home network. This test case was conducted to observe any change in speed in basic web browsing and video streaming.

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



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

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



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

2.2. General Test Environment

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

- CGN
- CMTS

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

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

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

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

11. Internet Archive - Video and Audio streaming; large file downloads

12. Video streaming using iClips

13. SIP Calls - X-Lite, Skype, PJSIP

14. MS Smooth Streaming (Silverlight)

15. Video chat - Skype, OoVoo

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

1. Windows 7, XP and Vista based laptops

2. MAC OS X laptop

3. iPad

4. Xbox gaming consoles

5. iPhone and Android smartphones

6. LG Blu-Ray player (test applications such as Netflix, Vudu, etc.)

7. Home routers - Netgear, Linksys, D-Link, Cisco, Apple

2.3. Test Metrics

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

1. Initial finding on the CGN configuration used for testing

2. Retest of the same test scenario with the CGN removed from the network

3. Retest with a new configuration (optimized) on the CGN (when possible)

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

downstream channel.

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

CGN Performance

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

2.4. Test Scenarios Executed

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

1. Single ISP, Single Home Network with Single User

2. Single ISP, Two Home Networks With One User on Each Network

3. Dual ISPs, Single Home Network with Single User on each ISP

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

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

2.5. General Test Methodologies

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

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

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

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

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

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

3. Observed CGN Impacts

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

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

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

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

3.1. Dropped Services

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

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

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

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

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

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

3.2. Performance Impacted Services

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

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

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

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

3.3. Improvements since 2010

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

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

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

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

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

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

3.4. Additional CGN Challenges

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

- Loss of geolocation information - Often, translation zones will cross traditional geographic boundaries. Since the source addresses of packets traversing an LSN are set to the external address of the LSN, it is difficult for external entities to associate IP/Port information to specific locations/areas.

- Lawful Intercept/Abuse Response - Due to the nature of NAT444 address sharing, it will be hard to determine the customer/ endpoint responsible for initiating a specific IPv4 flow based on source IP address alone. Content providers, service providers, and law enforcement agencies will need to use new mechanisms (e.g., logging source port and timestamp in addition to source IP address) to potentially mitigate this new problem. This may impact the timely response to various identification requests. See [RFC6269].

- Antispoofing - Multiplexing users behind a single IP address can lead to situations where traffic from that address triggers antispoofing/DDoS protection mechanisms, resulting in unintentional loss of connectivity for some users. We have received reports of such antispoofing/DDoS mechanisms affecting email and web services in some instances, but did not experience them in our environment.

4. 2011 Summary of Results

4.1. NAT444

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

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

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

NAT-444

4.2. DS-Lite

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

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

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

DSLite

5. 2010 Summary of Results

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

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

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

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

Case1

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

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

Case2

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

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

Case3

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

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

Case4

6. CGN Mitigation

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

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

CGN mitigation

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

7. IANA Considerations

This document has no IANA considerations.

8. Security Considerations

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

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

9. Informative References

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

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

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

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

[I-D.tsou-stateless-nat44]
Tsou, T., Liu, W., Perreault, S., Penno, R., and M. Chen, "Stateless IPv4 Network Address Translation",

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

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4689] Poretsky, S., Perser, J., Erramilli, S., and S. Khurana, "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, October 2006.

[RFC6264] Jiang, S., Guo, D., and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", RFC 6264, June 2011.

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

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

Appendix A. Acknowledgements

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

Paul Eldridge

Abishek Chandrasekaran

Vivek Ganti

Joey Padden

Lane Johnson

Authors' Addresses

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

Email: c.donley@cablelabs.com

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

Email: william.howard@twcable.com

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

Email: victor.kuarsingh@rci.rogers.com

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

Email: j.berg@cablelabs.com

Jinesh Doshi
University of Colorado

Email: jinesh.doshi@colorado.edu

Internet Engineering Task Force
Internet-Draft
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.

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:

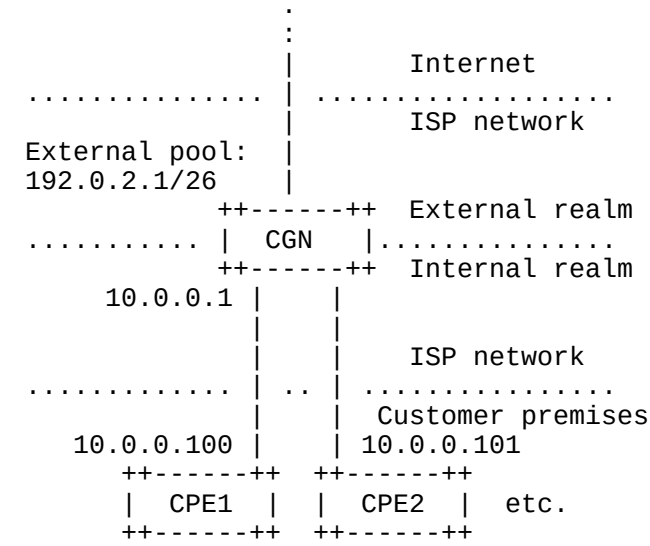
Carrier-Grade NAT (CGN): A NAT-based [RFC2663] logical function used to share the same IPv4 address among several subscribers. A CGN is not managed by the subscribers.

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.

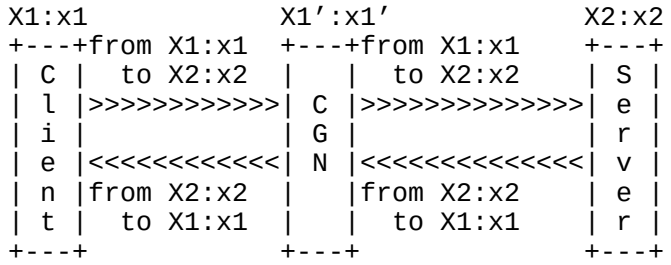


Figure 2: CGN pass-through

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
Intended status: Standards Track
Expires: October 06, 2013

T. Savolainen
Nokia
J. Korhonen
Nokia Siemens Networks
D. Wing
Cisco Systems
April 04, 2013

Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis
draft-ietf-behave-nat64-discovery-heuristic-17.txt

Abstract

This document describes a method for detecting the presence of DNS64 and for learning the IPv6 prefix used for protocol translation on an access network. The method depends on the existence of a well-known IPv4-only fully qualified domain name "ipv4only.arpa". The information learned enables nodes to perform local IPv6 address synthesis and to potentially avoid NAT64 on dual-stack and multi-interface deployments.

Table of Contents

1. Introduction

As part of the transition to IPv6, NAT64 [RFC6146] and DNS64 [RFC6147] technologies will be utilized by some access networks to provide IPv4 connectivity for IPv6-only nodes [RFC6144]. DNS64 utilizes IPv6 address synthesis to create local IPv6 addresses for peers having only IPv4 addresses, hence allowing DNS-using IPv6-only nodes 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 addresses that would be translated to the peers' IPv4 addresses.

Additionally, DNS64 is not able to do IPv6 address synthesis for nodes running validating DNSSEC-enabled DNS resolvers, but instead the synthesis must be done by the nodes themselves. In order to perform IPv6 synthesis, nodes have to learn the IPv6 prefix(es) used on the access network for protocol translation. A prefix, which may be a Network Specific Prefix (NSP) or Well-Known Prefix (WKP) [RFC6052], is referred to in this document as Pref64::/n [RFC6146].

This document describes a best-effort method for applications and nodes to learn the information required to perform local IPv6 address synthesis. The IPv6 address synthesis procedure itself is out-of-scope of this document. An example application is a browser encountering IPv4 address literals in an IPv6-only access network. Another example is a node running a validating security-aware DNS resolver in an IPv6-only access network.

The knowledge of IPv6 address synthesis taking place may also be useful if DNS64 and NAT64 are used in dual-stack enabled access networks or if a node is multi-interfaced [RFC6418]. In such cases, nodes may choose to prefer IPv4 or an alternative network interface in order to avoid traversal through protocol translators.

It is important to note that use of this approach will not result in a system that is as robust, secure, and well-behaved as an all-IPv6 system would be. Hence it is highly recommended to upgrade nodes' destinations to IPv6 and utilize the described method only as a transition solution.

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

NAT64 FQDN: a fully qualified domain name (FQDN) for a NAT64 protocol translator.

Pref64::/n: a IPv6 prefix used for IPv6 address synthesis [RFC6146].

Pref64::WKA: an IPv6 address consisting of Pref64::/n and WKA at any of the locations allowed by RFC 6052 [RFC6052].

Secure Channel: a communication channel a node has between itself and a DNS64 server protecting DNS protocol related messages from interception and tampering. The channel can be, for example, IPsec-based virtual private network (VPN) tunnel or a link layer utilizing data encryption technologies.

Well-Known IPv4-only Name (WKN): the fully qualified domain name, "ipv4only.arpa", well-known to have only A record(s).

Well-Known IPv4 Address (WKA): an IPv4 address that is well-known and present in an A record for the well-known name. Two well-known IPv4 addresses are defined for Pref64::/n discovery purposes: 192.0.0.170 and 192.0.0.171.

3. Node Behavior

A node requiring information about the presence (or absence) of NAT64, and one or more Pref64::/n used for protocol translation SHALL send a DNS query for AAAA resource records of the Well-Known IPv4-only Name (WKN) "ipv4only.arpa". The node MAY perform the DNS query in both IPv6-only and dual-stack access networks.

When sending a DNS AAAA resource record query for the WKN, a node MUST set the "Checking Disabled (CD)" bit to zero [RFC4035], as otherwise the DNS64 server will not perform IPv6 address synthesis (Section 3 of [RFC6147]) and hence would not reveal the Pref64::/n used for protocol translation.

A DNS reply with one or more AAAA resource records indicates that the access network is utilizing IPv6 address synthesis. In some scenarios captive portals, or NXDOMAIN and NODATA hijacking, performed by the access network may result in a false positive. One method to detect such hijacking is to query a fully qualified domain name that is known to be invalid (and normally return an empty response or an error response) and see if it returns a valid resource record. However, as long as the hijacked domain does not result in AAAA resource record responses that contain well-known IPv4 address in any location defined by RFC6052, the response will not disturb the Pref64::/n learning procedure.

A node MUST look through all of the received AAAA resource records to collect one or more Pref64::/n. The Pref64::/n list might include the Well-Known Prefix 64:ff9b::/96 [RFC6052] or one or more Network-Specific Prefixes. In the case of NSPs, the node SHALL determine the used address format by searching the received IPv6 addresses for the

WKN's well-known IPv4 addresses. The node SHALL assume the well-known IPv4 addresses might be found at the locations specified by [RFC6052] section 2.2. The node MUST check on octet boundaries to ensure a 32-bit well-known IPv4 address value is present only once in an IPv6 address. In case another instance of the value is found inside the IPv6 address, the node SHALL repeat the search with the other well-known IPv4 address.

If only one Pref64::/n was present in the DNS response: a node SHALL use that Pref64::/n for both local synthesis and for detecting synthesis done by the DNS64 server on the network.

If more than one Pref64::/n was present in the DNS response: a node SHOULD use all of them when determining whether other received IPv6 addresses are synthetic. The node MUST use all learned Pref64::/n when performing local IPv6 address synthesis, and use the prefixes in the order received from the DNS64 server. That is, when the node is providing a list of locally synthesized IPv6 addresses to upper layers, IPv6 addresses MUST be synthesized by using all discovered Pref64::/n in the received order.

If the well-known IPv4 addresses are not found within the standard locations, it indicates that the network is not using a standard address format or unexpected IPv4 addresses were used in the AAAA resource record synthesis. In either case, the Pref64::/n cannot be determined and the heuristic procedure has failed. Developers can over time learn of IPv6 translated address formats that are extensions or alternatives to the standard formats. Developers MAY at that point add additional steps to the described discovery procedure. The additional steps are outside the scope of the present document.

In case a node does not receive a positive DNS reply to the AAAA resource record query, the node MAY perform a DNS A resource record query for the well-known name. If the node receives a positive reply to the DNS A resource record query it means the used recursive DNS server is not a DNS64 server.

In the case of a negative response (NXDOMAIN, NODATA) or a DNS query timeout: a DNS64 server is not available on the access network, the access network filtered out the well-known query, or something went wrong in the DNS resolution. All unsuccessful cases result in a node being unable to perform local IPv6 address synthesis. In the case of timeout, the node SHOULD retransmit the DNS query like any other DNS query the node makes [RFC1035]. In the case of a negative response (NXDOMAIN, NODATA), the node MUST obey the Time-To-Live [RFC1035] of the response before resending the AAAA resource record query. The node MAY monitor for DNS replies with IPv6 addresses constructed from

the WKP, in which case if any are observed the node SHOULD use the WKP as if it were learned during the query for the well-known name.

To save Internet resources if possible, a node should perform Pref64::/n discovery only when needed (e.g., when local synthesis is required, a new network interface is connected to a new network, and so forth). The node SHALL cache the replies it receives during the Pref64::/n discovery procedure and it SHOULD repeat the discovery process ten seconds before the Time-To-Live of the Well-Known Name's synthetic AAAA resource record expires.

3.1. Validation of Discovered Pref64::/n

If a node is using an insecure channel between itself and a DNS64 server, or the DNS64 server is untrusted, it is possible for an attacker to influence the node's Pref64::/n discovery procedures. This may result in denial-of-service, redirection, man-in-the-middle, or other attacks.

To mitigate against attacks, the node SHOULD communicate with a trusted DNS64 server over a secure channel, or use DNSSEC. NAT64 operators SHOULD provide facilities for validating discovery of Pref64::/n via a secure channel and/or DNSSEC protection.

It is important to understand that DNSSEC only validates that the discovered Pref64::/n is the one that belongs to a domain used by NAT64 FQDN. Importantly, the DNSSEC validation does not tell if the node is at the network where the Pref64::/n is intended to be used. Furthermore, DNSSEC validation cannot be utilized in the case of WKP.

3.1.1. DNSSEC Requirements for the Network

If the operator has chosen to support nodes performing validation of discovered Pref64::/n with DNSSEC, the operator of the NAT64 device MUST perform the following configurations.

1. Have one or more fully qualified domain names for the NAT64 translator entities (later referred as NAT64 FQDN). In the case of more than one Pref64::/n being used in a network, e.g., for load-balancing purposes, it is for network administrators to decide whether a single NAT64's fully qualified domain name maps to more than one Pref64::/n, or whether there will be a dedicated NAT64 FQDN per Pref64::/n.

2. Each NAT64 FQDN MUST have one or more DNS AAAA resource records containing Pref64::WKA (Pref64::/n combined with WKA).

3. Each Pref64::WKA MUST have a PTR resource record that points to the corresponding NAT64 FQDN.

4. Sign the NAT64 FQDNs' AAAA and A resource records with DNSSEC.

3.1.2. DNSSEC Requirements for the Node

A node SHOULD prefer a secure channel to talk to a DNS64 server, whenever possible. In addition, a node that implements a DNSSEC validating resolver MAY use the following procedure to validate discovery of the Pref64::/n.

1. Heuristically find Pref64::/n candidates by making a AAAA resource record query for "ipv4only.arpa" by following the procedure in Section 3. This will result in IPv6 addresses consisting of Pref64::/n combined with WKA, i.e., Pref64::WKA. For each Pref64::/n that the node wishes to validate, the node performs the following steps.

2. Send a DNS PTR resource record query for the IPv6 address of the translator (for "ip6.arpa"), using the Pref64::WKA learned in step 1. CNAME and DNAME results should be followed according to the rules in RFC 1034 [RFC1034], RFC 1034 [RFC1035], and RFC 6672 [RFC6672]. The ultimate response will include one or more NAT64 FQDNs.

3. The node SHOULD compare the domains of learned NAT64 FQDNs to a list of the node's trusted domains and choose a NAT64 FQDN that matches. The means for a node to learn the trusted domains is implementation-specific. If the node has no trust for the domain, the discovery procedure is not secure and the remaining steps described below MUST NOT be performed.

4. Send a DNS AAAA resource record query for the NAT64 FQDN.

5. Verify the DNS AAAA resource record contains Pref64::WKA addresses received at the step 1. It is possible that the NAT64 FQDN has multiple AAAA records, in which case the node MUST check if any of the addresses match the ones obtained in step 1. The node MUST ignore other responses and not use them for local IPv6 address synthesis.

6. Perform DNSSEC validation of the DNS AAAA response.

After the node has successfully performed the above five steps, the node can consider Pref64::/n validated.

3.2. Connectivity Check

After learning a Pref64::/n, the node SHOULD perform a connectivity check to ensure the learned Pref64::/n is functional. It could be non-functional for a variety of reasons -- the discovery failed to work as expected, the IPv6 path to the NAT64 is down, the NAT64 is down, or the IPv4 path beyond the NAT64 is down.

There are two main approaches to determine if the learned Pref64::/n is functional. The first approach is to perform a dedicated connectivity check. The second approach is to simply attempt to use the learned Pref64::/n. Each approach has some trade-offs (e.g., additional network traffic or possible user-noticeable delay), and implementations should carefully weigh which approach is appropriate for their application and the network.

The node SHOULD use an implementation-specific connectivity check server and a protocol of the implementation's choice, but if that is not possible, a node MAY do a PTR resource record query of the Pref64::WKA to get a NAT64 FQDN. The node then does an A resource query of the NAT64 FQDN, which will return zero or more A resource records pointing to connectivity check servers used by the network operator. A negative response to the PTR or A resource query means there are no connectivity check servers available. A network operator that provides NAT64 services for a mix of nodes with and without implementation-specific connectivity check servers SHOULD assist nodes in their connectivity checks by mapping each NAT64 FQDN to one or more DNS A resource records with IPv4 address(es) pointing to connectivity check server(s). The Pref64::/n -based connectivity check approach works only with NSPs, as it is not possible to register A records for each different domain using WKP. The network operator MUST disable ICMPv6 rate limiting for connectivity check messages.

In case of multiple connectivity check servers being available for use, the node chooses the first one, preferring implementation-specific servers.

The connectivity check protocol used with implementation-specific connectivity check servers is implementation-specific.

The connectivity check protocol used with connectivity check servers pointed to by the NAT64 FQDN's A resource records is ICMPv6 [RFC4443]. The node performing a connectivity check against these servers SHALL send an ICMPv6 Echo Request to an IPv6 address synthesized by combining discovered Pref64::/n with an IPv4 address of the server as specified in [RFC6052]. This will test the IPv6 path to the NAT64, the NAT64's operation, and the IPv4 path all the

way to the connectivity check server. If no response is received for the ICMPv6 Echo Request, the node SHALL send another ICMPv6 Echo Request, a second later. If still no response is received, the node SHALL send a third ICMPv6 Echo Request two seconds later. If an ICMPv6 Echo Response is received, the node knows the IPv6 path to the connectivity check server is functioning normally. If, after the three transmissions and three seconds since the last ICMPv6 Echo Request, no response is received, the node learns this Pref64::/n might not be functioning, and the node MAY choose a different Pref64::/n (if available), choose to alert the user, or proceed anyway assuming the failure is temporary or with the connectivity check itself. After all, the ICMPv6 is by design unreliable and failure to receive ICMPv6 responses may not indicate anything other than network failure to transport ICMPv6 messages through.

If no separate connectivity check is performed before local IPv6 address synthesis, a node MAY monitor success of connection attempts performed with locally synthesized IPv6 addresses. Based on success of these connections, and based on possible ICMPv6 error messages received (such as Destination Unreachable messages), the node MAY cease to perform local address synthesis and MAY restart the Pref64::/n discovery procedures.

3.2.1. No Connectivity Checks Against ipv4only.arpa

Clients MUST NOT send a connectivity check to an address returned by the ipv4only.arpa query. This is because, by design, no server will be operated on the Internet at that address as such. Similarly, network operators MUST NOT operate a server on that address. The reason this address isn’t used for connectivity checks is that operators who neglect to operate a connectivity check server will allow that traffic towards the Internet where it will be dropped and cause a false negative connectivity check with the client (that is, the NAT64 is working fine, but the connectivity check fails because a server is not operating at "ipv4only.arpa" on the Internet and a server is not operated by the NAT64 operator). Instead, for the connectivity check, an additional DNS resource record is looked up and used for the connectivity check. This ensures that packets don’t unnecessarily leak to the Internet and reduces the chance of a false negative connectivity check.

3.3. Alternative Fully Qualified Domain Names

Some applications, operating systems, devices, or networks may find it advantageous to operate their own DNS infrastructure to perform a function similar to "ipv4only.arpa", but using a different resource record. The primary advantage is to ensure availability of the DNS infrastructure and ensure the proper configuration of the DNS record

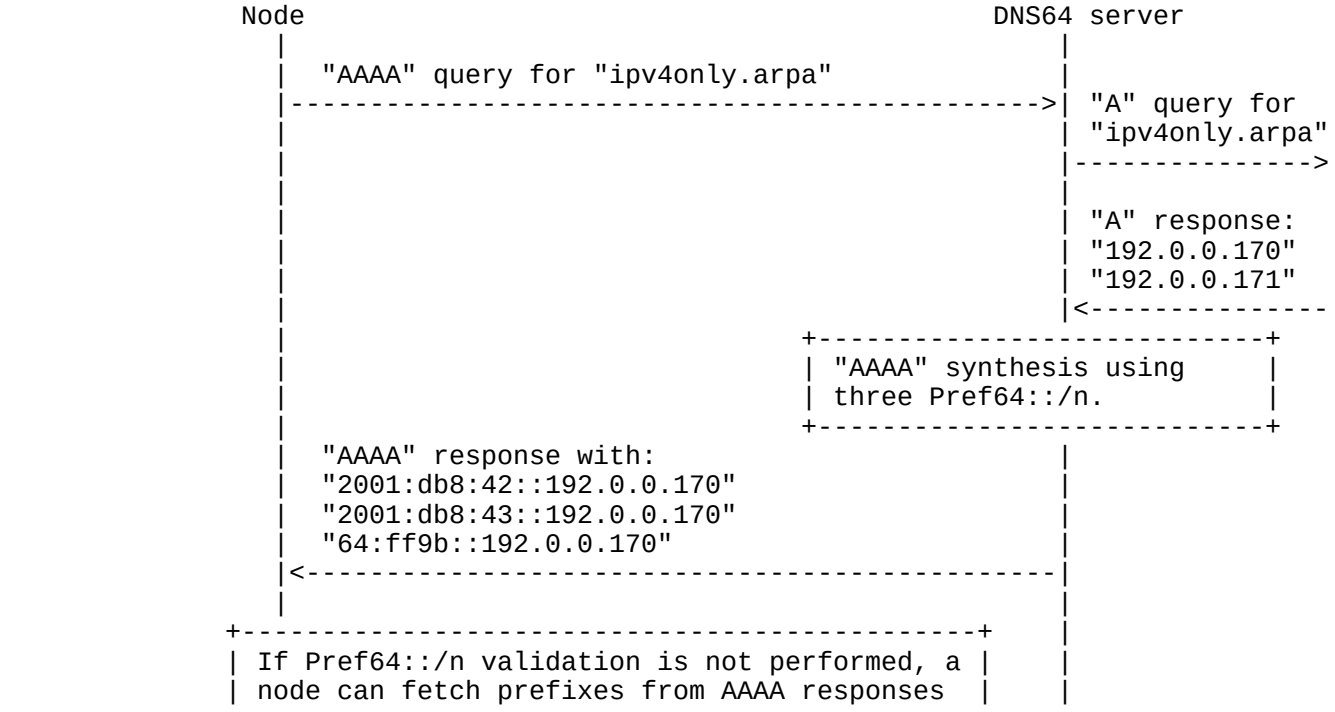
itself. For example, a company named Example might have their application query "ipv4only.example.com". Other than the different DNS resource record being queried, the rest of the operations are anticipated to be identical to the steps described in this document.

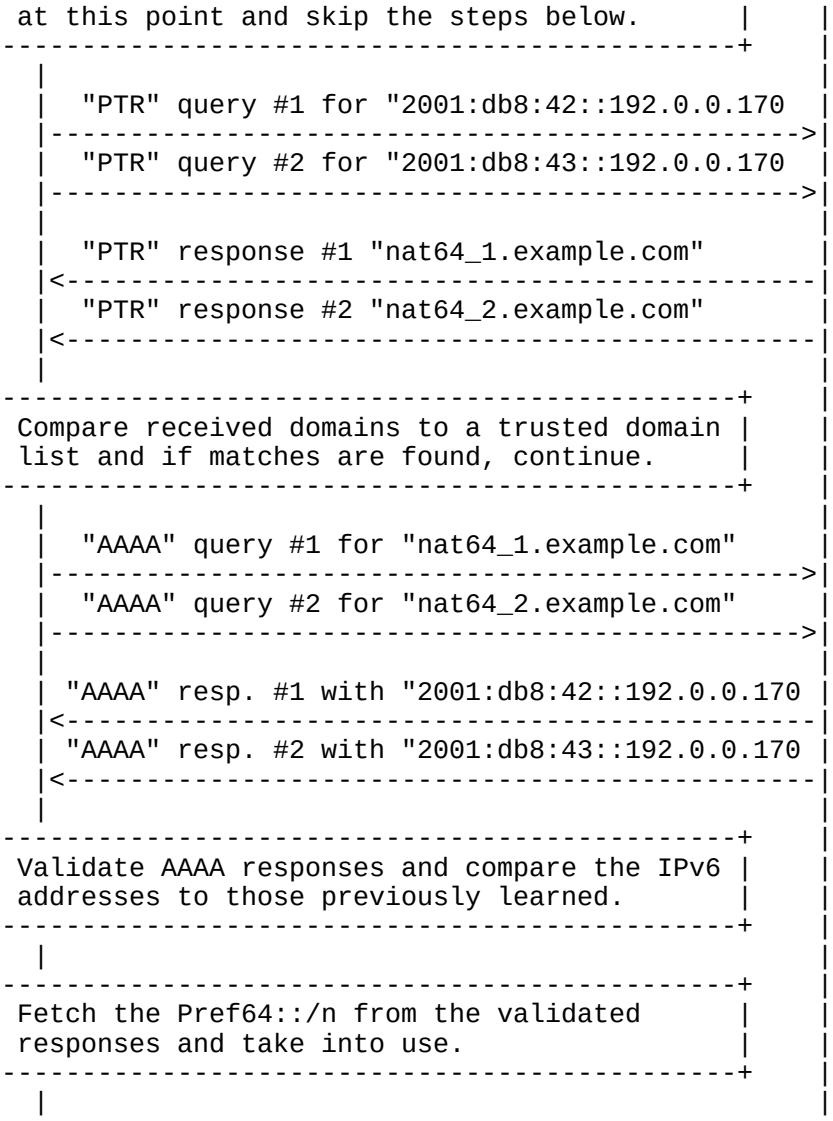
3.4. Message Flow Illustration

The figure below gives an example illustration of a message flow in the case of prefix discovery utilizing Pref64::/n validation. The figure shows also a step where the procedure ends if no Pref64::/n validation is performed.

In this example, three Pref64::/n are provided by the DNS64 server. The first Pref64::/n is using an NSP, in this example "2001:db8:42::/96". The second Pref64::/n is using an NSP, in this example "2001:db8:43::/96". The third Pref64::/n is using the WKP. Hence, when the Pref64::/n are combined with the WKA to form Pref64::WKA, the synthetic IPv6 addresses returned by DNS64 server are "2001:db8:42::192.0.0.170", "2001:db8:43::192.0.0.170", and "64:ff9b::192.0.0.170". The DNS64 server could also return synthetic addresses containing the IPv4 address 192.0.0.171.

The validation is not done for the WKP, see Section 3.1.





Pref64::/n discovery procedure

4. Operational Considerations for Hosting the IPv4-Only Well-Known Name

The authoritative name server for the well-known name SHALL have DNS record Time-To-Live (TTL) set to at least 60 minutes in order to improve effectiveness of DNS caching. The exact TTL value will be determined and tuned based on operational experiences.

The domain serving the well-known name MUST be signed with DNSSEC. See also section 7.

5. Operational Considerations for DNS64 Operator

A network operator of a DNS64 server can guide nodes utilizing heuristic discovery procedures by managing the responses a DNS64 server provides.

If the network operator would like nodes to utilize multiple Pref64::/n, the operator needs to configure DNS64 servers to respond with multiple synthetic AAAA records. As per Section 3 the nodes can then use them all.

There are no guarantees on which of the Pref64::/n nodes will end up using. If the operator wants nodes to specifically use a certain Pref64::/n or periodically change the Pref64::/n they use, for example for load balancing reasons, the only guaranteed method is to make DNS64 servers return only a single synthetic AAAA resource record, and have the Time-To-Live of that synthetic record such that the node repeats the Pref64::/n discovery when required.

Besides choosing how many Pref64::/n to respond and what Time-To-Live to use, DNS64 servers MUST NOT interfere with or perform other special procedures for the queries related to the well-known name.

5.1. Mapping of IPv4 Address Ranges to IPv6 Prefixes

RFC 6147 [RFC6147] allows DNS64 implementations to be able to map specific IPv4 address ranges to separate Pref64::/n. That allows handling of special use IPv4 addresses [RFC5735]. The example setup where this might be used is illustrated in Figure 1. The NAT64 "A" is used when accessing IPv4-only servers in the datacenter, and the NAT64 "B" is used for Internet access.

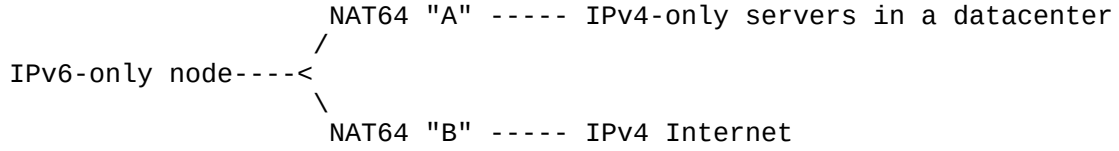


Figure 1: NAT64s with IPv4 Address Ranges

The heuristic discovery method described herein does not support learning of the possible rules used by a DNS64 server for mapping specific IPv4 address ranges to separate Pref64::/n. Therefore, nodes will use the same discovered Pref64::/n to synthesize IPv6 addresses from any IPv4 address. This can cause issues for routing

and connectivity establishment procedures. The operator of the NAT64 and the DNS64 ought to take this into account in the network design.

The network operators can help IPv6-only nodes by ensuring the nodes do not have to work with IPv4 address literals for which special mapping rules are used. That is, the IPv4-only servers addressed from the special IPv4 address ranges ought to have signed AAAA records, which allows IPv6-only nodes to avoid local address synthesis. If the IPv6-only nodes are not using DNSSEC, then it is enough if the network's DNS64 server returns synthetic AAAA resource records pointing to IPv4-only servers. Avoiding the need for IPv6-only nodes to perform address synthesis for IPv4 addresses belonging to special ranges is the best approach to assist nodes.

If the IPv6-only nodes have no other choice than using IPv4-address literals belonging to special IPv4 address ranges, and the IPv6-only node will perform local synthesis by using the discovered Pref64::/n, then the network ought to ensure with routing that the packets are delivered to the correct NAT64. For example, a router in the path from an IPv6-only host to NAT64s can forward the IPv6 packets to the correct NAT64 as illustrated in Figure 2. The routing could be based on the last 32-bits of the IPv6 address, but the network operator can also use some other IPv6 address format allowed by RFC 6052 [RFC6052], if it simplifies routing setup. This setup requires additional logic on the NAT64 providing connectivity to special IPv4 address ranges: it needs to be able to translate packets it receives that are using the Pref64::/n used with Internet connections.

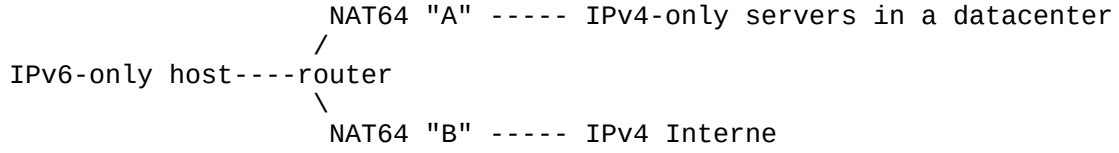


Figure 2: NAT64s with Assisting Router

6. Exit Strategy

A day will come when this tool is no longer needed. A node SHOULD implement a configuration knob for disabling the Pref64::/n discovery feature.

7. Security Considerations

The security considerations follow closely those of RFC 6147 [RFC6147]. The possible attacks are very similar in the case where an attacker controls a DNS64 server and returns tampered IPv6 addresses to a node and in the case where an attacker causes the node

to use tampered Pref64::/n for local address synthesis. DNSSEC cannot be used to validate responses created by a DNS64 server the node has no trust relationship with. Hence this document does not change the big picture for untrusted network scenarios. If an attacker alters the Pref64::/n used by a DNS64 server or a node, the traffic generated by the node will be delivered to an altered destination. This can result in either a denial-of-service (DoS) attack (if the resulting IPv6 addresses are not assigned to any device), a flooding attack (if the resulting IPv6 addresses are assigned to devices that do not wish to receive the traffic), or an eavesdropping attack (in case the altered NSP is routed through the attacker).

Even though well-known name's DNS A resource record would not necessarily need to be protected with DNSSEC, as both the name and IPv4 addresses well-known, for the DNS AAAA resource record queries DNSSEC protection is required. Without DNSSEC, fake positive AAAA responses could cause hosts to erroneously detect Pref64::/n, thus allowing attacker to inject malicious Pref64::/n for hosts' synthesis procedures. A signed ipv4only.arpa allows validating DNS64 servers (see [RFC6147] Section 3 case 5, for example) to detect malicious AAAA resource records. Therefore, the zone serving the well-known name has to be protected with DNSSEC.

For Pref64::/n discovery validation, the access network SHOULD sign the NAT64 translator's fully qualified domain name. A node SHOULD use the algorithm described in Section 3.1 to validate each discovered Pref64::/n.

The procedure of Section 3.1.2 requires a node using DNSSEC to validate discovery of Pref64::/n to have a list of trusted domains. In the case of not having matching domain at the step 3 of Section 3.1.2 an implementation might be tempted to ask for a user to add a received domain as trusted - temporarily or permanently. The history has shown that average users are unable to properly handle such queries, and tend to answer positively without thinking in an attempt to get quickly forward. Therefore, unless the DNSSEC-using implementation has a way to dynamically and reliably add trusted domains, it is better to fail the Pref64::/n discovery procedure.

Lastly, the best mitigation action against Pref64::/n discovery attacks is to add IPv6 support for nodes' destinations and hence reduce the need to perform local IPv6 address synthesis.

8. IANA Considerations

8.1. Domain Name Reservation Considerations

According to procedures described in [RFC3172] and [I-D.cheshire-dnsext-special-names] this document requests IANA to delegate a new second level domain in the .ARPA zone for the well-known domain name "ipv4only.arpa". The intention is that there will not be any further delegation of names below the ipv4only.arpa domain. The administrative and operational management of this zone is to be undertaken by IANA. The answers to seven questions of [I-D.cheshire-dnsext-special-names] are as follows:

1. No, although this is a domain delegated under the .arpa infrastructural identifier top level domain."

2. Yes. Any application attempting to perform NAT64 discovery will query the name.

3. Yes, to the extent the API or library is affected by NAT64.

4. No.

5. No.

6. This name has effects for operators of NAT64/DNS64, but otherwise is just another delegated .arpa domain.

7. The registry for .arpa is held at IANA, and only IANA needs to take action here.

8.2. IPv4 Address Allocation Considerations

The well-known name needs to map to two different global IPv4 addresses, which are to be allocated as described in [RFC5736] and [I-D.bonica-special-purpose]. The addresses are to be taken from the IANA IPv4 Special Purpose Address Registry [RFC5736], and 192.0.0.170 and 192.0.0.171 are to be assigned to this document with parameters shown below:

Attribute	Value
Address Block	192.0.0.170/32
	192.0.0.171/32
Name	NAT64/DNS64 Discovery
RFC	RFC TBD Section 2.2.
Allocation Date	February 2013
Termination Date	N/A
Source	False
Destination	False
Forwardable	False

| Global | False |
| Reserved-by-protocol | True |

The Record for IPv4 address allocation for IPv4 Special Purpose Address Registry

The following two records should be added to .arpa to the name servers run by ICANN/IANA:

ipv4only IN A 192.0.0.170

ipv4only IN A 192.0.0.171

8.3. IAB Statement Regarding This .arpa Request

With the publication of this document, the IAB approves of the delegation of "ipv4only" in the .arpa domain. Under [RFC3172], the IAB is requesting IANA to delegate and provision ipv4only.arpa as written in this specification. However, the IAB does not take any architectural or technical position about this specification.

9. Acknowledgements

Authors would like to thank Dmitry Anipko, Cameron Byrne, Aaron Yi Ding Christian Huitema, Washam Fan, Peter Koch, Stephan Lagerholm, Zhenqiang Li, Simon Perreault, Marc Petit-Huguenin, Andrew Sullivan, and Dave Thaler, for significant improvement ideas and comments.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

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

[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.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

10.2. Informative References

[I-D.bonica-special-purpose] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", draft-bonica-special-purpose-07 (work in progress), January 2013.

[I-D.cheshire-dnsext-special-names] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", draft-cheshire-dnsext-special-names-03 (work in progress), September 2012.

[RFC3172] Huston, G., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, September 2001.

[RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", BCP 153, RFC 5735, January 2010.

[RFC5736] Huston, G., Cotton, M., and L. Vegoda, "IANA IPv4 Special Purpose Address Registry", RFC 5736, January 2010.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.

Appendix A. Example of DNS Record Configuration

The following BIND-style examples illustrate how A and AAAA records could be configured by a NAT64 operator.

The examples use Pref64::/n of 2001:db8::/96, both WKAs, and the example.com domain.

The PTR record for reverse queries (Section 3.1.1 bullet 3):

```
$ORIGIN A.A.0.0.0.0.0.C\
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 12h 15m 3w 2h)
          IN      NS    ns.example.com.

          IN      PTR   nat64.example.com.

$ORIGIN B.A.0.0.0.0.0.C\
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 12h 15m 3w 2h)
          IN      NS    ns.example.com.

          IN      PTR   nat64.example.com.
```

If example.com does not use DNSSEC, the following configuration file could be used. Please note that nat64.example.com has both an AAAA record with the Pref64::/n and an A record for the connectivity check (Section 3.1.1 bullet 2).

```
example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
                      2002050501 ; serial
                      100        ; refresh (1 minute 40 seconds)
                      200        ; retry (3 minutes 20 seconds)
                      604800     ; expire (1 week)
                      100        ; minimum (1 minute 40 seconds)
                      )

example.com.  IN NS  ns.example.com.

nat64.example.com.
              IN AAAA  2001:db8:0:0:0:0:C000:00AA
              IN AAAA  2001:db8:0:0:0:0:C000:00AB
              IN A  192.0.2.1
```

To DNSSEC sign the records, the owner of the example.com zone would have RRSIG records for both the AAAA and A records for nat64.example.com. As a normal DNSSEC requirement, the zone and its parent also need to be signed.

Appendix B. About the IPv4 Address for the Well-Known Name

The IPv4 addresses for the well-known name cannot be non-global IPv4 addresses as listed in the Section 3 of [RFC5735]. Otherwise DNS64 servers might not perform AAAA record synthesis when the well-known prefix is used, as stated in Section 3.1 of [RFC6052]. However, the addresses do not have to be routable or allocated to any real node, as no communications will be initiated to these IPv4 address.

Allocation of at least two IPv4 addresses improves the heuristics in cases where the bit pattern of the primary IPv4 address appears more than once in the synthetic IPv6 address (i.e., the NSP prefix contains the same bit pattern as the IPv4 address).

If no well-known IPv4 addresses would be statically allocated for this method, the heuristic would require sending of an additional A query to learn the IPv4 addresses that would be then searched from inside of the received IPv6 address.

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

Dan Wing
Cisco Systems
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

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.

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".

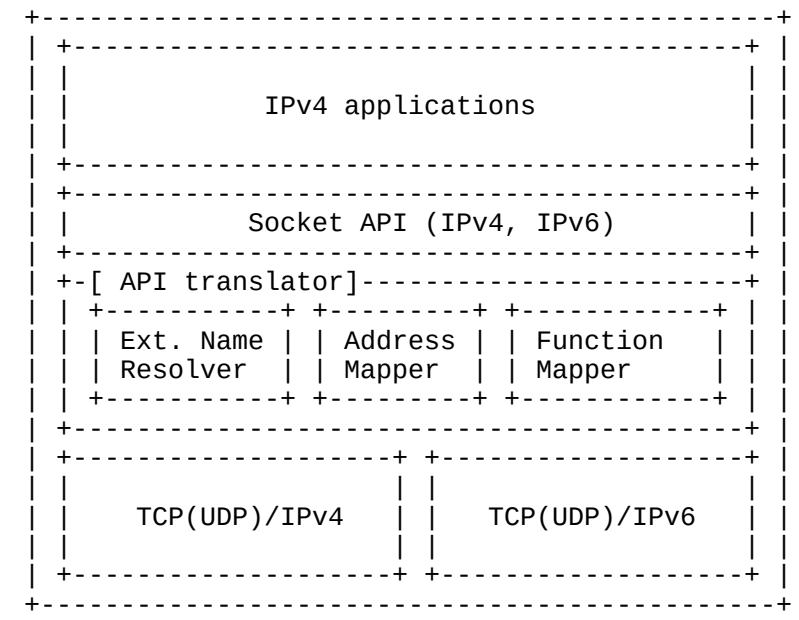


Figure 1: Architecture of a dual stack host using protocol translation at socket layer

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".

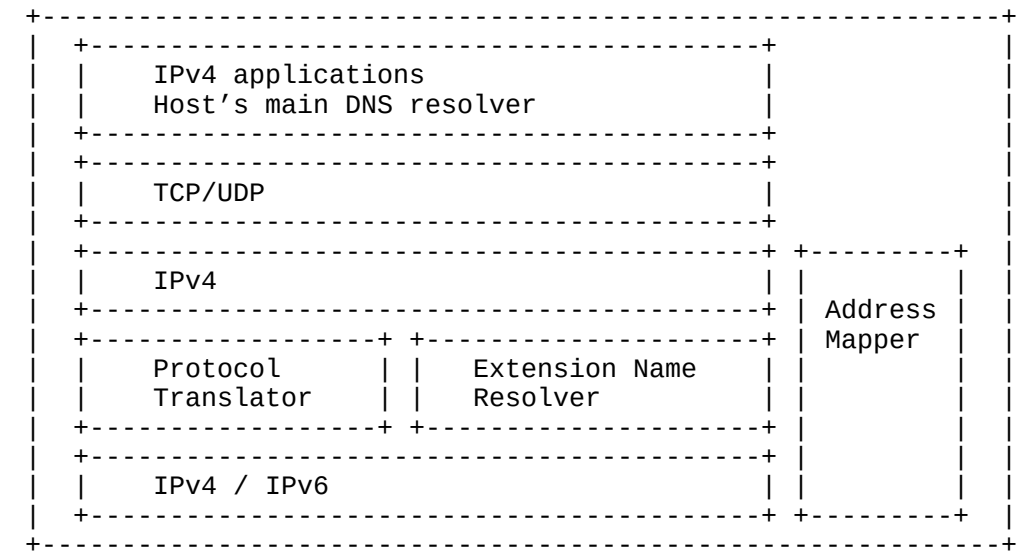


Figure 2: Architecture of a dual-stack host using protocol translation at the network layer

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.

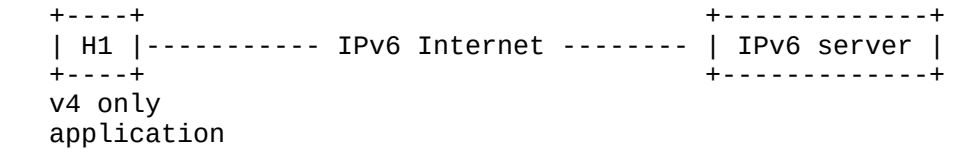


Figure 4: Network Scenario #1

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.

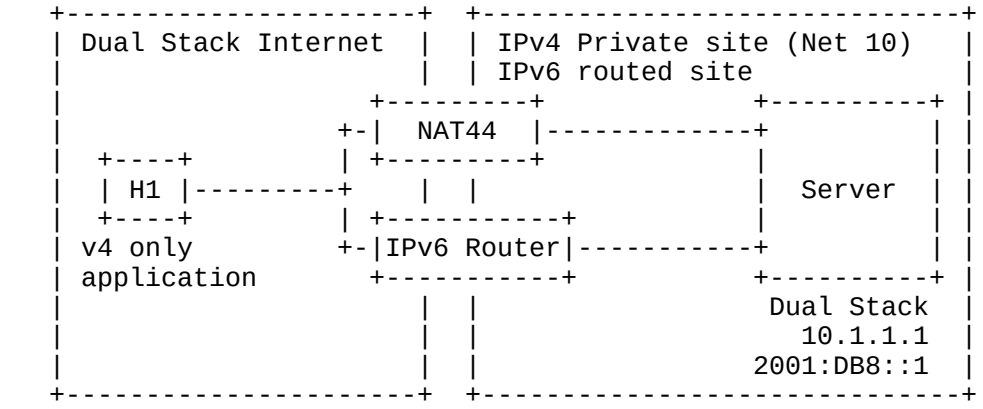
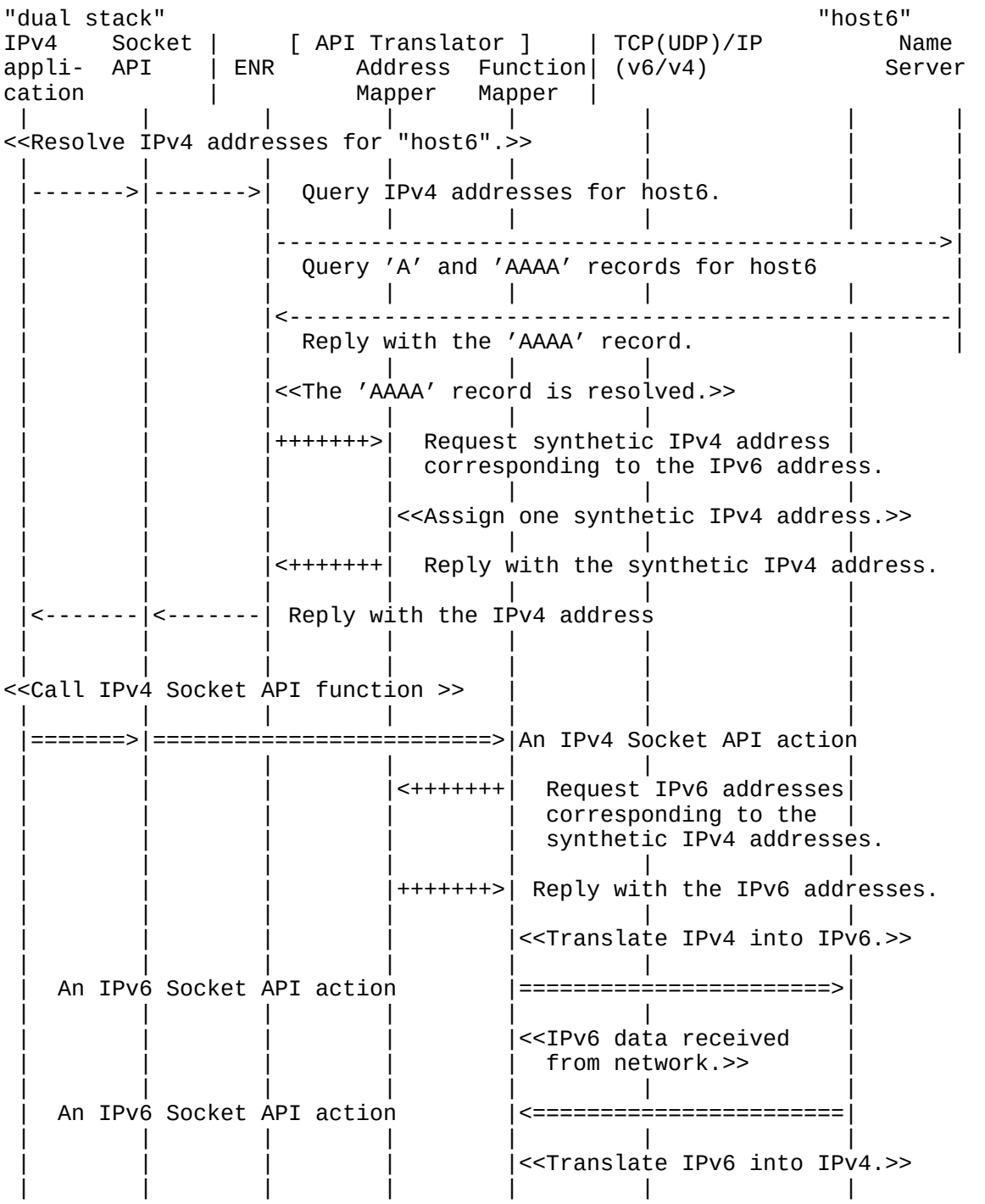


Figure 5: Network Scenario #2

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.>>

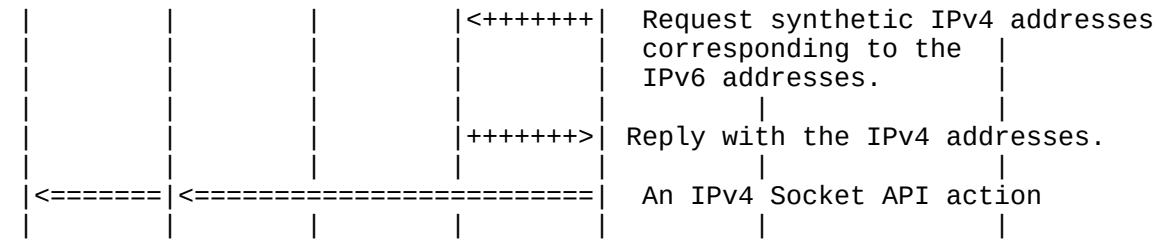
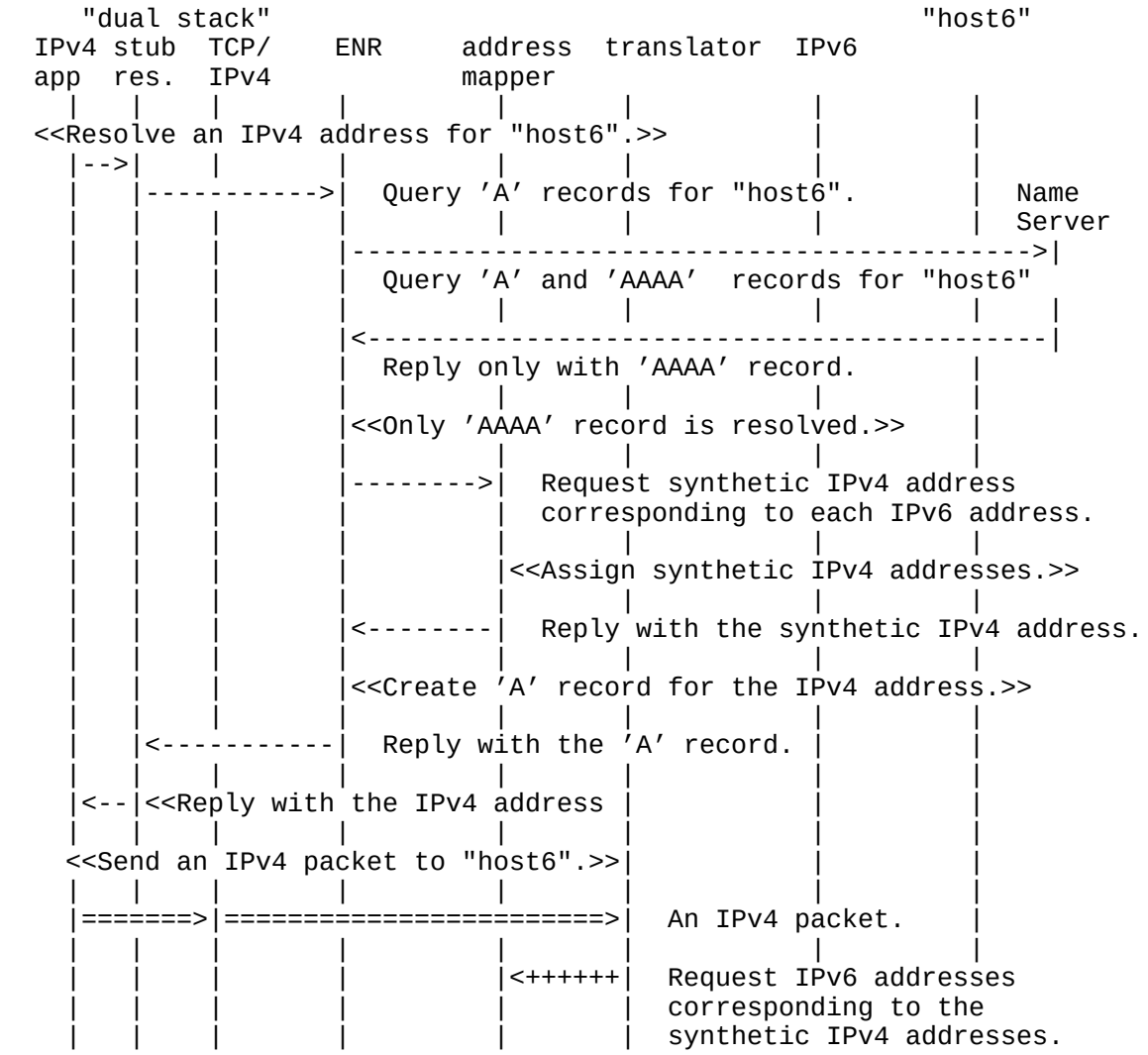


Figure 6: Example of BIH as API addition



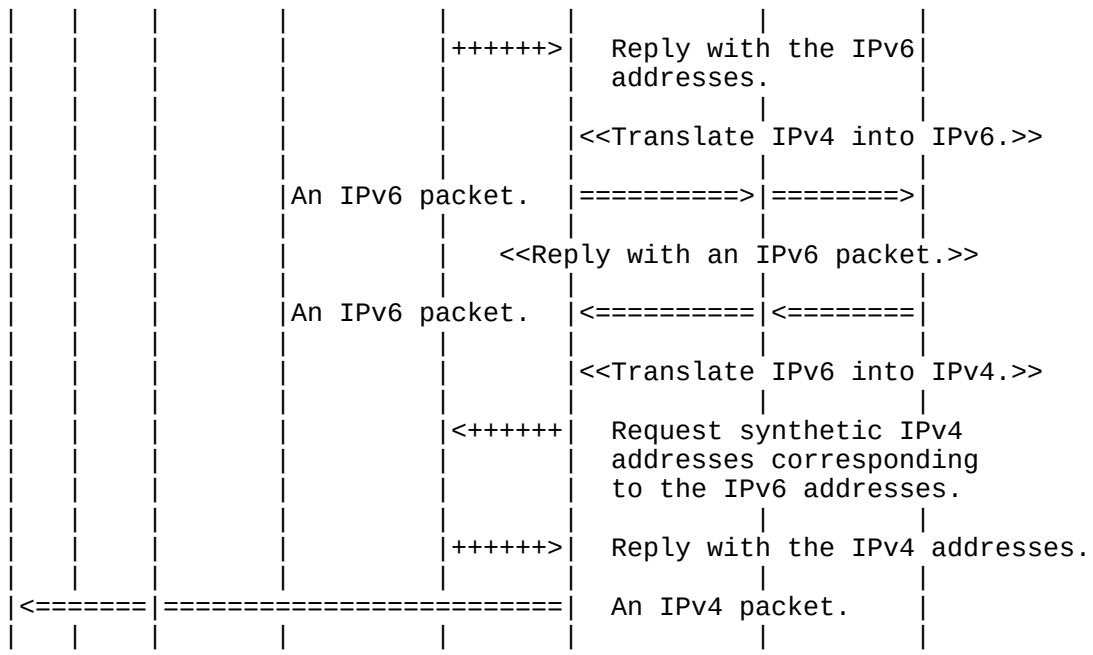


Figure 7: Example of BIH at the network layer

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.

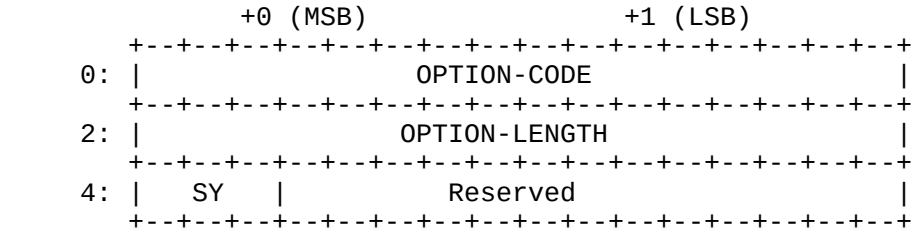
2. Requirements and Terminology

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

OPSAWG
Internet-Draft
Intended status: Informational
Expires: August 23, 2012

V. Kuarsingh, Ed.
J. Cianfarani
Rogers Communications

February 20, 2012

CGN Deployment with MPLS/VPNs
draft-kuarsingh-lsn-deployment-06

Abstract

This document specifies a framework to integrate a Network Address Translation layer into an operator's network to function as a Carrier Grade NAT (also known as CGN or Large Scale NAT). CGN is a concept also described in [I-D.ietf-behave-lsn-requirements] and describes the model as a dual layer translation model. Although operators may wish to deploy IPv6 to strategically overcome IPv4 exhaustion, near term needs may not be satisfied with an IPv6 deployment alone. This document provides a practical integration model which allows CGN to be integrated into the network meeting the connectivity needs of the customer while being mindful of not disrupting existing services and meeting the technical challenges that CGN brings. The model includes the use of MPLS/VPNs defined in [RFC4364] as a tool to achieve this goal. This document does not intend to defend the merits of CGN.

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 23, 2012.

Copyright Notice

Table of Contents

1. Introduction

Operators are faced with near term IPv4 address exhaustion challenges. Many operators may not have a sufficient amount of IPv4 addresses in the future to satisfy the needs of their growing customer base. This challenge may also be present before or during an active transition to IPv6 somewhat complicating the overall problem space.

To face this challenge, operators may need to deploy CGN (Carrier Grade NAT) as described in [I-D.ietf-behave-lsn-requirements] to help extend the connectivity matrix once IPv4 addresses run out in the network. CGN's addition to the network requires integration in an often running state environment with working IPv4 and/or IPv6 services.

The addition of the CGN introduces an operator controlled and administered translation layer which needs to be added in a manner which does not overly disrupt existing services. This addition may also include interworking in a dual stack environment where the IPv4 path requires translation.

This document shows how MPLS/VPNs as described in [RFC4364] can be used to integrate the CGN infrastructure solving key problems faced by the operator. This model has also been tested and validated in real production network models and allows fluid operation with existing IPv4 and IPv6 services.

2. Motivation

The selection of CGN may be made by an operator based on a number of factors. The overall driver may be the depletion of IPv4 address pools which leaves little to no addresses for IPv4 service growth. IPv6 is considered the strategic answer, but it's applicability and usefulness in many networks is limited by the current access network and consumer home network. These environments often are filled with IPv4-Only equipment which may not be upgradable to IPv6.

The ability to replace IPv4-Only equipment may be out of the control of the operator, and even when it's in the administrative control; it poses both cost and technical challenges as operators build out massive programs for equipment retirement or upgrade. Theses issues leave an operator in a precarious position which may lead to the decision to deploy CGN. Other address IPv4 sharing options do exist which are more architecturally desirable, but the practical and workable approach in many cases is a CGN deployment using NAT444.

If the operator as has chosen to deploy CGN, they should this in a manner as not to negatively impact the existing IPv4 or IPv6 customer base. This will include solving a number of challenges since customers who's connections require translation will have network routing and flow needs which are different from legacy IPv4 connections.

The solution will also need to work in a dual stack environment where other options such as DS-Lite [RFC6333] are not yet viable. Even technologies like 6RD [RFC5969] still require an IPv4 connectivity path to service the customer endpoint. The solution will need to address basic Internet connectivity, on-net service offerings, back office management, billing, policy and security models already in place within the operator's network. CGN will often integrate quite readily with the aforementioned requirements where as other transition mechanism may not due to the requirements to support IPv6 as the base protocol for IPv4 connectivity.

3. CGN Network Deployment Requirements

If a service provider is considering a CGN deployment with a provider NAT44 function, there are a number of basic requirements which are of importance. Preliminary requirements may require the following from the incoming CGN system architecture:

- Support distributed (sparse) and centralized (dense) deployment models;

- Allow co-existence with traditional IPv4 based deployments, which provide global scoped IPs to CPEs;

- Provide a framework for CGN by-pass supporting non-translated flows between endpoints within a provider's network;

- Provide routing framework which allows the segmentation of routing control and forwarding paths between CGN and non-CGN mediated flows;

- Provide flexibility for operators to modify their deployments over time as translation demands change (connections, bandwidth, translation realms/zones and other vectors);

- Flexibility should include integration options for common access technologies such as DSL (BRAS), DOCSIS (CMTS), Mobile (GGSN/PGW/ ASN-GW), and Ethernet access;

- Support deployment modes that allow for IPv4 address overlap within the operator's network (between various translation realms or zones);

- Allow for evolution to future dual-stack and IPv4/IPv6 transition deployment modes;

- Transactional logging and export capabilities to support auxiliary functions including abuse mitigation;

- Support for stateful connection synchronization between translation instances/elements (redundancy);

- Support for CGN Shared Space [I-D.weil-shared-transition-space-request] deployment modes if applicable;

- Allows for the enablement of CGN functionality (if required) while still minimizing costs and customer impact to the best extend possible;

Other requirements may be assessed on a operator-by-operator basis, but those listed above should be considered for any given deployment architecture.

3.1. Centralized versus Distributed Deployment

Centralized deployments of CGN (longer proximity to end user and/or higher densities of subscribers/connections to CGN instances) differ from distributed deployments of CGN (closer proximity to end user and/or lower densities of subscribers/connections to CGN instances). Service providers will likely deploy CGN translation points more centrally during initial phases. Early deployments will likely see light loading on these new systems since legacy IPv4 services will continue to operate with most endpoints using globally unique IPv4 addresses. Exceptional cases which may drive heavy usage in initial stages may include operators who already translate most IPv4 traffic and will migrate to a CGN implementation from legacy firewalls; or a green field deployment which may see quick growth in the number of new IPv4 endpoints which require Internet connectivity.

Over time, most providers will likely need to expand and possibly distribute the translation points as demand for the CGN system increases. The extent of the expansion of the CGN infrastructure will depend on factors such as growth in the number of IPv4 endpoints, status of IPv6 content on the Internet and the overall progress globally to an IPv6-dominate Internet (reducing the demand for IPv4 connectivity).

3.2. CGN and Traditional IPv4 Service Co-existence

Newer CGN serviced endpoints will exist alongside endpoints served by traditional IPv4 global IPs. Providers will need to rationalize these environments since both have distinct forwarding needs. Traditional IPv4 services will likely require (or be best served) direct forwarding towards Internet peering points while CGN mediated flows require access to a translator. CGN and non-CGN mediated flows post two fundamentally different forwarding needs.

The new CGN environments should not negatively impact the existing IPv4 service base by forcing all traffic to translation enabled network points since many flows do not require translation and this would reduce performance of the existing flows. This would also require massive scaling of the CGN which is a cost and efficiency concern as well.

Traffic flow and forwarding efficiency is considered important since networks are under considerable demand to deliver more and more bandwidth without the luxury of needless inefficiencies which can be introduced with CGN.

3.3. CGN By-Pass

The CGN environment is only needed for flows with translation requirements. Many flows which remain in a service provider environment, do not require translation. Such services include operator offered DNS Services, DHCP Services, NTP Services, Web Caching, Mail, News and other services which are local to the operator’s network.

The operator may want to leverage opportunities to offer third parties a platform to also provide services without translation. CGN By-pass can be accomplished in many ways, but a simplistic, deterministic and scalable model is preferred.

3.4. Routing Plane Separation

Many operators will want to engineer traffic separately for CGN flows versus flows which are part of the more traditional IPv4 environment. Many times the routing of these two major flow types differ, therefore route separation may be required.

Routing plane separation also allows the operator to utilize other addressing techniques, which may not be feasible on a single routing plane. Such examples include the use of overlapping private address space [RFC1918] or use of other IPv4 space which may overlap globally within the operator’s network.

3.5. Flexible Deployment Options

Service providers operate complex routing environments and offer a variety of IPv4 based services. Many operator environments utilize distributed peering infrastructures for transit and peering and these may span large geographical areas and regions. A CGN solution should offer the operator an ability to place CGN translation points at various points within their network.

The CGN deployment should also be flexible enough to change over time as demand for translation services increase. In turn, the deployment will need to then adapt as translation demand decreases caused by the transition of flows to IPv6. Translation points should be able to be placed and moved with as little re-engineering effort as possible minimizing the risks to the customer base.

Depending on hardware capabilities, security practices and IPv4 address availability, the translation environments my need to be segmented and/or scaled over time to meet organic IPv4 demand growth. Operators will want to seek deployment models which are conducive to meeting these goals as well.

3.6. IPv4 Overlap Space

IP address overlap for CGN translation realms may be required if insufficient IPv4 addresses are available within the service provider environment to assign internally unique IPs to the CGN customer base . The CGN deployment should provide mechanisms to manage IPv4 overlap if required.

3.7. Transactional Logging for LSN Systems

CGNs may require transactional logging since the source IP and related transport protocol information is not easily visible to external hosts and system.

If needed, the CGN systems should be able to generate logs which identify 'internal' host parameters (i.e. IP/Port) and associated them to external translated parameters imposed by the translator. The logged information should be stored on the CGN hardware and/or exported to an external system for processing. Operators may need to keep track of this information (securely) to meet regulatory and/or legal obligations. Further information can be found in [I-D.ietf-behave-lsn-requirements] with respect to CGN logging requirements (Logging Section).

3.8. Additional CGN Requirements

The CGN platform will also need to meet the needs of additional requirements such as Bulk Port Allocation and other CGN device specific functions. These additional requirements are captured within [I-D.ietf-behave-lsn-requirements].

4. MPLS/VPN based CGN Framework

The MPLS/VPN [RFC4364] framework for CGN segregates the 'pre-translated' realms within the service provider space into layer-3 MPLS/VPNs. The operator can deploy a single realm for all CGN based flows, or can deploy multiple realms based on translation demand and other factors such as geographical proximity. A realm in this model refers to a 'VPN' which shares a unique RD/RT combination, routing plane and forwarding behaviours.

The MPLS/VPN infrastructure provides control plane and forwarding separation for the traditional IPv4 service environment and CGN environment(s). The separation allows for routing information (such as default routes) to be propagated separately for CGN and non-CGN based customer flows. Traffic can be efficiently routed to the Internet for normal flows, and routed directly to translators for CGN mediated flows. Although many operators may run a "default-route-free" core, IPv4 flows which require translation must obviously be routed first to a translator, so a default route is acceptable for the pre-translated realms.

The physical location of the VRF Termination point for a MPLS/VPN enabled CGN can vary and be located anywhere within the operator's network. This model fully virtualizes the translation service from the base IPv4 forwarding environment which will likely carrying Internet bound traffic. The base IPv4 environment can continue to service traditional IPv4 customer flows plus post translated CGN flows.

Figure 1 provides a view of the basic model. The Access node provides CPE access to either the CGN VRF or the Global Routing Table, depending on whether the customer receives a private or public IP. Translator mediated traffic follows an MPLS LSP which can be setup dynamically and can span one hop, or many hops (with no need for complex routing policies). Traffic is then forwarded to the translator (shown below) which can be an external appliance or integrated into the VRF Termination (Provider Edge) router. Once traffic is translated, it is forwarded to the global routing table for general Internet forwarding. The Global Routing table can also be a separate VRF (Internet Access VPN/VRF) should the provider

choose to implement their Internet based services in that fashion. The translation services are effectively overlaid onto the network, but are maintained within a separate forwarding and control plane.

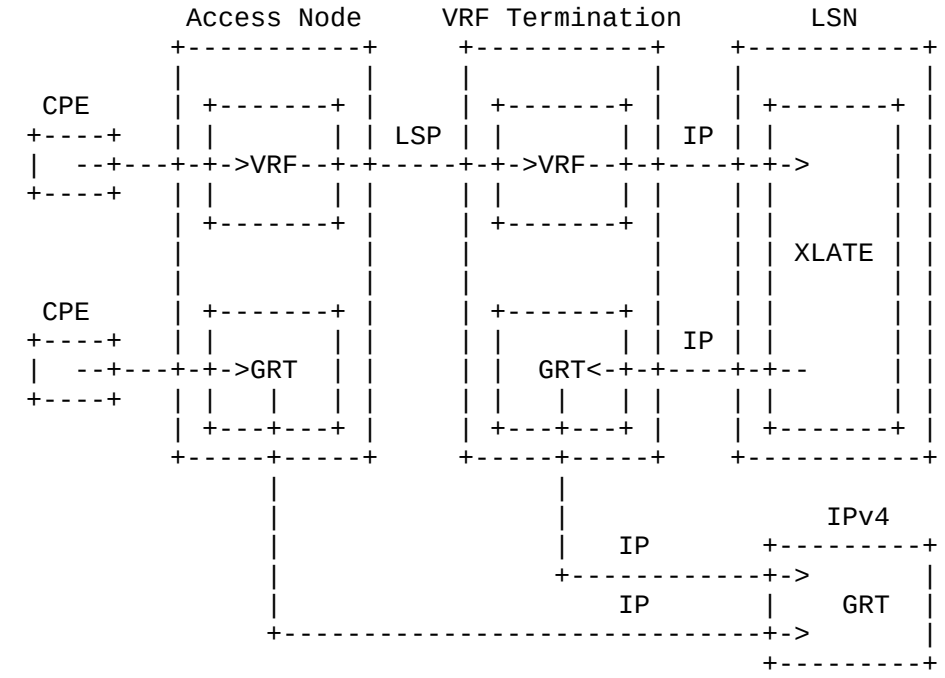


Figure 1: Basic MPLS/VPN CGN Model

If more then one VRF (translation realm) is used within the operator's network, each VPN instance can manage CGN flows independently for the respective realm. Various redundancy models can be used within this architecture to support failover from one physical CGN hardware instance to another. If state information needs to be passed or maintained between hardware instances, the vendor would need to enable this feature in a suitable manner.

4.1. Service Separation

The MPLS/VPN CGN framework supports route separation. The traditional IPv4 flows can be separated at the access node (Initial Layer 3 service point) from those which require translation. This type of service separation is possible on common technologies used for Internet access within many operator networks. Service separation can be accomplished on common access technology including

those used for DOCSIS (CMTS), Ethernet Access, DSL (BRAS), and Mobile Access (GGSN/ASN-GW) architectures.

4.2. Internal Service Delivery

Internal services can be delivered directly to the privately addressed endpoint within the CGN domain without translation. This can be accomplished using direct route exchange (import/export) between the CGN VRFs and the Services VRFs. The previous statement assumes the provider puts key services into a VRF for simple route exchange. This model allows the provider to maintain separate forwarding rules for translated flows, which require a pass through the translator to reach external network entities, versus those flows which need to access internal services. This operational detail can be advantageous for a number of reasons.

First, the provider can reduce the load on the translator since internal services do not need to be factored into the scaling of the CGN hardware. Secondly, more direct forwarding paths can be maintained providing better network efficiency. Thirdly, geographic locations of the translators and the services infrastructure can be deployed in a location in an independent manner. Additionally, the operator can allow CGN subject endpoints to be accessible via an untranslated path reducing the complexities of provider initiated management flows. This last point is of key interest since NAT removes transparency to the end device in normal cases.

Figure 2 below shows how internal services are provided untranslated since flows are sent directly from the access node to the services node/VRF via an MPLS LSP. This traffic is not forwarded to the CGN translator and therefore is not subject to problematic behaviours related to NAT. The services VRF contains routing information which can be "imported" into the access node VRF and the CGN VRF routing information can be "imported" into the Services VRF.

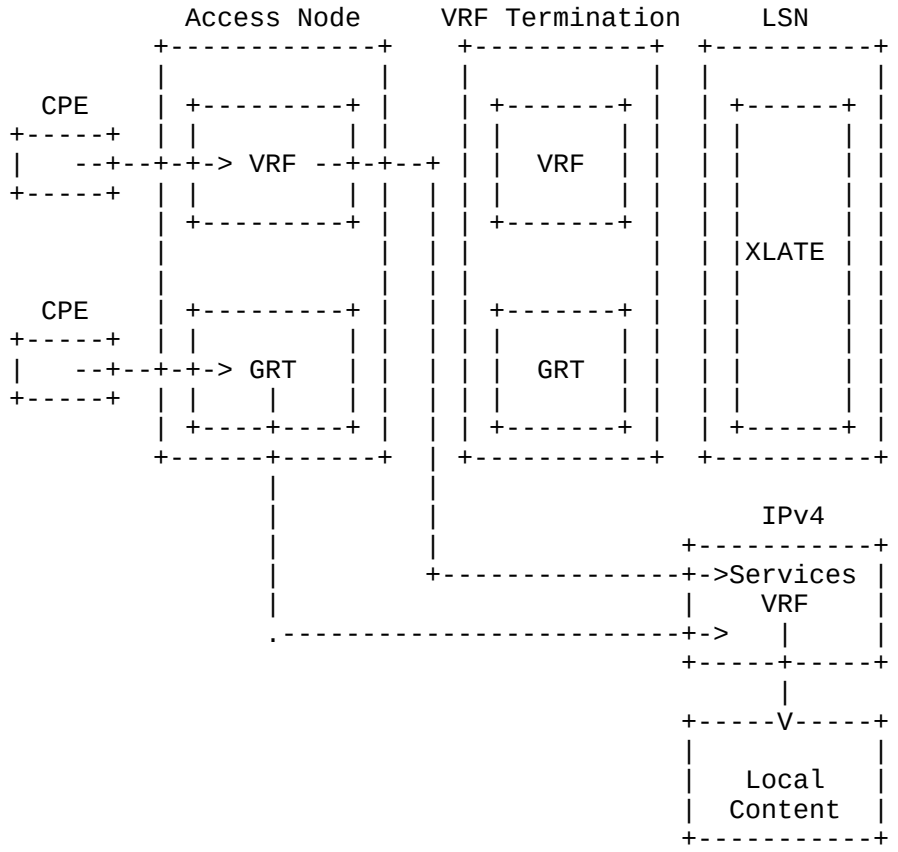


Figure 2: Internal Services and CGN By-Pass

This demonstrates the ability to offer CGN By-Pass in a simple and deterministic manner without the need of policy based routing or traffic engineering.

4.2.1. Dual Stack Operation

The MPLS/VPN CGN model can also be used in conjunction with IPv4/IPv6 dual stack service modes. Since many providers will use CGNs on an interim basis while IPv6 matures within the global Internet or due to technical constraints, a dual stack option is of strategic importance. Operators can offer this dual stack service for both traditional IPv4 (global IP) endpoints and CGN mediated endpoints.

Operators can separate the IP flows for IPv4 and IPv6 traffic, or use other routing techniques to move IPv6 based flows towards the GRT

(Global Routing Table or Instance) while allowing IPv4 flows to remain within the IPv4 CGN VRF for translator services.

The Figure 3 below shows how IPv4 translation services can be provided alongside IPv6 based services. The model shown allows the provider to enable CGN to manage IPv4 flows (translated) and IPv6 flows are routed without translation efficiently towards the Internet. Once again, forwarding of flows to the translator does not impact IPv6 flows which do not require this service.

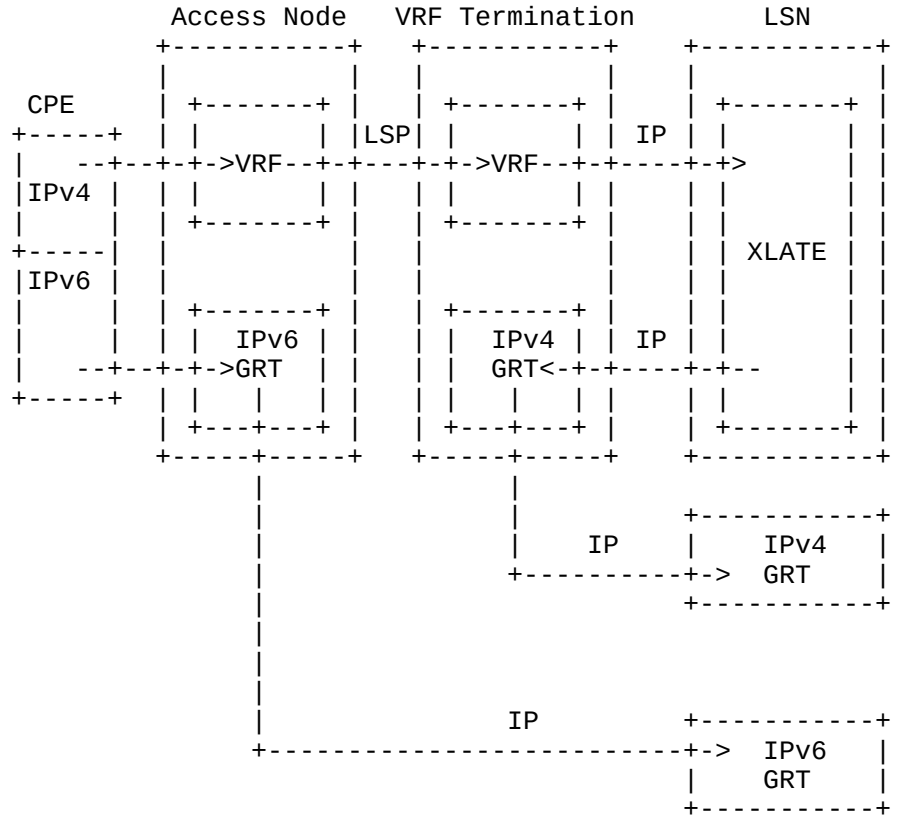


Figure 3: CGN with IPv6 Dual Stack Operation

4.3. Deployment Flexibility

The CGN translator services can be moved, separated or segmented (new translation realms) without the need to change the overall translation design. Since dynamic LSPs are used to forward traffic

from the access nodes to the translation points, the physical location of the VRF termination points can vary and be changed easily.

This type of flexibility allows the service provider to initially deploy more centralized translation services based on relatively low loading factors, and distribute the translation points over time to improve network traffic efficiencies and support higher translation load.

Although traffic engineered paths are not required within the MPLS/VPN deployment model, nothing precludes an operator from using technologies like MPLS with Traffic Engineering [RFC3031]. Additional routing mechanisms can be used as desired by the provider and can be seen as independent. There is no specific need to diversify the existing infrastructure in most cases.

4.4. Comparison of MPLS/VPN Option versus other CGN Attachment Options

Other integration architecture options exist which can attach CGN based service flows to a translator instance. Alternate options which can be used to attach such services include:

- IEEE 802.1Q for direct attachment to a next hop translator;

- Policy Based Routing (Static) to direct translation bound traffic to a network based translator;

- Traffic Engineering or;

- Multiple Routing Topologies

4.4.1. IEEE 802.1Q

IEEE 802.1Q can be used to associate separated traffic from the access node to the next hop router’s CGN instance. This technology option may limit the CGN placement to the next hop router unless a second technology option is paired with it to extend connectivity deeper in the network.

This option is most effective if CGN instances are placed directly upstream of the access node. Distributed CGN instance placement is not likely an initial stage of the CGN deployment due to cost and demand factors.

4.4.2. Policy Based Routing

Policy Based Routing (PBR) provides another option to direct CGN mediated flows to a translator. PBR options, although possible, are difficult to maintain (static policy) and must be configured throughout the network with considerable maintenance overhead.

More centralized deployments may be difficult or too onerous to deploy using Policy Based Routing methods. Policy Based Routing would not achieve route separation (unless used with others options), and may add complexities to the providers' routing environment.

4.4.3. Traffic Engineering

Traffic Engineering can also be used to direct traffic from an access node towards a translator. Traffic Engineering, like MPLS-TE, may be difficult to setup and maintain. Traffic Engineering provides additional benefits if used with MPLS by adding potentials for faster path re-convergence. Traffic Engineering paths would need to be updated and redefined overtime as CGN translation points are augmented or moved.

4.4.4. Multiple Routing Topologies

Multiple routing topologies can be used to direct CGN based flows to translators. This option would achieve the same basic goal as the MPLS/VPN option but with additional implementation overhead and platform configuration complexity. Since operator based translation is expected to have an unknown lifecycle, and may see various degrees of demand (dependant on operator IPv4 Global space availability and shift of traffic to IPv6), it may be too large of an undertaking for the provider to enabled this as their primary option for CGN.

5. Experiences

6. Basic Integration and Requirements Support

The MPLS/VPN CGN environment has been successfully integrated into real network environments utilizing existing network service delivery mechanisms. It solves many issues related to provider based translation environments, while still subject to problematic behaviours inherent within NAT.

Key issues which are solved or managed with the MPLS/VPN option include:

- Centralized and Distributed Deployment model support

- Routing Plane Separation for CGN flows versus traditional IPv4 flows

- Flexible Translation Point Design (can relocate translators and split translation zones easily)

- Low maintenance overhead (dynamic routing environment with little maintenance of separate routing infrastructure other then management of MPLS/VPNs)

- CGN By-pass options (for internal and third party services which exist within the provider domain)

- IPv4 Translation Realm overlap support (can reuse IP addresses between zones with some impact to extranet service model)

- Simple failover techniques can be implemented with redundant translators, such as using a second default route

7. Performance

The MPLS/VPN CGN model was observed to support basic functions which are typically used by customers within an operator environment. Examples of successful operation include:

- Traditional Web (HTTP) Surfing (client initiated)

- Internet Video Streaming

- HTTP Based Client Connections

- High Connection Count sites (i.e. Google Maps)

- Email Transaction Support (POP, IMAP, SMTP)

- Instant Messaging Support (Online Status, File transfers, text chat)

- ICMP Operation (client initiated Echo, Traceroute)

- Peer to Peer application support (download)

- DNS (based on services extranet option, but was problematic when passed through a translator)

CGNs are still subject to problematic connectivity even within the MPLS/VPN technology approach. Problems which arise, or are not inherently addressed in this model include:

- Inward services from the Internet to the CPE
- Web session tracking
- Restricting usage and/or access based on source IP
- Abuse mitigation (masquerade of potential offenders)
- Increased network or server IDS false positives
- Increased customer risk for session hijacking
- Exceeding firewall TCP/UDP limits
- Customer identification (external site)
- Poor source based load balancing
- Customer usage tracking / Ad insertion
- Other applications or operations may be negatively impacted

8. IANA Considerations

There are not specific IANA considerations known at this time with the architecture described herein. Should a provide choose to use non-assigned IP address space within their translation realms, then considerations may apply.

9. Security Considerations

The same security considerations would typically exist for CGN deployments when compared with traditional IPv4 based services. With the MPLS/VPN model, the operator would want to consider security issues related to offering IP services over MPLS.

If a provider plans to operate the pre-translation realm (CPE towards translator IPv4 zone) as a non-public like network, then additional security measures may be needed to secure this environment. It is however the position in this document that CGN realms are public domains which utilize non-Internet routable IP addresses for endpoint addressing.

10. Conclusions

The MPLS/VPN delivery method for a CGN deployment is an effective and scalable way to deliver mass translation services. The architecture avoids the complex requirements of traffic engineering and policy based routing when combining these new service flows to existing IPv4 operation. This is advantageous since the NAT44/CGN environments should be introduced with as little impact as possible and these environments are expected to change over time.

The MPLS/VPN based CGN architecture solves many of this issues related to deploying this technology in existing operator networks.

11. Acknowledgements

Thanks to the following people for their participating in integrating and testing the CGN environment: Chris Metz, Syd Alam, Richard Lawson, John E Spence.

Additional thanks for the following people for the guidance on IPv6 transition considerations: John Jason Brzozowski, Chris Donley, Jason Weil, Lee Howard, Jean-Francois Tremblay

12. References

12.1. Normative References

[I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", draft-ietf-behave-lsn-requirements-05 (work in progress), November 2011.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

12.2. Informative References

[I-D.weil-shared-transition-space-request]
Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA Reserved IPv4 Prefix for Shared Address Space", draft-weil-shared-transition-space-request-15 (work in progress), February 2012.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets",

BCP 5, RFC 1918, February 1996.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC5969] Townsley, W. and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification", RFC 5969, August 2010.

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

Authors' Addresses

Victor Kuarsingh (editor)
Rogers Communications
8200 Dixie Road
Brampton, Ontario L6T 0C1
Canada

Email: victor.kuarsingh@gmail.com
URI: http://www.rogers.com

John Cianfarani
Rogers Communications
8200 Dixie Road
Brampton, Ontario L6T 0C1
Canada

Email: john.cianfarani@rci.rogers.com
URI: http://www.rogers.com

Port Control Protocol
Internet-Draft
Intended status: BCP
Expires: July 11, 2013

R. Penno
Cisco
S. Perreault
Viagenie
S. Kamiset
Consultant
M. Boucadair
France Telecom
K. Naito
NTT

January 07, 2013

Network Address Translation (NAT) Behavioral Requirements Updates draft-penno-behave-rfc4787-5382-5508-bis-04

Abstract

This document clarifies and updates several requirements of RFC4787, RFC5382 and RFC5508 based on operational and development experience. The focus of this document is NAPT44.

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 11, 2013.

Copyright Notice

Table of Contents

1. Terminology

The reader should be familiar with all terms defined in RFC2663 [RFC2663],RFC4787 [RFC4787],RFC5382 [RFC5382],RFC5508 [RFC5508]

2. Introduction

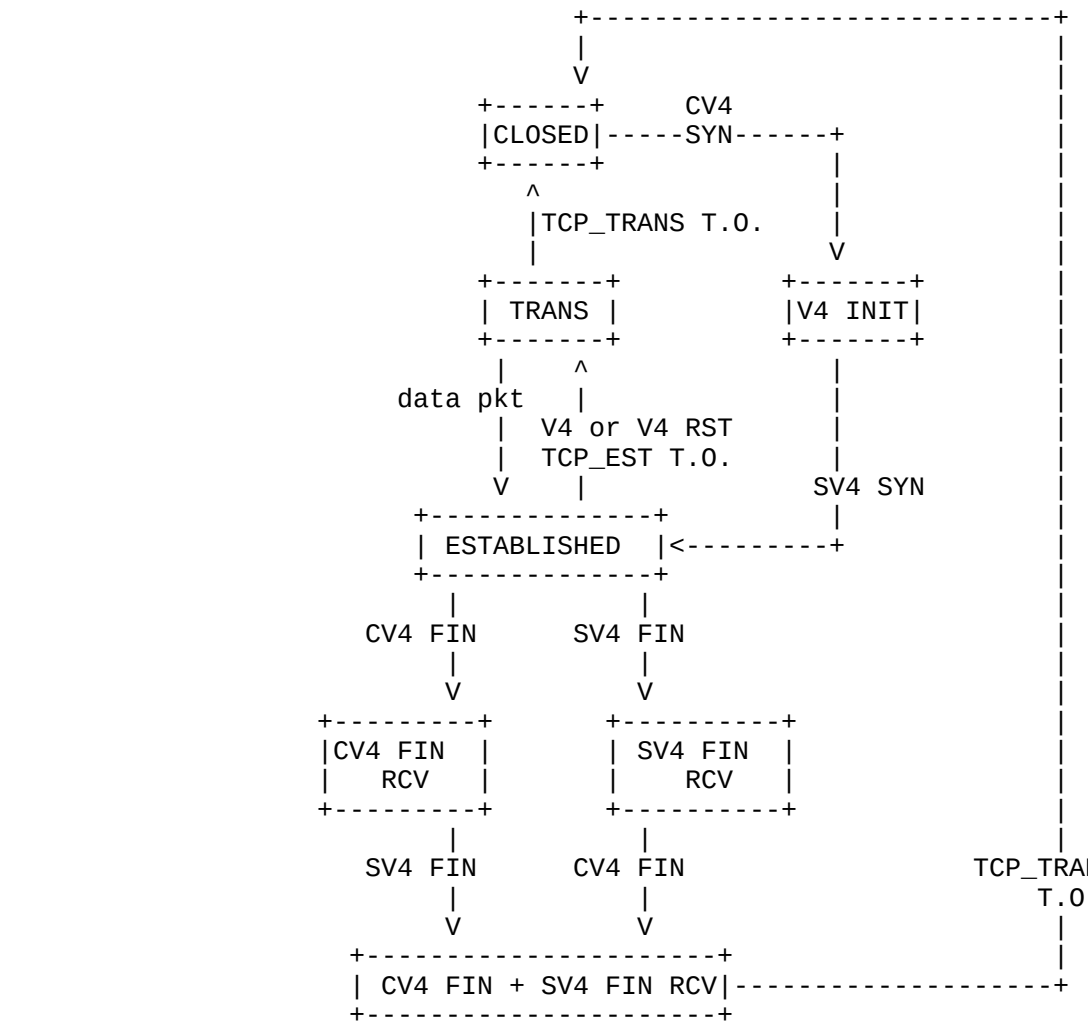
[RFC4787], [RFC5382] and [RFC5508] greatly advanced NAT interoperability and conformance. But with widespread deployment and evolution of NAT more development and operational experience was acquired some areas of the original documents need further clarification or updates. This documents provides such clarifications and updates.

2.1. Scope

This document focuses solely on NAPT44 and its goal is to clarify, fill gaps or update requirements of [RFC4787], [RFC5382] and [RFC5508]. It is out of the scope of this document the creation of completely new requirements not associated with the documents cited above. New requirements would be better served elsewhere and if they are CGN specific in [I-D.ietf-behave-lsn-requirements]

3. TCP Session Tracking

[RFC5382] specifies TCP timers associated with various connection states but does not specify the TCP state machine a NAPT44 should use as a basis to apply such timers. The TCP state machine below, adapted from [RFC6146], provides guidance on how TCP session tracking could be implemented - it is non-normative.



(postamble)

3.1. TCP Transitory Connection Idle-Timeout

[RFC5382]:REQ-5 The transitory connection idle-timeout is defined as the minimum time a TCP connection in the partially open or closing phases must remain idle before the NAT considers the associated session a candidate for removal. But the document does not clearly states if these can be configured separately. This document clarifies that a NAT device SHOULD provide different knobs for configuring the open and closing idle timeouts. This document further acknowledges that most TCP flows are very short (less than 10 seconds) [FLOWRATE][TCPWILD] and therefore a partially open timeout

of 4 minutes might be excessive if security is a concern. Therefore it MAY be configured to be less than 4 minutes in such cases. There also may be a case that timeout of 4 minutes might be excessive. The case and the solution are written below.

3.1.1. Port resources limited case

After IPv4 addresses run out, IPv4 address resources will be further restricted site-by-site. If global IPv4 address are shared between several clients, assignable port resources at each client will be limited.

NAT is a tool that is widely used to deal with this IPv4 address shortage problem. However, the demand for resources to provide Internet access to users and devices will continue to increase. IPv6 is a fundamental solution to this problem, but the deployment of IPv6 will take time.

In some cases, e.g. browsing a dynamic web page for a map service, a lot of sessions are used by the browser, and a number of ports are eaten up in a short time. What is worse is that when a NAT is between a PC and a server, TIME_WAIT state of each TCP connection is kept for certain period, typically for four minutes, which consumes port resources. Therefore, new connections cannot be established.

This problem is caused or worsened by the following behavior.

> TIME_WAIT state assigned for a TCP connection remains active for 2MSL after the last ACK to the last FIN is transferred.

To reuse resources effectively, reducing TIME_WAIT without making any bad effect is important. To reduce TIME_WAIT, [RFC6191] is proposed for clients and remote hosts. To prevent bad effects, there is a PAWS mechanism, which prevent the old duplicate problem. We propose mechanisms adopting to NAT, to change the TIME_WAIT behavior that make it possible to save addresses and ports resources.

3.1.1.1. RFC6191 Reducing the TIME-WAIT State Using TCP Timestamps

[RFC6191] defines a mechanism for reducing the TIME_WAIT state using TCP timestamps and sequence numbers. When a connection request is received with a four-tuple that is in the TIME-WAIT state, the connection request may be accepted if the sequence number or the timestamp of the incoming SYN segment is greater than the last sequence number seen on the previous incarnation of the connection

3.1.1.2. TCP TIME_WAIT

The TCP TIME_WAIT state is described in [RFC0793]. The TCP TIME_WAIT state needs to be kept for 2MSL before a connection is CLOSED, for the reasons below.

1: In the event that packets from a session are delayed in the in-between network, and delivered to the end relatively later, we should prevent the packets from being transferred and interpreted as a packet that belongs to a new session.

2: If the remote TCP has not received the acknowledgment of its connection termination request, it will re-send the FIN packet several times.

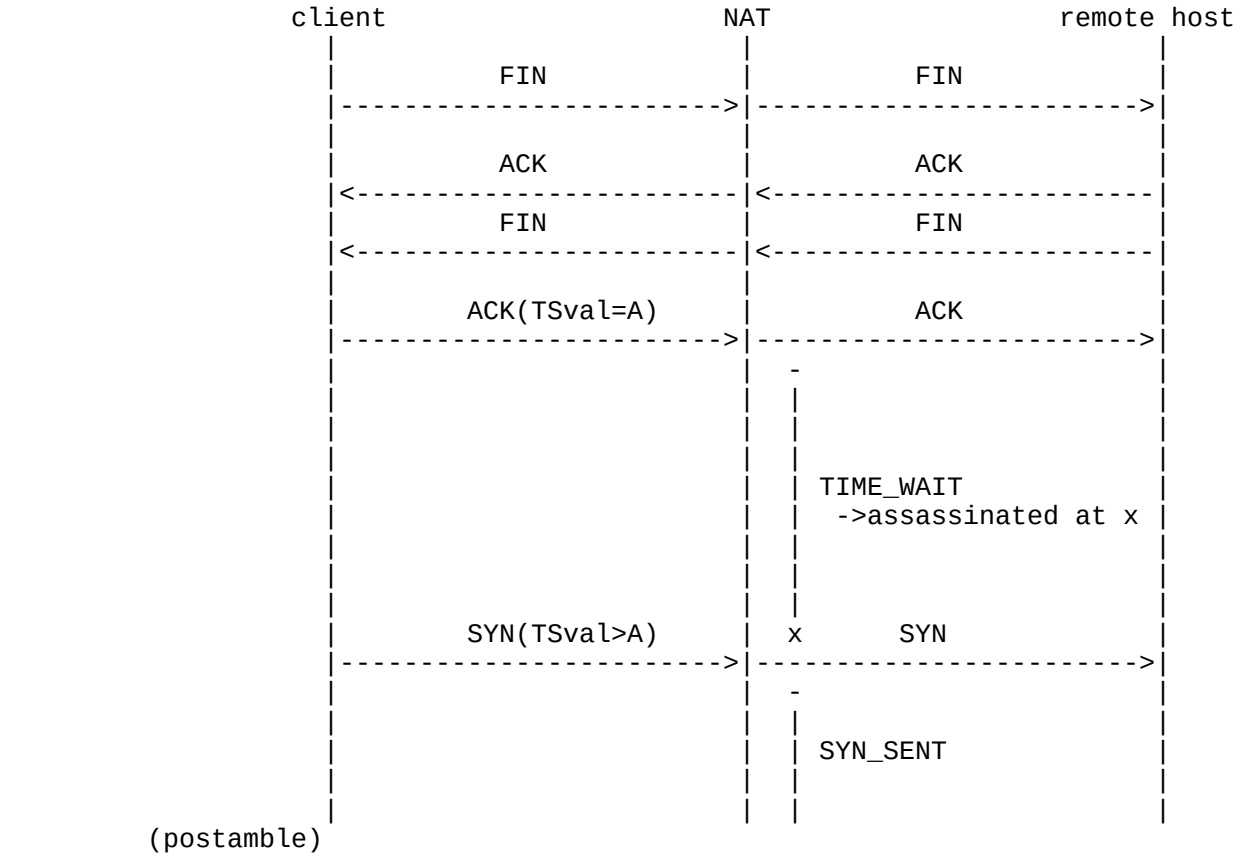
These points are important for the TCP to work without problems.

3.1.1.3. Protect Against Wrapped Sequence numbers (PAWS)

The TCP sequence number wraps frequently especially in a high bandwidth session. PAWS is used to prevent old duplicate packets that occurred in a previous session from being transferred to the new session whose valid TCP sequence numbers happen to overlap with the old duplicate packets. This is implemented by introducing TCP timestamp option, and checking the timestamp option value of each packet. PAWS is described in [RFC1323].

3.1.2. Proposal: Apply RFC6191 and PAWS to NAT

This section proposes to apply [RFC6191] mechanism at NAT. This mechanism MAY be adopted for both clients' and remote hosts' TCP active close.



(postamble)

Also, PAWS works to discard old duplicate packets at NAT. A packet can be discarded as an old duplicate if it is received with a timestamp or sequence number value less than a value recently received on the connection.

To make these mechanisms work, we should concern the case that there are several clients with nonsuccessive timestamp or sequence number values are connected to a NAT device (i.e. not monotonically increasing among clients). Two mechanisms to solve this mechanism and applying [RFC6191] and PAWS to NAT are described below. These mechanisms are optional.

3.1.2.1. Rewrite timestamp and sequence number values at NAT

Rewrite timestamp and sequence number values of outgoings packets at NAT to be monotonically increasing. This can be done by adopting following mechanisms at NAT.

A: Store the newest rewritten value of timestamp and sequence number as the "max value at the time".

B: NAT rewrite timestamp and sequence number values of incoming packets to be monotonically increasing.

When packets come back as replies from remote hosts, NAT rewrite again the timestamp and sequence number values to be the original values. This can be done by adopting following mechanisms at NAT.

C: Store the values of original timestamp and sequence number of packets, and rewritten values of those.

3.1.2.2. Split an assignable number of port space to each client

Adopt following mechanisms at NAT.

A: Choose clients that can be assigned ports.

B: Split assignable port numbers between clients.

Packets from other clients which are not chosen by these mechanisms are rejected at NAT, unless there is unassigned port left.

3.1.2.3. Resend the last ACK to the resended FIN

We should concern another case to make RFC6191 work at NAT. In case the remote TCP could not receive the acknowledgment of its connection termination request, NAT, on behalf of clients. resends the last ACK packet when it receives an FIN packet of the previous connection, and when the state of the previous connection is deleted from the NAT. This mechanism MAY be used when clients starts closing process, and the remote host could not receive the last ACK.

3.1.2.4. Remote host behavior of several implementations

To solve the port shortage problem on the client side, the behavior of remote host should be compliant to [RFC6191] or the mechanism written in 4.2.2.13 of [RFC1122], since NAT may reuse the same 5 tuple for a new connection.We have investigated behaviors of OSes (e.g., Linux, FreeBSD, Windows, MacOS), and found that they implemented the server side behavior of the above two.

3.2. TCP RST

[RFC5382] leaves the handling of TCP RST packets unspecified. This document does not try standardize such behavior but clarifies based on operational experience that a NAT that receives a TCP RST for an active mapping and performs session tracking MAY immediately delete the sessions and remove any state associated with it. If the NAT device that performs TCP session tracking receives a TCP RST for the first session that created a mapping, it MAY remove the session and the mapping immediately.

4. Port Overlapping behavior

There may be another solution to the address resource restricted environment written in 3.1.1. Also NAT are required to be maped endpoint-independent in [RFC4787] and [RFC5382] REQ-1, the mechanism below MAY be one optional implement to NAT.

If destination addresses and ports are different for outgoing connections started by local clients, NAT MAY assign the same external port as the source ports for the connections. The port overlapping mechanism manages mappings between external packets and internal packets by looking at and storing the 5-tuple (protocol, source address, source port, destination address, destination port) of them. This enables concurrent use of a single NAT external port for multiple transport sessions, which enables NAT to work correctly in IP address resource limited network.

Discussions:

[RFC4787]and[FC5382] requires "endpoint-independent mapping" at NAT, and port overlapping NAT cannot meet the requirement. This mechanism can degrade the transparency of NAT in that its mapping mechanism is endpoint-dependent and makes NAT traversal harder. However, if a NAT adopts endpoint-independent mapping together with endpoint-dependent filtering, then the actual behavior of the NAT will be the same as port overlapping NAT. It should also be noted that a lot of existing NAT devices(e.g., SEIL, FITELnet Series) adopted this port overlapping mechanism.

A: Reference URL for SEIL -> www.seil.jp

B: Reference URL for FITELnet -> www.furukawa.co.jp/fitelnet

The netfilter, which is a popular packet filtering mechanism for

Linux, also adopts port overlapping behavior.

5. Address Pooling Paired (APP)

[RFC4787]: REQ-2 [RFC5382]:ND Address Pooling Paired behavior for NAT is recommended in previous documents but behavior when a public IPv4 run out of ports is left undefined. This document clarifies that if APP is enabled new sessions from a subscriber that already has a mapping associated with a public IP that ran out of ports SHOULD be dropped. The administrator MAY provide a knob that allows a NAT device to starting using ports from another public IP when the one that anchored the APP mapping ran out of ports. This is trade-off between subscriber service continuity and APP strict enforcement. (NE: It is sometimes referred as 'soft-APP')

6. EIF Security

[RFC4787]:REQ-8 and [RFC5382]:REQ-3 End-point independent filtering could potentially result in security attacks from the public realm. In order to handle this, when possible there MUST be strict filtering checks in the inbound direction. A knob SHOULD be provided to limit the number of inbound sessions and a knob SHOULD be provided to enable or disable EIF on a per application basis. This is specially important in the case of Mobile networks where such attacks can consume radio resources and count against the user quota.

7. EIF Protocol Independence

[RFC4787]:REQ-8 and[RFC5382]: REQ-3 Current RFCs do not specify whether EIF mappings are protocol independent. In other words, if a outbound TCP SYN creates a mapping it is left undefined whether inbound UDP packets create sessions and are forwarded. EIF mappings SHOULD be protocol independent in order allow inbound packets for protocols that multiplex TCP and UDP over the same IP: port through the NAT and maintain compatibility with stateful NAT64 RFC6146 [RFC6146]. But the administrator MAY provide a configuration knob to make it protocol dependent.

8. EIF Mapping Refresh

[RFC4787]: REQ-6 [RFC5382]: ND The NAT mapping Refresh direction MAY have a "NAT Inbound refresh behavior" of "True" but it does not clarifies how this applies to EIF mappings. The issue in question is whether inbound packets that match an EIF mapping but do not create a

new session due to a security policy should refresh the mapping timer. This document clarifies that even when a NAT device has a inbound refresh behavior of TRUE, that such packets SHOULD NOT refresh the mapping. Otherwise a simple attack of a packet every 2 minutes can keep the mapping indefinitely.

8.1. Outbound Mapping Refresh and Error Packets

In the case of NAT outbound refresh behavior there might be certain types of packets that should not refresh the mapping. For example, if the mapping is kept alive by ICMP Error or TCP RST outbound packets sent as response to inbound packets, these SHOULD NOT refresh the mapping.

9. EIM Protocol Independence

[RFC4787] [RFC5382]: REQ-1 Current RFCs do not specify whether EIM are protocol independent. In other words, if a outbound TCP SYN creates a mapping it is left undefined whether outbound UDP can reuse such mapping and create session. On the other hand, Stateful NAT64 [RFC6146] clearly specifies three binding information bases (TCP, UDP, ICMP). This document clarifies that EIM mappings SHOULD be protocol dependent . A knob MAY be provided in order allow protocols that multiplex TCP and UDP over the same source IP and port to use a single mapping.

10. Port Parity

A NAT devices MAY disable port parity preservation for dynamic mappings. Nevertheless, A NAT SHOULD support means to explicitly request to preserve port parity (e.g., [I-D.boucadair-pcp-rtp-rtcp]).

11. Port Randomization

A NAT SHOULD follow the recommendations specified in Section 4 of [RFC6056] especially: "A NAPT that does not implement port preservation [RFC4787] [RFC5382] SHOULD obfuscate selection of the ephemeral port of a packet when it is changed during translation of that packet. A NAPT that does implement port preservation SHOULD obfuscate the ephemeral port of a packet only if the port must be changed as a result of the port being already in use for some other session. A NAPT that performs parity preservation and that must change the ephemeral port during translation of a packet SHOULD obfuscate the ephemeral ports. The algorithms described in this document could be easily adapted such that the parity is preserved

(i.e., force the lowest order bit of the resulting port number to 0 or 1 according to whether even or odd parity is desired)."

12. IP Identification (IP ID)

A NAT SHOULD handle the Identification field of translated IPv4 packets as specified in Section 9 of [I-D.ietf-intarea-ipv4-id-update].

13. ICMP Query Mappings Timeout

Section 3.1 of [RFC5508] says that ICMP Query Mappings are to be maintained by NAT device. However, RFC doesn't discuss about the Query Mapping timeout values. Section 3.2 of that RFC only discusses about ICMP Query Session Timeouts. ICMP Query Mappings MAY be deleted once the last the session using the mapping is deleted.

14. Hairpinning Support for ICMP Packets

[RFC5508]:REQ-7 This requirement specifies that NAT devices enforcing Basic NAT MUST support traversal of hairpinned ICMP Query sessions. This implicitly means that address mappings from external address to internal address (similar to Endpoint Independent Filters) MUST be maintained to allow inbound ICMP Query sessions. If an ICMP Query is received on an external address, NAT device can then translate to an internal IP. [RFC5508]:REQ-7 This requirement specifies that all NAT devices (i.e., Basic NAT as well as NAPT devices) MUST support the traversal of hairpinned ICMP Error messages. This too requires NAT devices to maintain address mappings from external IP address to internal IP address in addition to the ICMP Query Mappings described in section 3.1 of that RFC.

15. IANA Considerations

TBD

16. Security Considerations

In the case of EIF mappings due to high risk of resource crunch, a NAT device MAY provide a knob to limit the number of inbound sessions spawned from a EIF mapping.

[TCP-Security] contains a detailed discussion of the security

implications of TCP Timestamps and of different timestamp generation algorithms.

17. Acknowledgements

Thanks to Dan Wing, Suresh Kumar, Mayuresh Bakshi, Rajesh Mohan and Senthil Sivamular for review and discussions

18. References

18.1. Normative 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-29 (work in progress), November 2012.

[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.

[RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

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

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.

[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.

[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.

[RFC6191] Gont, F., "Reducing the TIME-WAIT State Using TCP Timestamps", BCP 159, RFC 6191, April 2011.

18.2. Informative References

[FLOWRATE]
Zhang, Y., Breslau, L., Paxson, V., and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates".

[I-D.boucadair-pcp-rtp-rtcp]
Boucadair, M. and S. Sivakumar, "Reserving N and N+1 Ports with PCP", draft-boucadair-pcp-rtp-rtcp-05 (work in progress), October 2012.

[I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", draft-ietf-behave-lsn-requirements-10 (work in progress), December 2012.

[I-D.naito-nat-resource-optimizing-extension]
Kengo, K. and A. Matsumoto, "NAT TIME_WAIT reduction", draft-naito-nat-resource-optimizing-extension-02 (work in progress), July 2012.

[TCPWILD] Qian, F., Subhabrata, S., Spatscheck, O., Morley Mao, Z., and W. Willinger, "TCP Revisited: A Fresh Look at TCP in the Wild".

Authors' Addresses

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: repenno@cisco.com

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

Email: simon.perreault@viagenie.ca

Sarat Kamiset
Consultant
California

Phone:
Fax:

Mohamed Boucadair
France Telecom
Rennes,   35000
France

Email: mohamed.boucadair@orange.com

Kengo Naito
NTT
Tokyo
Japan

Email: kengo@lab.ntt.co.jp

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2012

R. Penno
A. Durand
Juniper Networks
A. Clauberg
Deutsche Telekom AG
L. Hoffmann
Bouygues Telecom

March 11, 2012

Stateless DS-Lite
draft-penno-softwire-sdnat-02

Abstract

This memo define a simple stateless and deterministic mode of operating a carrier-grade NAT as a backward compatible evolution of DS-Lite.

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 12, 2012.

Copyright Notice

Table of Contents

1. Introduction

DS-Lite [RFC6333],is a solution to deal with the IPv4 exhaustion problem once an IPv6 access network is deployed. It enables unmodified IPv4 application to access the IPv4 Internet over the IPv6 access network. In the DS-Lite architecture, global IPv4 addresses are shared among subscribers in the AFTR, acting as a Carrier-Grade NAT (CGN).

[I-D.ietf-softwire-public-4over6] extends the original DS-Lite model to offer a mode where the NAT function is performed in the CPE. This simplifies the AFTR operation as it does not have to perform the NAT function anymore, however, the flip side is that the address sharing function among subscribers was no longer available. [I-D.cui-softwire-b4-translated-ds-lite] introduces port restrictions, but does not completely specifies how the CPE acquires the information about its IPv4 address and its port range. More importantly, that draft does not explain how this solution can be deployed in a regular DS-Lite environment. This memo addresses these issues and clarifies the operation model.

Other approaches like variations of 4rd allows also for a full stateless operation of the decapsulation device. By introducing a strong coupling between the IPv6 address and the derived IPv4 address, they get rid of the per-subscriber state on the decapsulation devices. The approach take here argues that such per-subscriber state is not an issue as it is easily replicated among all decapsulation devices. Eliminating the strong coupling between IPv6 and IPv4 derived addresses, the approach presented here enables service providers a greater flexibility on how their limited pool of IPv4 addresses is managed. It also provide greater freedom on how IPv6 addresses are allocated, as sequential allocation is no longer a pre-requisite.

The approach presented here is stateless and deterministic. It is stateless is NAT bindings are maintained on the CPE, not on the AFTR. It is deterministic as no logs are required on the AFTR to identify which subscriber is using an external Ipv4 address and port.

The stateless DS-Lite architecture has the following characteristics:

- Backward compatible with DS-Lite. A mix of regular DS-Lite CPE and stateless DS-Lite CPEs can interoperate with a stateless DS-Lite AFTR.

- Zero log: Because the AFTR relies only on a per-subscriber mapping table that is reversible, the ISP does not need to keep any NAT binding logs.

- Stateless AFTR: There is no per-session state on the AFTRs. By leveraging this stateless and deterministic mode of operation, an ISP can deploy any number of AFTRs to provide redundancy and scalability at low cost. Because there is no per-flow state to maintain, AFTR can implement the functionality in hardware and perform it at high speed with low latency.

- Flexibility of operation: The ISP can add or remove addresses from the NAT pool without having to renumber the access network.

- Leverage IPv6: This stateless DS-Lite model leverage the IPv6 access network deployed by the ISPs.

2. Stateless DS-Lite CPE

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].

A Stateless DS-Lite CPE operates in similar fashion than a regular DS-Lite CPE, where the NAT function is re-introduced in CPE with a modification on how ports are managed.

2.1. Learning external IPv4 address

A stateless DS-Lite CPE MUST implement the DHCPv4 client relay option defined in [I-D.ietf-dhc-dhcpv4-over-ipv6] to learn is external IPv4 address. Other mechanism, such as manual configuration or TR69, MAY be implemented.

2.2. Learning external port range

A stateless DS-Lite CPE MUST implement the ICMP "port restricted" option defined later in this memo.

At boot time and later at intervals of 1h +/- a random number of seconds between 0 and 900), the stateless DS-Lite CPE MUST send packets with source port 0, source IPv4 address of the B4 element, destination IPv4 address 192.0.0.1 (the AFTR well-known IPv4 address) destination port 0, for each of the supported transport protocols (usually TCP and UDP). This will trigger an ICMP "port restricted" message from the AFTR.

After validating the content of the "ICMP port restricted" message, the stateless DS-Lite CPE MUST configure its port pool with it. If existing connections were using source ports outside of that range, the stateless DS-Lite CPE MUST terminate them.

2.3. Stateless DS-Lite CPE operation

The stateless DS-Lite CPE performs IPv4 NAPT from the internal RFC1918 addresses to the IPv4 address configured on the WAN interface, restricting its available ports to the range obtained as described above.

2.4. Host-based Stateless DS-Lite

Any host initiating directly a DS-Lite IPv4 over IPv6 tunnel can benefit from this techniques by implementing a 'virtual' stateless DS-Lite CPE function within its IP stack.

3. Stateless AFTR

3.1. Anycast IPv6 address for Stateless AFTR

All stateless AFTRs associated to a domain (or group of subscribers) will be configured with the same IPv6 address on the interface facing IPv6 subscribers. A route for that IPv6 address will be anycasted within the access network.

3.2. Stateless AFTR IPv4 address pool

All stateless AFTRs associated to a domain (or group of subscribers) MUST be configured with the same pool of global IPv4 addresses.

Routes to the pool of global IPv4 addresses configured on the stateless AFTRs will be anycasted by the relevant AFTRs within the ISP routing domain.

3.3. Stateless AFTR per-subscriber mapping table

Stateless AFTRs associated to a domain (or group of subscribers) MUST be configured with the same per-subscriber mapping table, associating the IPv6 address of the subscriber CPE to the external IPv4 address and port range provisioned for this subscriber.

Because the association IPv6 address --- IPv4 address + port range is not tied to a mathematical formula, the ISP maintains all flexibility to allocate independently IPv6 address and IPv4 addresses. In particular, IPv6 addresses do not have to be allocated sequentially and IPv4 resources can be modified freely.

IPv6 address	IPv4 address	port-range
2001:db8::1	1.2.3.4	1000-1999
2001:db8::5	1.2.3.4	2000-2999
2001:db8::a:1	1.2.3.4	3000-3999

Figure 1: Per-subscriber mapping table example

This per-subscriber mapping table can be implemented in various ways which details are out of scope for this memo. In its simplest form, it can be a static file that is replicated out-of-band on the AFTRs. In a more elaborated way, this table can be dynamically built using radius queries to a subscriber database.

3.4. Stateless AFTR decapsulation rules

Upstream IPv4 over IPv6 traffic will be decapsulated by the AFTR. The AFTR MUST check the outer IPv6 source address belongs to an identified subscriber and drop the traffic if not. The AFTR MUST then check the inner IPv4 header to make sure the IPv4 source address and ports are valid according to the per-subscriber mapping table.

If the inner IPv4 source address does not match the entry in the per-subscriber mapping table, the packet MUST be discarded and an ICMP 'administratively prohibited' message MAY be returned.

If the IPv4 source port number falls outside of the range allocated to the subscriber, the AFTR MUST discard the datagram and MUST send back an ICMP "port restricted" message to the IPv6 source address of the packet.

Fragmentation and reassembly is treated as in DS-Lite [RFC6333].

3.5. Stateless AFTR encapsulation rules

Downstream traffic is validated using the per-subscriber mapping table. Traffic that falls outside of the IPv4 address/port range entries in that table MUST be discarded. Validated traffic is then encapsulated in IPv6 and forwarded to the associated IPv6 address.

Fragmentation and reassembly is treated as in DS-Lite [RFC6333].

3.6. Redundancy and fail over

Because there is no per-flow state, upstream and downstream traffic can use any stateless AFTR.

3.7. SD-AFTR stateless domain

Using the DHCPv6 DS-Lite tunnel-end-point option, groups of subscribers and can be associated to a different stateless AFTR domain. That can allow for differentiated level of services, e.g. number of ports per customer device, QoS, bandwidth, value added services,...

4. Backward compatibility with DS-Lite

A number of service providers are, or are in the process of, deploying DS-Lite in their network. They are interested in evolving their design toward a stateless model. Backward compatibility is a critical issue, as, from an operational perspective, it is difficult to get all CPEs evolve at the same time.

So AFTRs have to be ready to service CPEs that are pure DS-Lite, some that are implementing only DHCPv4 over IPv6 and handle the NAT on the full IPv4 address themselves and some that also implement port restrictions via the ICMP message described here. For this reason, a AFTR operating in backward compatibility mode MAY decide to re-NAT upstream packets which source port number do not fall into the predefined range instead of simply dropping the packets.

The operating model is the following:

- Stateless DS-Lite: for CPEs that pre-NAT and pre-shape the source port space into the range assigned to the subscriber: decapsulate, check per-subscriber mapping, forward.

- B4-translated DS-Lite: for CPEs that performs NAT before encapsulation and are allocated a full IPv4 address: decapsulate, check per-subscriber mapping, forward.

- Re-shaper DS-Lite: for CPEs that pre NAT but fail to restrict the source ports: decapsulate, check per-subscriber mapping, re-NAT statefully the packets into the restricted port range, mark range as ’stateful’, forward.

- Regular DS-Lite: for regular DS-Lite CPEs that do not pre-NAT: decapsulate, NAT statefully, forward.

In such a backward compatibility mode, the AFTR is only operating statelessly for the stateless DS-Lite CPEs. It needs to maintain per-flow state for the regular DS-Lite CPEs and the non-ICMP port restricted compliant CPEs. In this legacy mode where per-flow state is required, the simple anycast-based fail-over mechanism is no longer available.

5. ICMP port restricted message

Note: this section may end-up being a separate Internet draft.

5.1. Introduction

In the framework of A+P RFC 6346 [RFC6346], sources may be restricted to use only a subset of the port range of a transport protocol associated with an IPv4 address. When that source transmit a packet with a source outside of the pre-authorized range, the upstream NAT will drop the packet and use the ICMP message defined here to inform the source of the actual port range allocated.

This memo defines such ICMP messages for TCP and UDP and leaves the definition of the ICMP option for other transport protocol for future work.

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].

5.2. Source port restricted ICMP

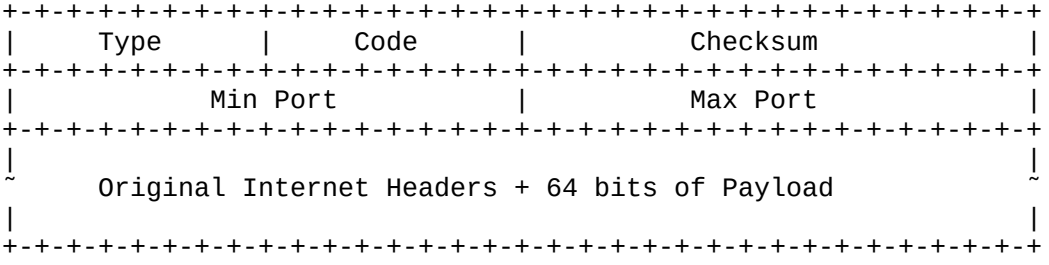


Figure 2: Source Port Restricted ICMP

Type: TBD for Source Port Restricted

Checksum: The checksum is the 16-bit ones’s complement of the one’s complement sum of the ICMP message starting with the ICMP Type. For

computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Code: 6 for TCP, 17 for UDP

Min Port: The lowest port number allocated for that source.

Max Port: The highest port number allocated for that source.

5.3. Host behavior

A host receiving an ICMP type TBD message for a given transport protocol SHOULD NOT send packets sourced by the IP address(es) corresponding to the interface that received that ICMP message with source ports outside of the range specified for the given transport protocol.

Packets sourced with port numbers outside of the restricted range MAY be dropped or NATed upstream to fit within the restricted range.

A host MUST NOT take port restriction information applying to a given IP address and transport protocol and applies it to other IP addresses on other interfaces and/or other transport protocols.

If Min Port = 0 and Max Port = 65535, it indicates that the entire port range for the given transport protocol is available. If such 'full range' messages are received for all transport protocols, the host can take this as an indication that its IP address is probably not shared with other devices.

In order to mitigate possible man in the middle attacks, a host MUST discard ICMP type TBD messages if the associated port range (Max Port - Min Port) is lower than 64.

6. IANA Considerations

IANA is to allocated a code point for this ICMP message type.

7. Security Considerations

This ICMP message type has the same security properties as other ICMP messages such as Redirect or Destination Unreachable. A man-in-the-middle attack can be mounted to create a DOS attack on the source. Ingress filtering on network boundary can mitigate such attacks. However, in case such filtering measures are not enough, the additional provision that a host MUST discard such ICMP message with

a port range smaller than 64 can mitigate even further such attacks.

As described in [RFC6269], with any fixed size address sharing techniques, port randomization is achieved with a smaller entropy.

Recommendations listed in [RFC6302] applies.

8. References

8.1. Normative references

[I-D.ietf-dhc-dhcpv4-over-ipv6]
Lemon, T., Cui, Y., Wu, P., and J. Wu, "DHCPv4 over IPv6 Transport", draft-ietf-dhc-dhcpv4-over-ipv6-00 (work in progress), November 2011.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

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

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

8.2. Informative references

[I-D.cui-softwire-b4-translated-ds-lite]
Boucadair, M., Sun, Q., Tsou, T., Lee, Y., and Y. Cui, "Lightweight 4over6: An Extension to DS-Lite Architecture", draft-cui-softwire-b4-translated-ds-lite-05 (work in progress), February 2012.

[I-D.ietf-pcp-base]
Cheshire, S., Boucadair, M., Selkirk, P., Wing, D., and R. Penno, "Port Control Protocol (PCP)", draft-ietf-pcp-base-23 (work in progress), February 2012.

[I-D.ietf-softwire-public-4over6]
Cui, Y., Wu, J., Wu, P., Metz, C., Vautrin, O., and Y. Lee, "Public IPv4 over Access IPv6 Network", draft-ietf-softwire-public-4over6-00 (work in progress), September 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.

[RFC6346] Bush, R., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, August 2011.

Authors' Addresses

Reinaldo Penno
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089-1206
USA

Email: rpenno@juniper.net

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

Email: adurand@juniper.net

Alex Clauberg
Deutsche Telekom AG
GTN-FM4
Landgrabenweg 151
Bonn, CA 53227
Germany

Email: axel.clauberg@telekom.de

Lionel Hoffmann
Bouygues Telecom
TECHNOPOLE
13/15 Avenue du Marechal Juin
Meudon 92360
France

Email: lhoffman@bouyguestelecom.fr

Network Working Group
Internet-Draft
Obsoletes: 4008 (if approved)
Intended status: Standards Track
Expires: March 8, 2012

S. Perreault
Viagenie
T. Tsou
Huawei Technologies
September 5, 2011

Definitions of Managed Objects for Network Address Translators (NAT)
draft-perreault-opsawg-natmib-bis-00

Abstract

This memo defines a portion of the Management Information Base (MIB) for devices implementing Network Address Translator (NAT) function. This MIB module may be used for configuration as well as monitoring of a device capable of NAT function. This memo is a revision of the previous NAT-MIB [RFC4008] to take into account new types of NAT.

Status of This Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

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

This Internet-Draft will expire on March 8, 2012.

Copyright Notice

Table of Contents

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for devices implementing NAT function. This MIB module may be used for configuration and monitoring of a device capable of NAT function. NAT types and their characteristics are defined in [RFC2663]. Traditional NAT function, in particular is defined in [RFC3022]. This MIB does not address the firewall functions and must not be used for configuring or monitoring these. Section 3 provides references to the SNMP management framework, which was used as the basis for the MIB module definition. Section 4 describes the terms used throughout the document. Section 5 provides an overview of the key objects, their inter-relationship, and how the MIB module may be used to configure and monitor a NAT device. Lastly, Section 6 has the complete NAT MIB definition.

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. Changes from RFC4008

TODO: Move this section to an appendix after initial reviews.

- Address pools can now be shared between multiple interfaces. This change makes this MIB applicable to DS-Lite's AFTR [RFC6333]. See [draft-schoenw-behave-nat-mib-bis-00] for rationale.

- TODO: Merge CGN stuff from draft-jpdionne-behave-cgn-mib.

- TODO: Merge NAT64 stuff from draft-jpdionne-behave-nat64-mib.

- TODO: Update to RFC 4787 terminology for describing NAT behavior.

- TODO: Support protocols other than UDP and TCP.

- TODO: Add support to limit and/or throttle binding allocations.

- TODO: Clarify existing notifications (e.g., natPacketDiscard) and add any additional notifications that may be needed for binding limits / binding throttling.

- TODO: Are we missing anything for PCP support? (time-limited static entries)

- TODO: Include (for example in an appendix) a description plus examples how the revised NAT-MIB can be used by NAT64 implementations, CGNs, and DS- Lite implementations.

3. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

4. Terminology

[To be Reviewed]

Definitions for a majority of the terms used throughout the document may be found in [RFC2663]. Additional terms that further classify NAPT implementations are defined in [RFC3489]. Listed below are terms used in this document.

Address realm - An address realm is a realm of unique network addresses that are routable within the realm. For example, an enterprise address realm could be constituted of private IP addresses in the ranges specified in [RFC1918], which are routable within the enterprise, but not across the Internet. A public realm is constituted of globally unique network addresses.

Symmetric NAT - Symmetric NAT, as defined in [RFC3489], is a variation of Network Address Port Translator (NAPT). Symmetric NAT does not use port bind for translation across all sessions originating from the same private host. Instead, it assigns a new public port to each new session, irrespective of whether the new session used the same private end-point as before.

Bind or Binding - Several variations of the term ’Bind’ (or ’Binding’) are used throughout the document. Address Bind (or Address Binding) is a tuple of (Private IP address, Public IP Address) used for translating an IP address end-point in IP packets. Port Bind (or, Port Binding, or Address Port Bind, or Address Port Binding) is a tuple of (transport protocol, Private IP address, Private port, Public IP Address, Public port) used for translating a port end-point tuple of (transport protocol, IP address, port). Bind is used to refer to either Address Bind or Port Bind. Bind Mode identifies whether a bind is Address Bind or Port Bind.

NAT Session - A NAT session is an association between a session as seen in the private realm and a session as seen in the public realm, by virtue of NAT translation. If a session in the private realm were to be represented as (PrivateSrcAddr, PrivateDstAddr, TransportProtocol, PrivateSrcPort, PrivateDstPort) and the same session in the public realm were to be represented as (PublicSrcAddr, PublicDstAddr, TransportProtocol, PublicSrcPort, PublicDstPort), the NAT session will provide the translation glue between the two session representations. NAT sessions in the document are restricted to sessions based on TCP and UDP only. In the future, NAT sessions may be extended to be based on other transport protocols such as SCTP, UDP-lite and DCCP.

The terms 'local' and 'private' are used interchangeably throughout the document when referring to private networks, IP addresses, and ports. Likewise, the terms 'global' and 'public' are used interchangeably when referring to public networks, IP addresses, and ports.

5. Overview

NAT MIB is configurable on a per-interface basis and depends in several parts on the IF-MIB [RFC2863].

NAT MIB requires that an interface for which NAT is configured be connected to either a private or a public realm. The realm association of the interface plays an important role in the definition of address maps for the interface. An address map entry identifies the orientation of the session (inbound or outbound to the interface) for which the entry may be used for NAT translation. The address map entry also identifies the end-point of the session that must be subject to translation. An SNMP Textual-Convention 'NatTranslationEntity' is defined to capture this important characteristic that combines session orientation and applicable session endpoint for translation.

An address map may consist of static or dynamic entries. NAT creates static binds from a static address map entry. Each static bind has a direct one-to-one relationship with a static address map entry. NAT creates dynamic binds from a dynamic address map entry upon seeing the first packet of a new session.

The following subsections define the key objects used in NAT MIB, their inter-relationship, and how to configure a NAT device using the MIB module.

5.1. natInterfaceTable

[To be reviewed]

natInterfaceTable is defined in the MIB module to configure interface specific realm type and the NAT services enabled for the interface. natInterfaceTable is indexed by ifIndex and also includes interface specific NAT statistics.

The first step for an operator in configuring a NAT device is determining the interface over which NAT service is to be configured. When NAT service is operational, translated packets traverse the NAT device by ingressing on a private interface and egressing on a public interface or vice versa. An operator may configure the NAT service on either the public interface or the private interface in the traversal path.

As the next step, the operator must identify the NAT service(s) desired for the interface. The operator may configure one or more NAT services on the same interface. The MIB module identifies four types of NAT services: Basic NAT, NAPT, twice NAT and bidirectional NAT. These are NAT varieties as defined in [RFC2663]. Note that [RFC3489] further classifies NAPT implementations based on the behavior exhibited by the NAPT devices from different vendors. However, the MIB module does not explicitly distinguish between the NAPT implementations. NAPT implementations may be distinguished between one another by monitoring the BIND and NAT Session objects generated by the NAT device as described in section Section 5.6.

5.2. natAddrMapTable

[To be reviewed]

natAddrMapTable is defined in the MIB module to configure address maps on a per-interface basis. natAddrMapTable is indexed by the tuple of (ifIndex, natAddrMapIndex). The same table is also used to collect Statistics for the address map entries. Address maps are key to NAT configuration. An operator may configure one or more address map entries per interface. NAT looks up address map entries in the order in which they are defined to determine the translation function at the start of each new session traversing the interface. An address map may consist of static or dynamic entries. A static address map entry has a direct one-to-one relationship with binds. NAT will dynamically create binds from a dynamic address map entry.

The operator must be careful in selecting address map entries for an interface based on the interface realm-type and the type of NAT service desired. The operator can be amiss in the selection of

address map entries when not paying attention to the associated interface characteristics defined in natInterfaceTable (described in section 4.1). For example, say the operator wishes to configure a NAPT map entry on an interface of a NAT device. If the operator chooses to configure the NAPT map entry on a public interface (i.e., interface realm-type is public), the operator should set the TranslationEntity of the NAPT address map entry to be outboundSrcEndPoint. On the other hand, if the operator chooses to configure the NAPT map entry on a private interface (i.e., interface realm-type is private), the operator should set the TranslationEntity of the NAPT address map entry to be InboundSrcEndPoint.

5.3. Default Timeouts, Protocol Table, and Other Scalars

[To be reviewed]

DefTimeouts is defined in the MIB module to configure idle Bind timeout and IP protocol specific idle NAT session timeouts. The timeouts defined are global to the system and are not interface specific.

Protocol specific statistics are maintained in natProtocolTable, which is indexed by the protocol type.

The scalars natAddrBindNumberOfEntries and natAddrPortBindNumberOfEntries hold the number of entries that currently exist in the Address Bind and the Address Port Bind tables, respectively.

The generation of natPacketDiscard notifications can be configured by using the natNotifThrottlingInterval scalar MIB object.

5.4. natAddrBindTable and natAddrPortBindTable

[To be reviewed]

Two Bind tables, natAddrBindTable and natAddrPortBindTable, are defined to hold the bind entries. Entries are derived from the address map table and are not configurable. natAddrBindTable contains Address Binds, and natAddrPortBindTable contains Address Port Binds. natAddrBindTable is indexed by the tuple of (ifIndex, LocalAddrType, LocalAddr). natAddrPortBindTable is indexed by the tuple of (ifIndex, LocalAddrType, LocalAddr, LocalPort, Protocol). These tables also maintain bind specific statistics. A Symmetric NAT will have no entries in the Bind tables.

5.5. natSessionTable

[To be reviewed]

natSessionTable is defined to hold NAT session entries. NAT session entries are derived from NAT Binds (except in the case of Symmetric NAT) and are not configurable.

The NAT session provides the necessary translation glue between two session representations of the same end-to-end session; that is, a session as seen in the private realm and in the public realm. Session orientation (inbound or outbound) is determined from the orientation of the first packet traversing the NAT interface. Address map entries and bind entries on the interface determine whether a session is subject to NAT translation. One or both endpoints of a session may be subject to translation.

With the exception of symmetric NAT, all other NAT functions use end-point specific bind to perform individual end-point translations. Multiple NAT sessions would use the same bind as long as they share the same endpoint. Symmetric NAT does not retain a consistent port bind across multiple sessions using the same endpoint. For this reason, the bind identifier for a NAT session in symmetric NAT is set to zero. natSessionTable is indexed by the tuple of (ifIndex, natSessionIndex). Statistics for NAT sessions are also maintained in the same table.

5.6. RFC 3489 NAPT Variations, NAT Session and Bind Tables

[To be reviewed, translate to new terminology]

[RFC3489] defines four variations of NAPT - Full Cone, Restricted Cone, Port Restricted Cone, and Symmetric NAT. These can be differentiated in the NAT MIB based on different values for the objects in the session and the bind tables, as indicated below.

In a Port Restricted Cone NAT, NAT Session objects will contain a non-zero PrivateSrcEPBindId object. Further, all address and port objects within a NAT session will have non-zero values (i.e., no wildcard matches).

An Address Restricted Cone NAT may have been implemented in the same way as a Port Restricted Cone NAT, except that the UDP NAT Sessions may use ANY match on PrivateDstPort and PublicDstPort objects; i.e., PrivateDstPort and PublicDstPort objects within a NAT session may be set to zero.

A Full Cone NAT may have also been implemented in the same way as a

Port Restricted Cone NAT, except that the UDP NAT Sessions may use ANY match on PrivateDstAddr, PrivateDstPort, PublicDstAddr, and PublicDstPort objects. Within a NAT Session, all four of these objects may be set to zero. Alternately, all address and port objects within a NAT Session may have non-zero values, yet the TranslationEntity of the PrivateSrcEPBindId for the NAT Sessions may be set bi-directionally, i.e., as a bit mask of (outboundSrcEndPoint and inboundDstEndPoint) or (inboundSrcEndPoint and outboundDstEndPoint), depending on the interface realm type. Lastly, a Symmetric NAT does not maintain Port Bindings. As such, the NAT Session objects will have the PrivateSrcEPBindId set to zero.

5.7. Notifications

[To be reviewed]

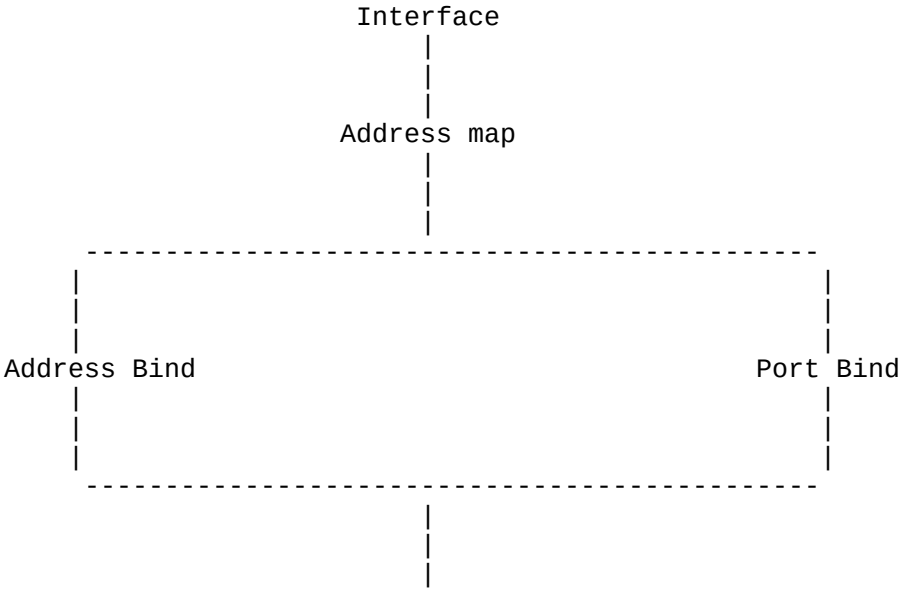
natPacketDiscard notifies the end user/manager of packets being discarded due to lack of address mappings.

[Port exhaustion, CGN-MIB?]

5.8. Notifications

[To be reviewed]

The association between the various NAT tables can be represented as follows:



NAT Session

All NAT functions, with the exception of Symmetric NAT, use Bind(s) to provide the glue necessary for a NAT Session. natSessionPrivateSrcEPBindId and natSessionPrivateDstEPBindId objects represent the endpoint Binds used by NAT Sessions.

5.9. Configuration via the MIB

[To be reviewed]

Section 5.1, and Section 5.2 and part of Section 5.3 refer to objects that are configurable on a NAT device. NAT derives Address Bind and Address Port Bind entries from the Address Map table. Hence, an Address Bind or an Address Port Bind entry must not exist without an associated entry in the Address Map table.

Further, NAT derives NAT session entries from NAT Binds, except in the case of symmetric NAT, which derives translation parameters for a NAT session directly from an address map entry. Hence, with the exception of Symmetric NAT, a NAT session entry must not exist in the NAT Session table without a corresponding bind.

A Management station may use the following steps to configure entries in the NAT-MIB:

- Create an entry in the natInterfaceTable specifying the value of ifIndex as the interface index of the interface on which NAT is being configured. Specify appropriate values, as applicable, for the other objects (e.g., natInterfaceRealm, natInterfaceServiceType) in the table (refer to Section 5.1).

- Create one or more address map entries sequentially in reduced order of priority in the natAddrMapTable, specifying the value of ifIndex to be the same for all entries. The ifIndex specified would be the same as that specified for natInterfaceTable (refer to Section 5.2).

- Configure the maximum permitted idle time duration for BINDs and TCP, UDP, and ICMP protocol sessions by setting the relevant scalars in natDefTimeouts object (refer to Section 5.3).

5.10. Relationship to Interface MIB

[To be reviewed, relationship to other MIB?]

The natInterfaceTable specifies the NAT configuration attributes on each interface. The concept of "interface" is as defined by

InterfaceIndex/ifIndex of the IETF Interfaces MIB [RFC2863].

6. Definitions

This MIB module IMPORTs objects from [RFC2578], [RFC2579], [RFC2580], [RFC2863], [RFC3411], and [RFC4001]. It also refers to information in [RFC0792], [RFC2463], and [RFC3413].

```
NAT-MIB DEFINITIONS ::= BEGIN

IMPORTS
     MODULE-IDENTITY,
     OBJECT-TYPE,
     Integer32,
     Unsigned32,
     Gauge32,
     Counter64,
     TimeTicks,
     mib-2,
     NOTIFICATION-TYPE
             FROM SNMPv2-SMI
     TEXTUAL-CONVENTION,
     StorageType,
     RowStatus
             FROM SNMPv2-TC
     MODULE-COMPLIANCE,
     NOTIFICATION-GROUP,
     OBJECT-GROUP
             FROM SNMPv2-CONF
     ifIndex,
     ifCounterDiscontinuityGroup
             FROM IF-MIB
     SnmpAdminString
             FROM SNMP-FRAMEWORK-MIB
     InetAddressType,
     InetAddress,
     InetPortNumber
             FROM INET-ADDRESS-MIB;

natMIB MODULE-IDENTITY
     LAST-UPDATED "YYYYMMDDhhmmZ"
     ORGANIZATION "IETF Transport Area"
     CONTACT-INFO
              "
               Simon Perreault
               Viagenie
               2875 boul. Laurier, suite D2-630
               Quebec
               Canada
```

```
               Phone: +1-418-656-9254
               EMail: simon.perreault@viagenie.ca

               Tina Tsou
               Huawei Technologies
               2330 Central Expressway
               Santa Clara
               USA
               Phone: +1-408-330-4424
               EMail: tena@huawei.com
              "
     DESCRIPTION
             "This MIB module defines the generic managed objects
              for NAT.

              Copyright (C) The Internet Society (YYYY).  This version
              of this MIB module is part of RFC XXXX;  see the RFC
              itself for full legal notices."
     REVISION     "200503210000Z"  -- 21th March 2005
     DESCRIPTION
             "Initial version, published as RFC 4008."
     REVISION     "YYYYMMDDhhmmZ"
     DESCRIPTION
             "Second version, published as RFC XXXX."

     ::= { mib-2 123 }

natMIBObjects OBJECT IDENTIFIER ::= { natMIB 1 }

NatProtocolType ::= TEXTUAL-CONVENTION
       STATUS       current
       DESCRIPTION
               "A list of protocols that support the network
                address translation.  Inclusion of the values is
                not intended to imply that those protocols
                need to be supported.  Any change in this
                TEXTUAL-CONVENTION should also be reflected in
                the definition of NatProtocolMap, which is a
                BITS representation of this."
       SYNTAX   INTEGER {
                    none (1),  -- not specified
                    other (2), -- none of the following
                    icmp (3),
                    udp (4),
                    tcp (5)
                  }

NatProtocolMap ::= TEXTUAL-CONVENTION
```

```
        STATUS       current
        DESCRIPTION
                "A bitmap of protocol identifiers that support
                 the network address translation.  Any change
                 in this TEXTUAL-CONVENTION should also be
                 reflected in the definition of NatProtocolType."
        SYNTAX   BITS {
                   other (0),
                   icmp (1),
                   udp (2),
                   tcp (3)
                 }

NatAddrMapId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each address map
                 by a NAT enabled device."
        SYNTAX   Unsigned32 (1..4294967295)

NatSharedAddrMapId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each shared address
                 map by a NAT enabled device."
        SYNTAX   Unsigned32 (1..4294967295)

NatBindIdOrZero ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each bind by
                 a NAT enabled device.  The bind id will be zero
                 in the case of a Symmetric NAT."
        SYNTAX   Unsigned32 (0..4294967295)

NatBindId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each bind by
                 a NAT enabled device."
        SYNTAX   Unsigned32 (1..4294967295)

NatSessionId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
```

```
       STATUS current
       DESCRIPTION
               "A unique id that is assigned to each session by
                a NAT enabled device."
       SYNTAX   Unsigned32 (1..4294967295)

NatBindMode ::= TEXTUAL-CONVENTION
       STATUS current
       DESCRIPTION
               "An indication of whether the bind is
                an address bind or an address port bind."
       SYNTAX   INTEGER {
                    addressBind (1),
                    addressPortBind (2)
                }

NatAssociationType ::= TEXTUAL-CONVENTION
       STATUS current
       DESCRIPTION
               "An indication of whether the association is
                static or dynamic."
       SYNTAX   INTEGER {
                    static (1),
                    dynamic (2)
                }

NatTranslationEntity ::= TEXTUAL-CONVENTION
       STATUS       current
       DESCRIPTION
               "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."
       SYNTAX   BITS {
                  inboundSrcEndPoint (0),
                  outboundDstEndPoint(1),
                  inboundDstEndPoint (2),
                  outboundSrcEndPoint(3)
                }

--
-- Default Values for the Bind and NAT Protocol Timers
--

natDefTimeouts OBJECT IDENTIFIER ::= { natMIBObjects 1 }

natNotifCtrl OBJECT IDENTIFIER ::= { natMIBObjects 2 }
```

```
--
-- Address Bind and Port Bind related NAT configuration
--

natBindDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (0..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default Bind (Address Bind or Port Bind) idle
             timeout parameter.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 0 }
    ::= { natDefTimeouts 1 }

--
-- UDP related NAT configuration
--

natUdpDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default UDP idle timeout parameter.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 300 }
    ::= { natDefTimeouts 2 }

--
-- ICMP related NAT configuration
--

natIcmpDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
```

```
    DESCRIPTION
            "The default ICMP idle timeout parameter.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 300 }
    ::= { natDefTimeouts 3 }

--
-- Other protocol parameters
--

natOtherDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default idle timeout parameter for protocols
             represented by the value other (2) in
             NatProtocolType.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 60 }
    ::= { natDefTimeouts 4 }

--
-- TCP related NAT Timers
--

natTcpDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default time interval that a NAT session for an
             established TCP connection is allowed to remain
             valid without any activity on the TCP connection.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
```

```
             system."
    DEFVAL { 86400 }
    ::= { natDefTimeouts 5 }

natTcpDefNegTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default time interval that a NAT session for a TCP
             connection that is not in the established state
             is allowed to remain valid without any activity on
             the TCP connection.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 60 }
    ::= { natDefTimeouts 6 }

natNotifThrottlingInterval OBJECT-TYPE
    SYNTAX      Integer32 (0 | 5..3600)
    UNITS       "seconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "This object controls the generation of the
             natPacketDiscard notification.

             If this object has a value of zero, then no
             natPacketDiscard notifications will be transmitted by the
             agent.

             If this object has a non-zero value, then the agent must
             not generate more than one natPacketDiscard
             'notification-event' in the indicated period, where a
             'notification-event' is the generation of a single
             notification PDU type to a list of notification
             destinations.  If additional NAT packets are discarded
             within the throttling period, then notification-events
             for these changes must be suppressed by the agent until
             the current throttling period expires.

             If natNotifThrottlingInterval notification generation
             is enabled, the suggested default throttling period is
             60 seconds, but generation of the natPacketDiscard
```

```
             notification should be disabled by default.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system.

             The actual transmission of notifications is controlled
             via the MIB modules in RFC 3413."
    DEFVAL { 0 }
    ::= { natNotifCtrl 1 }

--
-- The NAT Interface Table
--

natInterfaceTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatInterfaceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This table specifies the attributes for interfaces on a
             device supporting NAT function."
    ::= { natMIBObjects 3 }

natInterfaceEntry OBJECT-TYPE
    SYNTAX      NatInterfaceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Each entry in the natInterfaceTable holds a set of
             parameters for an interface, instantiated by
             ifIndex.  Therefore, the interface index must have been
             assigned, according to the applicable procedures,
             before it can be meaningfully used.
             Generally, this means that the interface must exist.

             When natStorageType is of type nonVolatile, however,
             this may reflect the configuration for an interface whose
             ifIndex has been assigned but for which the supporting
             implementation is not currently present."
    INDEX   { ifIndex }
    ::= { natInterfaceTable 1 }

NatInterfaceEntry ::= SEQUENCE {
    natInterfaceRealm               INTEGER,
    natInterfaceServiceType         BITS,
    natInterfaceInTranslates        Counter64,
```

```
    natInterfaceOutTranslates       Counter64,
    natInterfaceDiscards            Counter64,
    natInterfaceStorageType         StorageType,
    natInterfaceRowStatus           RowStatus,
    natInterfaceSharedAddrMapIndex  NatSharedAddrMapId
}

natInterfaceRealm OBJECT-TYPE
    SYNTAX     INTEGER {
                   private (1),
                   public (2)
               }
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
            "This object identifies whether this interface is
             connected to the private or the public realm."
    DEFVAL  { public }
    ::= { natInterfaceEntry 1 }

natInterfaceServiceType OBJECT-TYPE
    SYNTAX  BITS {
                basicNat (0),
                napt (1),
                bidirectionalNat (2),
                twiceNat (3)
            }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "An indication of the direction in which new sessions
             are permitted and the extent of translation done within
             the IP and transport headers."
    ::= { natInterfaceEntry 2 }

natInterfaceInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of packets received on this interface that
             were translated.
             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natInterfaceEntry 3 }
```

```
natInterfaceOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of translated packets that were sent out this
             interface.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natInterfaceEntry 4 }

natInterfaceDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of packets that had to be rejected/dropped due to
             a lack of resources for this interface.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
     ::= { natInterfaceEntry 5 }

natInterfaceStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row.
             Conceptual rows having the value 'permanent'
             need not allow write-access to any columnar objects
             in the row."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    DEFVAL { nonVolatile }
    ::= { natInterfaceEntry 6 }

natInterfaceRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row.
```

```
             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the natInterfaceRowStatus
             column is 'notReady'.


             In particular, a newly created row cannot be made
             active until the corresponding instance of
             natInterfaceServiceType has been set.

             None of the objects in this row may be modified
             while the value of this object is active(1)."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    ::= { natInterfaceEntry 7 }

natInterfaceSharedAddrMapIndex OBJECT-TYPE
    SYNTAX      NatSharedAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Link to a NatSharedAddrMapEntry.  If NULL,
             it is expected that there exist at least one
             NatAddrMapEntry pointing to this interface entry."
    ::= { natInterfaceEntry 8 }


--
-- The Address Map Table
--

natAddrMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This table lists address map parameters for NAT."
    ::= { natMIBObjects 4 }

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
}

natAddrMapIndex  OBJECT-TYPE
    SYNTAX      NatAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Along with ifIndex, this object uniquely
             identifies an entry in the natAddrMapTable.
             Address map entries are applied in the order
             specified by natAddrMapIndex."
    ::= { natAddrMapEntry 1 }

natAddrMapName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(1..32))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "Name identifying all map entries in the table associated
             with the same interface.  All map entries with the same
             ifIndex MUST have the same map name."
    ::= { natAddrMapEntry 2 }

natAddrMapEntryType OBJECT-TYPE
```

```
    SYNTAX      NatAssociationType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This parameter can be used to set up static
             or dynamic address maps."
    ::= { natAddrMapEntry 3 }

natAddrMapTranslationEntity OBJECT-TYPE
    SYNTAX      NatTranslationEntity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The end-point entity (source or destination) in
             inbound or outbound sessions (i.e., first packets) that
             may be translated by an address map entry.

             Session direction (inbound or outbound) is
             derived from the direction of the first packet
             of a session traversing a NAT interface.
             NAT address (and Transport-ID) maps may be defined
             to effect inbound or outbound sessions.

             Traditionally, address maps for Basic NAT and NAPT are
             configured on a public interface for outbound sessions,
             effecting translation of source end-point.  The value of
             this object must be set to outboundSrcEndPoint for
             those interfaces.

             Alternately, if address maps for Basic NAT and NAPT were
             to be configured on a private interface, the desired
             value for this object for the map entries
             would be inboundSrcEndPoint (i.e., effecting translation
             of source end-point for inbound sessions).

             If TwiceNAT were to be configured on a private interface,
             the desired value for this object for the map entries
             would be a bitmask of inboundSrcEndPoint and
             inboundDstEndPoint."
    ::= { natAddrMapEntry 4 }

natAddrMapLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo."
```

```
    ::= { natAddrMapEntry 5 }

natAddrMapLocalAddrFrom OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range
             of IP addresses mapped by this translation entry.  The
             value of this object must be less than or equal to the
             value of the natAddrMapLocalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natAddrMapEntry 6 }

natAddrMapLocalAddrTo OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses mapped by this translation entry.  If only
             a single address is being mapped, the value of this object
             is equal to the value of natAddrMapLocalAddrFrom.  For a
             static NAT, the number of addresses in the range defined
             by natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo must
             be equal to the number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natAddrMapEntry 7 }

natAddrMapLocalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped.

             The value of this object must be less than or equal to the
```

```
             value of the natAddrMapLocalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortTo."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 8 }

natAddrMapLocalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the last port number in the range
             of ports being mapped.

             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalPortFrom object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortFrom."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 9 }

natAddrMapGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo."
    ::= { natAddrMapEntry 10 }

natAddrMapGlobalAddrFrom OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range of
             IP addresses being mapped to.  The value of this object
             must be less than or equal to the value of the
             natAddrMapGlobalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natAddrMapEntry 11 }

natAddrMapGlobalAddrTo OBJECT-TYPE
```

```
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses being mapped to.  If only a single address is
             being mapped to, the value of this object is equal to the
             value of natAddrMapGlobalAddrFrom.  For a static NAT, the
             number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo must be
             equal to the number of addresses in the range defined by
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natAddrMapEntry 12 }

natAddrMapGlobalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped to.


             The value of this object must be less than or equal to the
             value of the natAddrMapGlobalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value natAddrMapGlobalPortTo."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 13 }

natAddrMapGlobalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of this
             object specifies the last port number in the range of
             ports being mapped to.
```

```
             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalPortFrom object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapGlobalPortFrom."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 14 }

natAddrMapProtocol OBJECT-TYPE
    SYNTAX      NatProtocolMap
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies a bitmap of protocol identifiers."
    ::= { natAddrMapEntry 15 }

natAddrMapInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets pertaining to this address
             map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrMapEntry 16 }

natAddrMapOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets pertaining to this
             address map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrMapEntry 17 }

natAddrMapDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
```

```
            "The number of packets pertaining to this address map
             entry that were dropped due to lack of addresses in the
             address pool identified by this address map.  The value of
             this object must always be zero in case of static
             address map.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrMapEntry 18 }

natAddrMapAddrUsed OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of addresses pertaining to this address map
             that are currently being used from the NAT pool.
             The value of this object must always be zero in the case
             of a static address map."
    ::= { natAddrMapEntry 19 }

natAddrMapStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row.
             Conceptual rows having the value 'permanent'
             need not allow write-access to any columnar objects
             in the row."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    DEFVAL { nonVolatile }
    ::= { natAddrMapEntry 20 }

natAddrMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row.

             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the natAddrMapRowStatus
             column is 'notReady'.
```

```
             None of the objects in this row may be modified
             while the value of this object is active(1)."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    ::= { natAddrMapEntry 21 }


--
-- Address Bind section
--

natAddrBindNumberOfEntries OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object maintains a count of the number of entries
             that currently exist in the natAddrBindTable."
    ::= { natMIBObjects 5 }

--
-- The NAT Address BIND Table
--

natAddrBindTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatAddrBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This table holds information about the currently
             active NAT BINDs."
    ::= { natMIBObjects 6 }

natAddrBindEntry OBJECT-TYPE
    SYNTAX     NatAddrBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "Each entry in this table holds information about
             an active address BIND.  These entries are lost
             upon agent restart.

             This row has indexing which may create variables with
             more than 128 subidentifiers.  Implementers of this table
             must be careful not to create entries that would result
             in OIDs which exceed the 128 subidentifier limit.
             Otherwise, the information cannot be accessed using
             SNMPv1, SNMPv2c or SNMPv3."
```

```
    INDEX   { ifIndex, natAddrBindLocalAddrType, natAddrBindLocalAddr }
    ::= { natAddrBindTable 1 }

NatAddrBindEntry ::= SEQUENCE {
    natAddrBindLocalAddrType        InetAddressType,
    natAddrBindLocalAddr            InetAddress,
    natAddrBindGlobalAddrType       InetAddressType,
    natAddrBindGlobalAddr           InetAddress,
    natAddrBindId                   NatBindId,
    natAddrBindTranslationEntity    NatTranslationEntity,
    natAddrBindType                 NatAssociationType,
    natAddrBindMapIndex             NatAddrMapId,
    natAddrBindSessions             Gauge32,
    natAddrBindMaxIdleTime          TimeTicks,
    natAddrBindCurrentIdleTime      TimeTicks,
    natAddrBindInTranslates         Counter64,
    natAddrBindOutTranslates        Counter64
}

natAddrBindLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrBindLocalAddr."
    ::= { natAddrBindEntry 1 }

natAddrBindLocalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This object represents the private-realm specific network
             layer address, which maps to the public-realm address
             represented by natAddrBindGlobalAddr.

             The type of this address is determined by the value of
             the natAddrBindLocalAddrType object."
   ::= { natAddrBindEntry 2 }

natAddrBindGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrBindGlobalAddr."
```

```
    ::= { natAddrBindEntry 3 }

natAddrBindGlobalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the public-realm network layer
             address that maps to the private-realm network layer
             address represented by natAddrBindLocalAddr.

             The type of this address is determined by the value of
             the natAddrBindGlobalAddrType object."
    ::= { natAddrBindEntry 4 }

natAddrBindId OBJECT-TYPE
    SYNTAX     NatBindId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents a bind id that is dynamically
             assigned to each bind by a NAT enabled device.  Each
             bind is represented by a bind id that is
             unique across both, the natAddrBindTable and the
             natAddrPortBindTable."
    ::= { natAddrBindEntry 5 }

natAddrBindTranslationEntity OBJECT-TYPE
    SYNTAX     NatTranslationEntity
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the direction of sessions
             for which this bind is applicable and the endpoint entity
             (source or destination) within the sessions that is
             subject to translation using the BIND.

             Orientation of the bind can be a superset of
             translationEntity of the address map entry which
             forms the basis for this bind.

             For example, if the translationEntity of an
             address map entry is outboundSrcEndPoint, the
             translationEntity of a bind derived from this
             map entry may either be outboundSrcEndPoint or
             it may be bidirectional (a bitmask of
             outboundSrcEndPoint and inboundDstEndPoint)."
    ::= { natAddrBindEntry 6 }
```

```
natAddrBindType OBJECT-TYPE
    SYNTAX     NatAssociationType
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind is static or
             dynamic."
    ::= { natAddrBindEntry 7 }

natAddrBindMapIndex OBJECT-TYPE
    SYNTAX     NatAddrMapId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object is a pointer to the natAddrMapTable entry
             (and the parameters of that entry) which was used in
             creating this BIND.  This object, in conjunction with the
             ifIndex (which identifies a unique addrMapName) points to
             a unique entry in the natAddrMapTable."
    ::= { natAddrBindEntry 8 }

natAddrBindSessions OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of sessions currently using this BIND."
    ::= { natAddrBindEntry 9 }

natAddrBindMaxIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates the maximum time for
             which this bind can be idle with no sessions
             attached to it.

             The value of this object is of relevance only for
             dynamic NAT."
    ::= { natAddrBindEntry 10 }

natAddrBindCurrentIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "At any given instance, this object indicates the
```

```
              time that this bind has been idle without any sessions
              attached to it.

              The value of this object is of relevance only for
              dynamic NAT."
    ::= { natAddrBindEntry 11 }

natAddrBindInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets that were successfully
             translated by using this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrBindEntry 12 }

natAddrBindOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets that were successfully
             translated using this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrBindEntry 13 }

--
-- Address Port Bind section
--

natAddrPortBindNumberOfEntries OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object maintains a count of the number of entries
             that currently exist in the natAddrPortBindTable."
    ::= { natMIBObjects 7 }
```

```
--
-- The NAT Address Port Bind Table
--

natAddrPortBindTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatAddrPortBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This table holds information about the currently
             active NAPT BINDs."
    ::= { natMIBObjects 8 }

natAddrPortBindEntry OBJECT-TYPE
    SYNTAX     NatAddrPortBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "Each entry in the this table holds information
             about a NAPT bind that is currently active.
             These entries are lost upon agent restart.

             This row has indexing which may create variables with
             more than 128 subidentifiers.  Implementers of this table
             must be careful not to create entries which would result
             in OIDs that exceed the 128 subidentifier limit.
             Otherwise, the information cannot be accessed using
             SNMPv1, SNMPv2c or SNMPv3."
    INDEX   { ifIndex, natAddrPortBindLocalAddrType,
              natAddrPortBindLocalAddr, natAddrPortBindLocalPort,
              natAddrPortBindProtocol }
    ::= { natAddrPortBindTable 1 }

NatAddrPortBindEntry ::= SEQUENCE {
    natAddrPortBindLocalAddrType        InetAddressType,
    natAddrPortBindLocalAddr            InetAddress,
    natAddrPortBindLocalPort            InetPortNumber,
    natAddrPortBindProtocol             NatProtocolType,
    natAddrPortBindGlobalAddrType       InetAddressType,
    natAddrPortBindGlobalAddr           InetAddress,
    natAddrPortBindGlobalPort           InetPortNumber,
    natAddrPortBindId                   NatBindId,
    natAddrPortBindTranslationEntity    NatTranslationEntity,
    natAddrPortBindType                 NatAssociationType,
    natAddrPortBindMapIndex             NatAddrMapId,
    natAddrPortBindSessions             Gauge32,
    natAddrPortBindMaxIdleTime          TimeTicks,
    natAddrPortBindCurrentIdleTime      TimeTicks,
```

```
    natAddrPortBindInTranslates         Counter64,
    natAddrPortBindOutTranslates        Counter64
}

natAddrPortBindLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrPortBindLocalAddr."
    ::= { natAddrPortBindEntry 1 }

natAddrPortBindLocalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This object represents the private-realm specific network
             layer address which, in conjunction with
             natAddrPortBindLocalPort, maps to the public-realm
             network layer address and transport id represented by
             natAddrPortBindGlobalAddr and natAddrPortBindGlobalPort
             respectively.


             The type of this address is determined by the value of
             the natAddrPortBindLocalAddrType object."
    ::= { natAddrPortBindEntry 2 }

natAddrPortBindLocalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "For a protocol value TCP or UDP, this object represents
             the private-realm specific port number.  On the other
             hand, for ICMP a bind is created only for query/response
             type ICMP messages such as ICMP echo, Timestamp, and
             Information request messages, and this object represents
             the private-realm specific identifier in the ICMP
             message, as defined in RFC 792 for ICMPv4 and in RFC
             2463 for ICMPv6.

             This object, together with natAddrPortBindProtocol,
             natAddrPortBindLocalAddrType, and natAddrPortBindLocalAddr,
             constitutes a session endpoint in the private realm.  A
             bind entry binds a private realm specific endpoint to a
```

```
             public realm specific endpoint, as represented by the
             tuple of (natAddrPortBindGlobalPort,
             natAddrPortBindProtocol, natAddrPortBindGlobalAddrType,
             and natAddrPortBindGlobalAddr)."
   ::= { natAddrPortBindEntry 3 }

natAddrPortBindProtocol OBJECT-TYPE
    SYNTAX      NatProtocolType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This object specifies a protocol identifier.  If the
             value of this object is none(1), then this bind entry
             applies to all IP traffic.  Any other value of this object
             specifies the class of IP traffic to which this BIND
             applies."
    ::= { natAddrPortBindEntry 4 }

natAddrPortBindGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrPortBindGlobalAddr."
    ::= { natAddrPortBindEntry 5 }

natAddrPortBindGlobalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the public-realm specific network
             layer address that, in conjunction with
             natAddrPortBindGlobalPort, maps to the private-realm

             network layer address and transport id represented by
             natAddrPortBindLocalAddr and natAddrPortBindLocalPort,
             respectively.

             The type of this address is determined by the value of
             the natAddrPortBindGlobalAddrType object."
    ::= { natAddrPortBindEntry 6 }

natAddrPortBindGlobalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
```

```
    DESCRIPTION
            "For a protocol value TCP or UDP, this object represents
             the public-realm specific port number.  On the other
             hand, for ICMP a bind is created only for query/response
             type ICMP messages such as ICMP echo, Timestamp, and
             Information request messages, and this object represents
             the public-realm specific identifier in the ICMP message,
             as defined in RFC 792 for ICMPv4 and in RFC 2463 for
             ICMPv6.

             This object, together with natAddrPortBindProtocol,
             natAddrPortBindGlobalAddrType, and
             natAddrPortBindGlobalAddr, constitutes a session endpoint
             in the public realm.  A bind entry binds a public realm
             specific endpoint to a private realm specific endpoint,
             as represented by the tuple of
              (natAddrPortBindLocalPort, natAddrPortBindProtocol,
               natAddrPortBindLocalAddrType, and
               natAddrPortBindLocalAddr)."
    ::= { natAddrPortBindEntry 7 }

natAddrPortBindId OBJECT-TYPE
    SYNTAX     NatBindId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents a bind id that is dynamically
             assigned to each bind by a NAT enabled device.  Each
             bind is represented by a unique bind id across both
             the natAddrBindTable and the natAddrPortBindTable."
    ::= { natAddrPortBindEntry 8 }

natAddrPortBindTranslationEntity OBJECT-TYPE
    SYNTAX     NatTranslationEntity
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the direction of sessions
             for which this bind is applicable and the entity
             (source or destination) within the sessions that is
             subject to translation with the BIND.

             Orientation of the bind can be a superset of the
             translationEntity of the address map entry that
             forms the basis for this bind.

             For example, if the translationEntity of an
             address map entry is outboundSrcEndPoint, the
```

```
             translationEntity of a bind derived from this
             map entry may either be outboundSrcEndPoint or
             may be bidirectional (a bitmask of
             outboundSrcEndPoint and inboundDstEndPoint)."
    ::= { natAddrPortBindEntry 9 }

natAddrPortBindType OBJECT-TYPE
    SYNTAX     NatAssociationType
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind is static or
             dynamic."
    ::= { natAddrPortBindEntry 10 }

natAddrPortBindMapIndex OBJECT-TYPE
    SYNTAX     NatAddrMapId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object is a pointer to the natAddrMapTable entry
             (and the parameters of that entry) used in
             creating this BIND.  This object, in conjunction with the
             ifIndex (which identifies a unique addrMapName), points
             to a unique entry in the natAddrMapTable."
    ::= { natAddrPortBindEntry 11 }

natAddrPortBindSessions OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of sessions currently using this BIND."
    ::= { natAddrPortBindEntry 12 }

natAddrPortBindMaxIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current

    DESCRIPTION
            "This object indicates the maximum time for
             which this bind can be idle without any sessions
             attached to it.
             The value of this object is of relevance
             only for dynamic NAT."
    ::= { natAddrPortBindEntry 13 }
```

```
natAddrPortBindCurrentIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "At any given instance, this object indicates the
             time that this bind has been idle without any sessions
             attached to it.

             The value of this object is of relevance
             only for dynamic NAT."
    ::= { natAddrPortBindEntry 14 }

natAddrPortBindInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets that were translated as per
             this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrPortBindEntry 15 }

natAddrPortBindOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets that were translated as per
             this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrPortBindEntry 16 }

--
-- The Session Table
--

natSessionTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatSessionEntry
    MAX-ACCESS not-accessible
```

```
    STATUS     current
    DESCRIPTION
            "The (conceptual) table containing one entry for each
             NAT session currently active on this NAT device."
    ::= { natMIBObjects 9 }

natSessionEntry OBJECT-TYPE
    SYNTAX     NatSessionEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry (conceptual row) containing information
             about an active NAT session on this NAT device.
             These entries are lost upon agent restart."
    INDEX   { ifIndex, natSessionIndex }
    ::= { natSessionTable 1 }

NatSessionEntry ::= SEQUENCE {
    natSessionIndex                     NatSessionId,
    natSessionPrivateSrcEPBindId        NatBindIdOrZero,
    natSessionPrivateSrcEPBindMode      NatBindMode,
    natSessionPrivateDstEPBindId        NatBindIdOrZero,
    natSessionPrivateDstEPBindMode      NatBindMode,
    natSessionDirection                 INTEGER,
    natSessionUpTime                    TimeTicks,
    natSessionAddrMapIndex              NatAddrMapId,
    natSessionProtocolType              NatProtocolType,
    natSessionPrivateAddrType           InetAddressType,
    natSessionPrivateSrcAddr            InetAddress,
    natSessionPrivateSrcPort            InetPortNumber,
    natSessionPrivateDstAddr            InetAddress,
    natSessionPrivateDstPort            InetPortNumber,
    natSessionPublicAddrType            InetAddressType,
    natSessionPublicSrcAddr             InetAddress,
    natSessionPublicSrcPort             InetPortNumber,
    natSessionPublicDstAddr             InetAddress,
    natSessionPublicDstPort             InetPortNumber,
    natSessionMaxIdleTime               TimeTicks,
    natSessionCurrentIdleTime           TimeTicks,
    natSessionInTranslates              Counter64,
    natSessionOutTranslates             Counter64
}

natSessionIndex OBJECT-TYPE
    SYNTAX     NatSessionId
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
```

```
            "The session ID for this NAT session."
    ::= { natSessionEntry 1 }

natSessionPrivateSrcEPBindId OBJECT-TYPE
    SYNTAX     NatBindIdOrZero
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The bind id associated between private and public
             source end points.  In the case of Symmetric-NAT,
             this should be set to zero."
    ::= { natSessionEntry 2 }

natSessionPrivateSrcEPBindMode OBJECT-TYPE
    SYNTAX     NatBindMode
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind indicated
             by the object natSessionPrivateSrcEPBindId
             is an address bind or an address port bind."
    ::= { natSessionEntry 3 }

natSessionPrivateDstEPBindId OBJECT-TYPE
    SYNTAX     NatBindIdOrZero
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The bind id associated between private and public
             destination end points."
    ::= { natSessionEntry 4 }

natSessionPrivateDstEPBindMode OBJECT-TYPE
    SYNTAX     NatBindMode
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind indicated
             by the object natSessionPrivateDstEPBindId
             is an address bind or an address port bind."
    ::= { natSessionEntry 5 }

natSessionDirection OBJECT-TYPE
    SYNTAX     INTEGER {
                   inbound (1),
                   outbound (2)
               }
```

```
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The direction of this session with respect to the
             local network.  'inbound' indicates that this session
             was initiated from the public network into the private
             network.  'outbound' indicates that this session was
             initiated from the private network into the public
             network."
    ::= { natSessionEntry 6 }

natSessionUpTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The up time of this session in one-hundredths of a
             second."
    ::= { natSessionEntry 7 }

natSessionAddrMapIndex OBJECT-TYPE
    SYNTAX     NatAddrMapId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object is a pointer to the natAddrMapTable entry
             (and the parameters of that entry) used in
             creating this session.  This object, in conjunction with
             the ifIndex (which identifies a unique addrMapName), points
             to a unique entry in the natAddrMapTable."
    ::= { natSessionEntry 8 }

natSessionProtocolType OBJECT-TYPE
    SYNTAX     NatProtocolType
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The protocol type of this session."
    ::= { natSessionEntry 9 }

natSessionPrivateAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natSessionPrivateSrcAddr and natSessionPrivateDstAddr."
    ::= { natSessionEntry 10 }
```

```
natSessionPrivateSrcAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The source IP address of the session endpoint that
             lies in the private network.

             The value of this object must be zero only when the
             natSessionPrivateSrcEPBindId object has a zero value.
             When the value of this object is zero, the NAT session
             lookup will match any IP address to this field.

             The type of this address is determined by the value of
             the natSessionPrivateAddrType object."
    ::= { natSessionEntry 11 }

natSessionPrivateSrcPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the source port in the first packet of session
             while in private-realm.  On the other hand, when the
             protocol is ICMP, a NAT session is created only for
             query/response type ICMP messages such as ICMP echo,
             Timestamp, and Information request messages, and this
             object represents the private-realm specific identifier
             in the ICMP message, as defined in RFC 792 for ICMPv4
             and in RFC 2463 for ICMPv6.

             The value of this object must be zero when the
             natSessionPrivateSrcEPBindId object has zero value
             and value of natSessionPrivateSrcEPBindMode is
             addressPortBind(2).  In such a case, the NAT session
             lookup will match any port number to this field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort, or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 12 }

natSessionPrivateDstAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
```

```
    DESCRIPTION
            "The destination IP address of the session endpoint that
             lies in the private network.

             The value of this object must be zero when the
             natSessionPrivateDstEPBindId object has a zero value.
             In such a scenario, the NAT session lookup will match
             any IP address to this field.

             The type of this address is determined by the value of
             the natSessionPrivateAddrType object."
    ::= { natSessionEntry 13 }

natSessionPrivateDstPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the destination port in the first packet
             of session while in private-realm.  On the other hand,
             when the protocol is ICMP, this object is not relevant
             and should be set to zero.

             The value of this object must be zero when the
             natSessionPrivateDstEPBindId object has a zero
             value and natSessionPrivateDstEPBindMode is set to
             addressPortBind(2).  In such a case, the NAT session
             lookup will match any port number to this field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort, or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 14 }

natSessionPublicAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natSessionPublicSrcAddr and natSessionPublicDstAddr."
    ::= { natSessionEntry 15 }

natSessionPublicSrcAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
```

```
    STATUS     current
    DESCRIPTION
            "The source IP address of the session endpoint that
             lies in the public network.

             The value of this object must be zero when the
             natSessionPrivateSrcEPBindId object has a zero value.
             In such a scenario, the NAT session lookup will match
             any IP address to this field.

             The type of this address is determined by the value of
             the natSessionPublicAddrType object."
    ::= { natSessionEntry 16 }

natSessionPublicSrcPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the source port in the first packet of
             session while in public-realm.  On the other hand, when
             protocol is ICMP, a NAT session is created only for
             query/response type ICMP messages such as ICMP echo,
             Timestamp, and Information request messages, and this
             object represents the public-realm specific identifier
             in the ICMP message, as defined in RFC 792 for ICMPv4
             and in RFC 2463 for ICMPv6.

             The value of this object must be zero when the
             natSessionPrivateSrcEPBindId object has a zero value
             and natSessionPrivateSrcEPBindMode is set to
             addressPortBind(2).  In such a scenario, the NAT
             session lookup will match any port number to this
             field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 17 }

natSessionPublicDstAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The destination IP address of the session endpoint that
```

```
             lies in the public network.

             The value of this object must be non-zero when the
             natSessionPrivateDstEPBindId object has a non-zero
             value.  If the value of this object and the
             corresponding natSessionPrivateDstEPBindId object value
             is zero, then the NAT session lookup will match any IP
             address to this field.

             The type of this address is determined by the value of
             the natSessionPublicAddrType object."
    ::= { natSessionEntry 18 }

natSessionPublicDstPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the destination port in the first packet of
             session while in public-realm.  On the other hand, when
             the protocol is ICMP, this object is not relevant for
             translation and should be zero.

             The value of this object must be zero when the
             natSessionPrivateDstEPBindId object has a zero value
             and natSessionPrivateDstEPBindMode is
             addressPortBind(2).  In such a scenario, the NAT
             session lookup will match any port number to this
             field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort, or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 19 }

natSessionMaxIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The max time for which this session can be idle
             without detecting a packet."
    ::= { natSessionEntry 20 }

natSessionCurrentIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
```

```
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The time since a packet belonging to this session was
            last detected."
    ::= { natSessionEntry 21 }

natSessionInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets that were translated for
             this session.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSessionEntry 22 }

natSessionOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets that were translated for
             this session.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSessionEntry 23 }

--
-- The Protocol table
--

natProtocolTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatProtocolEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The (conceptual) table containing per protocol NAT
             statistics."
    ::= { natMIBObjects 10 }
```

```
natProtocolEntry OBJECT-TYPE
    SYNTAX     NatProtocolEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry (conceptual row) containing NAT statistics
             pertaining to a particular protocol."
    INDEX   { natProtocol }
    ::= { natProtocolTable 1 }

NatProtocolEntry ::= SEQUENCE {
    natProtocol                 NatProtocolType,
    natProtocolInTranslates     Counter64,
    natProtocolOutTranslates    Counter64,
    natProtocolDiscards         Counter64
}

natProtocol    OBJECT-TYPE
    SYNTAX     NatProtocolType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This object represents the protocol pertaining to which
             parameters are reported."
    ::= { natProtocolEntry 1 }

natProtocolInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets pertaining to the protocol
             identified by natProtocol that underwent NAT.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natProtocolEntry 2 }

natProtocolOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets pertaining to the protocol
             identified by natProtocol that underwent NAT.
```

```
             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natProtocolEntry 3 }

natProtocolDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of packets pertaining to the protocol
             identified by natProtocol that had to be
             rejected/dropped due to lack of resources.  These
             rejections could be due to session timeout, resource
             unavailability, lack of address space, etc.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
     ::= { natProtocolEntry 4 }


--
-- The Shared Address Map Table
--

natSharedAddrMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatSharedAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This table lists address map parameters for NAT."
    ::= { natMIBObjects 11 }

natSharedAddrMapEntry OBJECT-TYPE
    SYNTAX      NatSharedAddrMapEntry
    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   { natSharedAddrMapIndex }
    ::= { natSharedAddrMapTable 1 }

NatSharedAddrMapEntry ::= SEQUENCE {
```

```
    natSharedAddrMapIndex               NatSharedAddrMapId,
    natSharedAddrMapName                SnmpAdminString,
    natSharedAddrMapEntryType           NatAssociationType,
    natSharedAddrMapTranslatEntity      NatTranslationEntity,
    natSharedAddrMapLocalAddrType       InetAddressType,
    natSharedAddrMapLocalAddrFrom       InetAddress,
    natSharedAddrMapLocalAddrTo         InetAddress,
    natSharedAddrMapLocalPortFrom       InetPortNumber,
    natSharedAddrMapLocalPortTo         InetPortNumber,
    natSharedAddrMapGlobalAddrType      InetAddressType,
    natSharedAddrMapGlobalAddrFrom      InetAddress,
    natSharedAddrMapGlobalAddrTo        InetAddress,
    natSharedAddrMapGlobalPortFrom      InetPortNumber,
    natSharedAddrMapGlobalPortTo        InetPortNumber,
    natSharedAddrMapProtocol            NatProtocolMap,
    natSharedAddrMapInTranslates        Counter64,
    natSharedAddrMapOutTranslates       Counter64,
    natSharedAddrMapDiscards            Counter64,
    natSharedAddrMapAddrUsed            Gauge32,
    natSharedAddrMapStorageType         StorageType,
    natSharedAddrMapRowStatus           RowStatus
}

natSharedAddrMapIndex  OBJECT-TYPE
    SYNTAX      NatSharedAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Along with ifIndex, this object uniquely
             identifies an entry in the natAddrMapTable.
             Address map entries are applied in the order
             specified by natAddrMapIndex."
    ::= { natSharedAddrMapEntry 1 }

natSharedAddrMapName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(1..32))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "Name identifying all map entries in the table associated
             with the same interface.  All map entries with the same
             ifIndex MUST have the same map name."
    ::= { natSharedAddrMapEntry 2 }

natSharedAddrMapEntryType OBJECT-TYPE
    SYNTAX      NatAssociationType
    MAX-ACCESS  read-create
    STATUS      current
```

```
    DESCRIPTION
            "This parameter can be used to set up static
             or dynamic address maps."
    ::= { natSharedAddrMapEntry 3 }

natSharedAddrMapTranslatEntity OBJECT-TYPE
    SYNTAX      NatTranslationEntity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The end-point entity (source or destination) in
             inbound or outbound sessions (i.e., first packets) that
             may be translated by an address map entry.

             Session direction (inbound or outbound) is
             derived from the direction of the first packet
             of a session traversing a NAT interface.
             NAT address (and Transport-ID) maps may be defined
             to effect inbound or outbound sessions.

             Traditionally, address maps for Basic NAT and NAPT are
             configured on a public interface for outbound sessions,
             effecting translation of source end-point.  The value of
             this object must be set to outboundSrcEndPoint for
             those interfaces.

             Alternately, if address maps for Basic NAT and NAPT were
             to be configured on a private interface, the desired
             value for this object for the map entries
             would be inboundSrcEndPoint (i.e., effecting translation
             of source end-point for inbound sessions).

             If TwiceNAT were to be configured on a private interface,
             the desired value for this object for the map entries
             would be a bitmask of inboundSrcEndPoint and
             inboundDstEndPoint."
    ::= { natSharedAddrMapEntry 4 }

natSharedAddrMapLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo."
    ::= { natSharedAddrMapEntry 5 }

natSharedAddrMapLocalAddrFrom OBJECT-TYPE
```

```
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range
             of IP addresses mapped by this translation entry.  The
             value of this object must be less than or equal to the
             value of the natAddrMapLocalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natSharedAddrMapEntry 6 }

natSharedAddrMapLocalAddrTo OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses mapped by this translation entry.  If only
             a single address is being mapped, the value of this object
             is equal to the value of natAddrMapLocalAddrFrom.  For a
             static NAT, the number of addresses in the range defined
             by natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo must
             be equal to the number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natSharedAddrMapEntry 7 }

natSharedAddrMapLocalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped.

             The value of this object must be less than or equal to the
             value of the natAddrMapLocalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortTo."
```

```
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 8 }

natSharedAddrMapLocalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the last port number in the range
             of ports being mapped.

             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalPortFrom object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortFrom."
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 9 }

natSharedAddrMapGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo."
    ::= { natSharedAddrMapEntry 10 }

natSharedAddrMapGlobalAddrFrom OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range of
             IP addresses being mapped to.  The value of this object
             must be less than or equal to the value of the
             natAddrMapGlobalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natSharedAddrMapEntry 11 }

natSharedAddrMapGlobalAddrTo OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
```

```
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses being mapped to.  If only a single address is
             being mapped to, the value of this object is equal to the
             value of natAddrMapGlobalAddrFrom.  For a static NAT, the
             number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo must be
             equal to the number of addresses in the range defined by
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natSharedAddrMapEntry 12 }

natSharedAddrMapGlobalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped to.


             The value of this object must be less than or equal to the
             value of the natAddrMapGlobalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value natAddrMapGlobalPortTo."
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 13 }

natSharedAddrMapGlobalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of this
             object specifies the last port number in the range of
             ports being mapped to.

             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalPortFrom object.  If the
```

```
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapGlobalPortFrom."
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 14 }

natSharedAddrMapProtocol OBJECT-TYPE
    SYNTAX      NatProtocolMap
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies a bitmap of protocol identifiers."
    ::= { natSharedAddrMapEntry 15 }

natSharedAddrMapInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets pertaining to this address
             map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSharedAddrMapEntry 16 }

natSharedAddrMapOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets pertaining to this
             address map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSharedAddrMapEntry 17 }

natSharedAddrMapDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of packets pertaining to this address map
             entry that were dropped due to lack of addresses in the
```

```
            address pool identified by this address map.  The value of
            this object must always be zero in case of static
            address map.

            Discontinuities in the value of this counter can occur at
            reinitialization of the management system and at other
            times, as indicated by the value of
            ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSharedAddrMapEntry 18 }

natSharedAddrMapAddrUsed OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of addresses pertaining to this address map
             that are currently being used from the NAT pool.
             The value of this object must always be zero in the case
             of a static address map."
    ::= { natSharedAddrMapEntry 19 }

natSharedAddrMapStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row.
             Conceptual rows having the value 'permanent'
             need not allow write-access to any columnar objects
             in the row."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    DEFVAL { nonVolatile }
    ::= { natSharedAddrMapEntry 20 }

natSharedAddrMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row.

             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the natAddrMapRowStatus
             column is 'notReady'.

             None of the objects in this row may be modified
```

```
             while the value of this object is active(1)."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    ::= { natSharedAddrMapEntry 21 }


--
-- Notifications section
--

natMIBNotifications OBJECT IDENTIFIER ::= { natMIB 0 }

--
-- Notifications
--

natPacketDiscard NOTIFICATION-TYPE
    OBJECTS { ifIndex }
    STATUS  current
    DESCRIPTION
            "This notification is generated when IP packets are
             discarded by the NAT function; e.g., due to lack of
             mapping space when NAT is out of addresses or ports.

             Note that the generation of natPacketDiscard
             notifications is throttled by the agent, as specified
             by the 'natNotifThrottlingInterval' object."
    ::= { natMIBNotifications 1 }

--
-- Conformance information.
--

natMIBConformance OBJECT IDENTIFIER ::= { natMIB 2 }

natMIBGroups      OBJECT IDENTIFIER ::= { natMIBConformance 1 }
natMIBCompliances OBJECT IDENTIFIER ::= { natMIBConformance 2 }

--
-- Units of conformance
--

natConfigGroup OBJECT-GROUP
    OBJECTS { natInterfaceRealm,
              natInterfaceServiceType,
              natInterfaceStorageType,
              natInterfaceRowStatus,
              natAddrMapName,
```

```
              natAddrMapEntryType,
              natAddrMapTranslationEntity,
              natAddrMapLocalAddrType,
              natAddrMapLocalAddrFrom,
              natAddrMapLocalAddrTo,
              natAddrMapLocalPortFrom,
              natAddrMapLocalPortTo,
              natAddrMapGlobalAddrType,
              natAddrMapGlobalAddrFrom,
              natAddrMapGlobalAddrTo,
              natAddrMapGlobalPortFrom,
              natAddrMapGlobalPortTo,
              natAddrMapProtocol,
              natAddrMapStorageType,
              natAddrMapRowStatus,
              natSharedAddrMapName,
              natSharedAddrMapEntryType,
              natSharedAddrMapTranslatEntity,
              natSharedAddrMapLocalAddrType,
              natSharedAddrMapLocalAddrFrom,
              natSharedAddrMapLocalAddrTo,
              natSharedAddrMapLocalPortFrom,
              natSharedAddrMapLocalPortTo,
              natSharedAddrMapGlobalAddrType,
              natSharedAddrMapGlobalAddrFrom,
              natSharedAddrMapGlobalAddrTo,
              natSharedAddrMapGlobalPortFrom,
              natSharedAddrMapGlobalPortTo,
              natSharedAddrMapProtocol,
              natSharedAddrMapStorageType,
              natSharedAddrMapRowStatus,
              natBindDefIdleTimeout,
              natUdpDefIdleTimeout,
              natIcmpDefIdleTimeout,
              natOtherDefIdleTimeout,
              natTcpDefIdleTimeout,
              natTcpDefNegTimeout,
              natNotifThrottlingInterval }
    STATUS  current
    DESCRIPTION
            "A collection of configuration-related information
             required to support management of devices supporting
             NAT."
    ::= { natMIBGroups 1 }

natTranslationGroup OBJECT-GROUP
    OBJECTS { natAddrBindNumberOfEntries,
              natAddrBindGlobalAddrType,
```

```
              natAddrBindGlobalAddr,
              natAddrBindId,
              natAddrBindTranslationEntity,
              natAddrBindType,
              natAddrBindMapIndex,
              natAddrBindSessions,
              natAddrBindMaxIdleTime,
              natAddrBindCurrentIdleTime,
              natAddrBindInTranslates,
              natAddrBindOutTranslates,
              natAddrPortBindNumberOfEntries,
              natAddrPortBindGlobalAddrType,
              natAddrPortBindGlobalAddr,
              natAddrPortBindGlobalPort,
              natAddrPortBindId,
              natAddrPortBindTranslationEntity,
              natAddrPortBindType,
              natAddrPortBindMapIndex,
              natAddrPortBindSessions,
              natAddrPortBindMaxIdleTime,
              natAddrPortBindCurrentIdleTime,
              natAddrPortBindInTranslates,
              natAddrPortBindOutTranslates,
              natSessionPrivateSrcEPBindId,
              natSessionPrivateSrcEPBindMode,
              natSessionPrivateDstEPBindId,
              natSessionPrivateDstEPBindMode,
              natSessionDirection,
              natSessionUpTime,
              natSessionAddrMapIndex,
              natSessionProtocolType,
              natSessionPrivateAddrType,
              natSessionPrivateSrcAddr,
              natSessionPrivateSrcPort,
              natSessionPrivateDstAddr,
              natSessionPrivateDstPort,
              natSessionPublicAddrType,
              natSessionPublicSrcAddr,
              natSessionPublicSrcPort,
              natSessionPublicDstAddr,
              natSessionPublicDstPort,
              natSessionMaxIdleTime,
              natSessionCurrentIdleTime,
              natSessionInTranslates,
              natSessionOutTranslates }
    STATUS  current

    DESCRIPTION
```

```
            "A collection of BIND-related objects required to support
             management of devices supporting NAT."
    ::= { natMIBGroups 2 }

natStatsInterfaceGroup OBJECT-GROUP
    OBJECTS { natInterfaceInTranslates,
              natInterfaceOutTranslates,
              natInterfaceDiscards }
    STATUS  current
    DESCRIPTION
            "A collection of NAT statistics associated with the
             interface on which NAT is configured, to aid
             troubleshooting/monitoring of the NAT operation."
    ::= { natMIBGroups 3 }

natStatsProtocolGroup OBJECT-GROUP
    OBJECTS { natProtocolInTranslates,
              natProtocolOutTranslates,
              natProtocolDiscards }
    STATUS  current
    DESCRIPTION
            "A collection of protocol specific NAT statistics,
             to aid troubleshooting/monitoring of NAT operation."
    ::= { natMIBGroups 4 }

natStatsAddrMapGroup OBJECT-GROUP
    OBJECTS { natAddrMapInTranslates,
              natAddrMapOutTranslates,
              natAddrMapDiscards,
              natAddrMapAddrUsed,
              natSharedAddrMapInTranslates,
              natSharedAddrMapOutTranslates,
              natSharedAddrMapDiscards,
              natSharedAddrMapAddrUsed }
    STATUS  current
    DESCRIPTION
            "A collection of address map specific NAT statistics,
             to aid troubleshooting/monitoring of NAT operation."
    ::= { natMIBGroups 5 }

natMIBNotificationGroup NOTIFICATION-GROUP
    NOTIFICATIONS { natPacketDiscard }
    STATUS        current
    DESCRIPTION
            "A collection of notifications generated by
            devices supporting this MIB."
    ::= { natMIBGroups 6 }
```

```
--
-- Compliance statements
--

natMIBFullCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "When this MIB is implemented with support for
             read-create, then such an implementation can claim
             full compliance.  Such devices can then be both
             monitored and configured with this MIB.

             The following index objects cannot be added as OBJECT
             clauses but nevertheless have the compliance
             requirements:
                "
             -- OBJECT  natAddrBindLocalAddrType
             -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

             -- OBJECT  natAddrBindLocalAddr
             -- SYNTAX  InetAddress (SIZE(4|16))
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

             -- OBJECT  natAddrPortBindLocalAddrType
             -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

             -- OBJECT  natAddrPortBindLocalAddr
             -- SYNTAX  InetAddress (SIZE(4|16))
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

    MODULE IF-MIB -- The interfaces MIB, RFC2863
      MANDATORY-GROUPS {
        ifCounterDiscontinuityGroup
      }
```

```
    MODULE  -- this module
      MANDATORY-GROUPS { natConfigGroup, natTranslationGroup,
                         natStatsInterfaceGroup }

      GROUP       natStatsProtocolGroup
      DESCRIPTION
              "This group is optional."
      GROUP       natStatsAddrMapGroup
      DESCRIPTION
              "This group is optional."
      GROUP       natMIBNotificationGroup
      DESCRIPTION
              "This group is optional."

      OBJECT  natAddrMapLocalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."
```

```
      OBJECT  natAddrMapGlobalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPrivateAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPrivateSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."
```

```
      OBJECT  natSessionPrivateDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPublicAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPublicSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPublicDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

    ::= { natMIBCompliances 1 }

natMIBReadOnlyCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "When this MIB is implemented without support for
             read-create (i.e., in read-only mode), then such an
             implementation can claim read-only compliance.
             Such a device can then be monitored but cannot be
             configured with this MIB.

             The following index objects cannot be added as OBJECT
             clauses but nevertheless have the compliance
             requirements:
             "
             -- OBJECT  natAddrBindLocalAddrType
             -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
```

```
            --          on its support for IPv4 and IPv6."

            -- OBJECT  natAddrBindLocalAddr
            -- SYNTAX  InetAddress (SIZE(4|16))

            -- DESCRIPTION
            --          "An implementation is required to support
            --           global IPv4 and/or IPv6 addresses, depending
            --           on its support for IPv4 and IPv6."

            -- OBJECT  natAddrPortBindLocalAddrType
            -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
            -- DESCRIPTION
            --          "An implementation is required to support
            --           global IPv4 and/or IPv6 addresses, depending
            --           on its support for IPv4 and IPv6."
            -- OBJECT  natAddrPortBindLocalAddr
            -- SYNTAX  InetAddress (SIZE(4|16))
            -- DESCRIPTION
            --          "An implementation is required to support
            --           global IPv4 and/or IPv6 addresses, depending
            --           on its support for IPv4 and IPv6."

    MODULE IF-MIB -- The interfaces MIB, RFC2863
      MANDATORY-GROUPS {
        ifCounterDiscontinuityGroup
      }

    MODULE  -- this module
      MANDATORY-GROUPS { natConfigGroup, natTranslationGroup,
                         natStatsInterfaceGroup }

      GROUP       natStatsProtocolGroup
      DESCRIPTION
               "This group is optional."
      GROUP       natStatsAddrMapGroup
      DESCRIPTION
               "This group is optional."
      GROUP       natMIBNotificationGroup
      DESCRIPTION
               "This group is optional."
      OBJECT natInterfaceRowStatus
      SYNTAX RowStatus { active(1) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required, and active is the only
               status that needs to be supported."
```

```
      OBJECT  natAddrMapLocalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."
```

```
      OBJECT natAddrMapRowStatus
      SYNTAX RowStatus { active(1) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required, and active is the only
               status that needs to be supported."

      OBJECT  natAddrBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natAddrBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPrivateAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPrivateSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."
```

```
      OBJECT  natSessionPrivateDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPublicAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPublicSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPublicDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

    ::= { natMIBCompliances 2 }

END
```

7. Acknowledgements

The authors would like to thank R. Rohit, P. Srisuresh, Rajiv Raghunarayan, Nalinksh Pai, and Cliff Wang, the original authors of [RFC4008], as well as the following individuals who have participated in the drafting, review, and discussion of this memo:

Cathy Zhou, Juergen Schoenwaelder, Marc Blanchet, and Yu Fu.

8. Security Considerations

[To be reviewed, note about large number of mappings/bindings]

It is clear that this MIB can potentially be useful for configuration. Unauthorized access to the write-able objects could cause a denial of service and/or widespread network disturbance.

Hence, the support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

At this writing, no security holes have been identified beyond those that SNMP Security is itself intended to address. These relate primarily to controlled access to sensitive information and the ability to configure a device - or which might result from operator error, which is beyond the scope of any security architecture.

There are a number of managed objects in this MIB that may contain information that may be sensitive from a business perspective, in that they may represent NAT bind and session information. The NAT bind and session objects reveal the identity of private hosts that are engaged in a session with external end nodes. A curious outsider could monitor these two objects to assess the number of private hosts being supported by the NAT device. Further, a disgruntled former employee of an enterprise could use the NAT bind and session information to break into specific private hosts by intercepting the existing sessions or originating new sessions into the host. There are no objects that are sensitive in their own right, such as passwords or monetary amounts. It may even be important to control GET access to these objects and possibly to encrypt the values of these objects when they are sent over the network via SNMP. Not all versions of SNMP provide features for such a secure environment.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. IANA Considerations

TBD

10. References

10.1. 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.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC2463] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An

Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

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

[RFC3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.

10.2. Informative References

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[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.

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

Authors' Addresses

Simon Perreault
Viagenie
2875 boul. Laurier, suite D2-630
Quebec
Canada

Phone: +1-418-656-9254
EMail: simon.perreault@viagenie.ca

Tina Tsou
Huawei Technologies
2330 Central Expressway
Santa Clara
USA

Phone: +1-408-330-4424
EMail: tena@huawei.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

Behave
Internet-Draft
Intended status: Standards Track
Expires: July 18, 2013

S. Sivakumar
R. Penno
Cisco Systems

January 14, 2013

IPFIX Information Elements for logging NAT Events
draft-sivakumar-behave-nat-logging-06

Abstract

NAT devices are required to log events like creation and deletion of translations and information about the resources it is managing. With the wide deployment of Carrier Grade NAT (CGN) devices, the logging of events have become very important for legal purposes. The logs are required in many cases to identify an attacker or a host that was used to launch malicious attacks and/or for various other purposes of accounting. Since there is no standard way of logging this information, different NAT devices behave differently and hence it is difficult to expect a consistent behavior. The lack of a consistent way makes it difficult to write the collector applications that would receive this data and process it to present useful information. This document describes the information that is required to be logged by the NAT devices.

Table of Contents

1. Terminology

The usage of the term "NAT device" in this document refer to any NAT44 and NAT64 devices. The usage of the term "collector" refers to any device that receives the binary data from a NAT device and converts that into meaningful information. This document uses the term "Session" as it is defined in [RFC2663] and the term BIB as it is defined in [RFC6146]

2. Introduction

This document details the IPFIX Information Elements(IEs) that are required for logging by a NAT device. The document will specify the format of the IE's that are required to be logged by the NAT device and all the optional fields. The fields specified in this document are gleaned from [RFC4787] and [RFC5382].

2.1. Requirements Language

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

3. Scope

This document provides the information model to be used for logging the NAT devices including Carrier Grade NAT (CGN) events. This document focuses exclusively on the specification of IPFIX IE's. This document does not provide guidance on the transport protocol like TCP, UDP or SCTP that is to be used to log NAT events. The log events SHOULD NOT be lost but the choice of the actual transport protocol is beyond the scope of this document.

The existing IANA IPFIX Information Elements registry [IPFIX-IANA] already has assignments for many NAT logging events. For convenience, this document uses those same Information Elements. However, as stated earlier, this document is not defining IPFIX or Netflow 9 as the framework for logging. Rather, the information contained in these elements is within the scope of this document.

This document assumes that the NAT device will use the existing IPFIX framework to send the log events to the collector. This would mean that the NAT device will specify the template that it is going to use for each of the events. The templates can be of varying length and there could be multiple templates that a NAT device could use to log the events.

The implementation details of the collector application is beyond the scope of this document.

The optimization of logging the NAT events are left to the implementation and are beyond the scope of this document.

4. Event based logging

An event in a NAT device can be viewed as a happening as it relates to the management of NAT resources. The creation and deletion of NAT sessions and bindings are examples of events as it results in the resources (addresses and ports) being allocated or freed. The events can happen either through the processing of data packets flowing through the NAT device or through an external entity installing policies on the NAT router or as a result of an asynchronous event like a timer. The list of events are provided in Section 4.1. Each of these events SHOULD be logged, unless they are administratively prohibited. A NAT device MAY log these events to multiple collectors if redundancy is required. The network administrator will specify the collectors to which the log records are to be sent.

A collector may receive NAT events from multiple CGN devices and should be able to distinguish between the devices. Each CGN device should have a unique source ID to identify themselves. The source ID is part of the IPFIX template and data exchange.

Prior to logging any events, the NAT device MUST send the template of the record to the collector to advertise the format of the data record that it is using to send the events. The templates can be exchanged as frequently as required given the reliability of the connection. There SHOULD be a configurable timer for controlling the template refresh. NAT device SHOULD combine as many events as possible in a single packet to effectively utilize the network bandwidth.

4.1. Information Elements

The templates could contain a subset of the Information Elements(IEs) shown in Table 1 depending upon the event being logged. For example a NAT44 session creation template record will contain,

{sourceIPv4Adress, postNATSourceIPv4Address, destinationIpv4Address, postNATDestinationIPv4Address, sourceTransportPort, postNAPTSourceTransportPort, destinationTransportPort, postNAPTDestTransportPort, natOriginatingAddressRealm, natEvent, timeStamp}

An example of the actual event data record is shown below - in a readable form

{192.168.16.1, 201.1.1.100, 207.85.231.104, 207.85.231.104, 14800, 1024, 80, 80, 0, 1, 09:20:10:789}

A single NAT device could be exporting multiple templates and the collector should support receiving multiple templates from the same source.

The following is the table of all the IE's that a CGN device would need to export the events. The formats of the IE's and the IPFIX IDs are listed below.

Field Name	Size (bits)	IANA IPFIX ID	Description
timeStamp	64	323	System Time when the event occured.
vlanID	16	58	VLAN ID in case of overlapping networks
ingressVRFID	32	234	VRF ID in case of overlapping networks
sourceIPv4Address	32	8	Source IPv4 Address
postNATSourceIPv4Address	32	225	Translated Source IPv4 Address
protocolIdentifier	8	4	Transport protocol
sourceTransportPort	16	7	Source Port
postNAPTsourceTransportPort	16	227	Translated Source port
destinationIPv4Address	32	12	Destination IPv4 Address
postNATDestinationIPv4Address	32	226	Translated IPv4 destination address
destinationTransportPort	16	11	Destination port
postNAPTdestinationTransportPort	16	228	Translated Destination port
sourceIPv6Address	27	128	Source IPv6 address
destinationIPv6Address	128	28	Destination IPv6 address

postNATSourceIPv6Address	128	281	Translated source IPv6 addresss	
postNATDestinationIPv6Address	128	282	Translated Destination IPv6 address	
natOriginatingAddressRealm	8	229	Address Realm	
natEvent	8	230	Type of Event	
portRangeStart	16	361	Allocated port block start	
portRangeEnd	16	362	Allocated Port block end	
portRangeStepSize	16	363	Step size of next port	
portRangeNumPorts	16	364	Number of ports	

Table 1: Template format Table

4.2. Definition of NAT Events

The following are the list of NAT events and the proposed event values. The list can be expanded in the future as necessary. The data record will have the corresponding natEvent value to identify the event that is being logged.

Event Name	Values
NAT44 Session create	1
NAT44 Session delete	2
NAT Addresses exhausted	3
NAT64 Session create	4
NAT64 Session delete	5
NAT44 BIB create	6
NAT44 BIB delete	7
NAT64 BIB create	8
NAT64 BIB delete	9
NAT ports exhausted	10
Quota exceeded	11
Address Binding	12
Port block allocation	13
Port block de-allocation	14

Table 2: NAT Event ID table

4.3. Quota exceeded - Sub Event types

The following table shows the sub event types for the Quota exceeded event

Quota Exceeded Event Name	Values
Max Session entries	1
Max BIB entries	2
Max entries per user	3

Table 3: Sub Event ID table

4.4. Templates for NAT Events

The following is the template of events that will have to logged. The events below are identified at the time of this writing but the events are expandable. Depending on the implementation and configuration various IE’s specified can be included or ignored.

4.4.1. NAT44 create and delete session event

This event will be generated when a NAT44 session is created or deleted. The template will be the same, the natEvent will indicate whether it is a create or a delete event. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv4Address	32	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	Yes
sourceTransportPort	16	Yes
postNAPTsourceTransportPort	16	Yes
destinationIPv4Address	32	No
postNATDestinationIPv4Address	32	No
destinationTransportPort	16	No
postNAPTdestinationTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 4: NAT44 Session delete/create template

4.4.2. NAT64 create and delete session event

This event will be generated when a NAT64 session is created. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv6Address	128	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	Yes
sourceTransportPort	16	Yes
postNAPTsourceTransportPort	16	Yes
destinationIPv6Address	128	No
postNATDestinationIPv4Address	32	No
destinationTransportPort	16	No
postNAPTdestinationTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 5: NAT64 session create/delete event template

4.4.3. NAT44 BIB create and delete event

This event will be generated when a NAT44 Bind entry is created. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv4Address	32	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	No
sourceTransportPort	16	No
postNAPTsourceTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 6: NAT44 BIB create/delete event template

4.4.4. NAT64 BIB create and delete event

This event will be generated when a NAT64 Bind entry is created. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv6Address	128	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	No
sourceTransportPort	16	No
postNAPTsourceTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 7: NAT64 BIB create/delete event template

4.4.5. Addresses Exhausted event

This event will be generated when a NAT device runs out of global IPv4 addresses in a given pool of addresses. Typically, this event would mean that the NAT device wont be able to create any new translations until some addresses/ports are freed. The following is

a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
natPoolName	String	Yes

Table 8: NAT Address Exhausted event template

4.4.6. Ports Exhausted event

This event will be generated when a NAT device runs out of ports for a global IPv4 address. Port exhaustion shall be reported per protocol (UDP, TCP etc) The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	Yes

Table 9: NAT Ports Exhausted event template

4.4.7. Quota exceeded

This event will be generated when a NAT device cannot allocate resources as a result of an administratively defined policy. The examples of Quota exceeded are to allow only certain number of NAT sessions per device, certain number of NAT sessions per user etc. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
natLimitEvent	32	Yes
sourceIPv4 address	32	No
sourceIPv6 address	128	No

Table 10: NAT Quota Exceeded event template

4.4.8. Address Binding

This event will be generated when a NAT device binds a local address with a global address. This binding event happens when the first packet of the first flow from a host in the private realm.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
sourceIPv4 address	32	No
sourceIPv6 address	128	No
Translated Source IPv4 Address	32	8

Table 11: NAT Address Binding template

4.4.9. Port block allocation and de-allocation

This event will be generated when a NAT device allocates/de-allocates ports in a bulk fashion, as opposed to allocating a port on a per flow basis. NAT devices would do this in order to reduce logs and potentially to limit the number of connections a subscriber is allowed to use. In the following Port Block allocation template, the portRangeStart must be specified. Along with portRangeStart, atleast one of portRangeEnd, portRangeStepSize or portRangeNumPorts MUST be specified. If portRangeEnd is specified, it MUST NOT be lesser than portRangeStart. The value of portRangeStepSize MUST be between 1 and 32K.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
portRangeStart	16	Yes
portRangeEnd	16	No
portRangeStepSize	16	No
portRangeNumPorts	16	No

Table 12: NAT Port Block Allocation event template

5. Encoding

5.1. IPFIX

This document uses IPFIX as the encoding mechanism to describe the logging of NAT events. However, the information that should be logged SHOULD be the same irrespective of what kind of encoding scheme is used. IPFIX is chosen because is it an IETF standard that meets all the needs for a reliable logging mechanism. IPFIX provides the flexibility to the logging device to define the data sets that it is logging. The information elements specified for logging MUST be the same irrespective of the encoding mechanism used.

6. Acknowledgements

Thanks to Dan Wing, Selvi Shanmugam, Mohamed Boucadir, Jacni Qin Ramji Vaithianathan, Simon Perreault, Jean-Francois Tremblay and Julia Renouard for their review and comments.

7. IANA Considerations

8. Security Considerations

None.

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.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[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.

[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.

9.2. Informative References

[NAT-EVENT-LOG-IANA]
IANA, "NAT event log entities", 2012, <http://www.iana.org/assignments/nat-event-log/nat-event-log.xml>.

[RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.

[RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.

Authors' Addresses

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, North Carolina 27709
USA

Phone: +1 919 392 5158
Email: ssenthil@cisco.com

Renaldo Penno
Cisco Systems
170 W Tasman Drive
San Jose, California 95035
USA

Phone:
Email: repenno@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 10, 2012

A. Yourtchenko
D. Wing
cisco

December 8, 2011

Revealing hosts sharing an IP address using TCP option draft-wing-nat-reveal-option-03

Abstract

When an IP address is shared among several subscribers -- with a NAT or with an application-level proxy -- it is impossible for the server to differentiate between different clients. Such differentiation is valuable in several scenarios. This memo describes a technique to differentiate TCP clients sharing an IP address. The proposed method uses a TCP option, which avoids altering the application-level payload and works well with SSL-protected connections.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

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

This Internet-Draft will expire on June 10, 2012.

Copyright Notice

Table of Contents

1. Introduction

When clients are allocated unique, publicly-routable IPv4 addresses, it is easy to associate certain characteristics with their IP address. For example, if an IP address sends a lot of spam, that IP address is classified by many public (and private) system as "a spammer". Such classification can cause email or other traffic from that IP address to be blocked, rate limited, challenged with a captcha, or to receive other treatment. Reputation systems of various sorts exist for a wide variety of services on the Internet including IMAP, HTTP, ssh -- often these systems will slow down or interfere with normal login attempts when a dictionary attack is detected. An IP address can be added to a multitude of 'reputation' systems. Some of these systems are distributed across the Internet, some are shared amongst consenting parties, and some are operated by individual enterprises or individual hosts. Further discussion of the impacts of address sharing can be found in [I-D.ietf-intarea-shared-addressing-issues].

With the exhaustion of the IPv4 address space, IPv4 addresses will be shared on a large scale. This sharing will persist long after IPv6 is ubiquitous -- in fact, IPv4 address sharing will persist until all content and services on the Internet are available over IPv6. Once all content and services are available over IPv6, an Internet service provider will no longer need to provide access to the IPv4 Internet.

Until that time, both legitimate users and attackers will share IPv4 addresses. This IP address sharing means legitimate users will share the reputation of attackers.

This document describes a TCP option which can be added by an address sharing device such as a NAT or an application-level proxy. This TCP option allows a TCP server to differentiate between the TCP clients sharing that IP address.

An analysis of other techniques is available in [I-D.boucadair-intarea-nat-reveal-analysis].

2. Terminology

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

subscriber: the client accessing an address sharing device, who is responsible for the actions of their device(s). This might be an individual handset (with mobile devices), a home Internet connection,

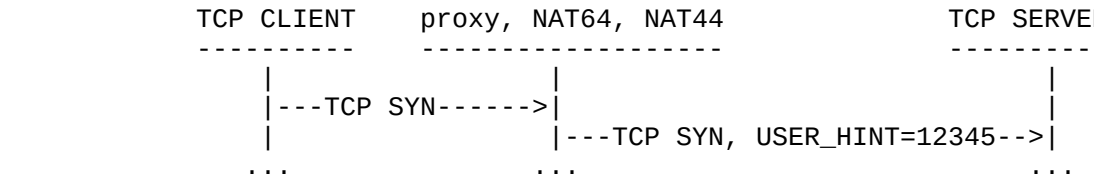
a small-medium business Internet connection, a University dormitory room, an individual employee of a company, or the company itself.

3. Description

This proposal defines one new TCP option, USER_HINT, to contain the TCP client's 16 bit identifier. This value might be the lower 16 bits of their IPv4 address, their VLAN ID, VRF ID, subscriber ID, or similar. The address sharing device (NAT, application proxy) would add the TCP option to the TCP SYN packet. TCP options are treated outside of the TCP sequence space, so no modifications of either sequence or acknowledgement numbers are needed.

3.1. Operation of Address Sharing Device

The address sharing device inserts the USER_HINT option into the TCP SYN, as depicted below. If the TCP SYN already has a USER_HINT option present, it is ignored and over-written with the new value.



3.2. Operation of the TCP Server

The TCP server identifies the client by combining the source IPv4 address in the IP header with the data in the USER_HINT option. This can be implemented by modifying the TCP stack to make the USER_HINT data available to the application via an API (e.g., via a socket option).

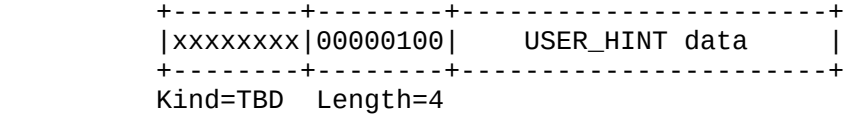
3.3. Reusing the USER_HINT value

The USER_HINT value is only 16 bits, so is obviously not globally unique. Even when combined with the publicly-routable IP address, the additional 16 bits are still not guaranteed to uniquely identify a particular subscriber. Out of necessity, the numbering space will be re-used by some address sharing devices, especially address sharing devices that are sharing many users on one IP address. As with today's IPv4 addresses which are assigned by an ISP, deterministically associating the IPv4 address (or IPv4 address and USER_HINT) with a particular subscriber requires more than simply completing a TCP 3-way handshake. For example, over a single day, an address sharing device might serve tens of thousands of different

subscribers from the same shared IP address, and thus it will need to rotate through the 16 bit USER_HINT space several times during the day. When doing so, the USER_HINT MUST NOT be re-used more often than every 2 minutes (a number chosen out of thin air); if an address sharing device needs to re-use a USER_HINT value more often than that, it should use additional IP addresses (to reduce how quickly the USER_HINT space is consumed on each address) or simply send TCP SYNs without USER_HINT until 2 minutes have elapsed. This 2 minute delay is necessary to allow the reputation system on a TCP server to differentiate between subscribers. For most implementations, the port sharing ratio (rather than a timer) is sufficient to meet this requirement.

4. USER_HINT Option Format

The USER_HINT option is always 4 bytes long, with 16 bits of USER_HINT data



User Hint option data: 16 bits.

If this option is present, it differentiates between active TCP hosts sharing the same IP address. This field MUST only be sent in the initial connection request (i.e., in segments with the SYN control bit set), or in the first ACK if the server's SYN contained the USER_HINT option.

5. Interaction with other TCP Options

This section details how USER_HINT functions in conjunction with other TCP options.

5.1. Option Space

As discussed in Appendix A of Multipath TCP (MPTCP) [I-D.ietf-mptcp-multiaddressed], there is a maximum of 40 bytes for TCP options, and a typical SYN (with MSS, window scale, SACK permitted, and timestamp options) leaves 16 bytes spare (if the options are word-aligned) or 21 bytes spare (if the options are not word-aligned).

Thus, the 4 byte option proposed in this memo would not cause a

problem with a typical TCP SYN.

5.2. Multipath TCP (MPTCP)

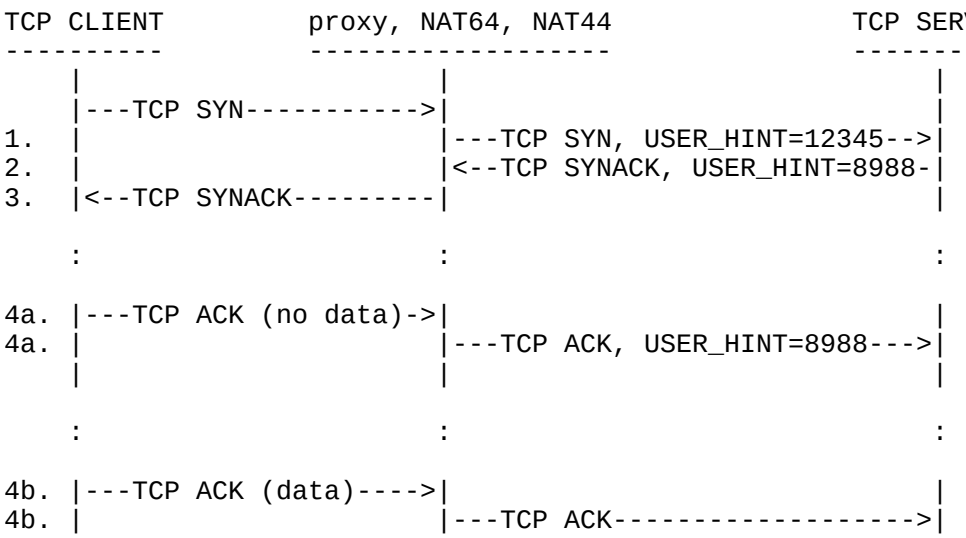
If the TCP client supports Multipath TCP (MPTCP) [I-D.ietf-mptcp-multiaddressed], the client will include the Multipath Capable or Multipath Join options to the TCP SYN. The Multipath Capable (MP_CAPABLE) option consumes 12 bytes, so a SYN containing all of these options would fully consume the 40 byte SYN option space. The Multipath Join (MP_JOIN) can consumes 12 or 16 bytes, but it is only used after successful early exchange containing the MP_CAPABLE option. Thus, there is reason to include USER_HINT if MP_JOIN is present in the TCP SYN -- if the MP_JOIN is not valid, it will be rejected by the server without creating any state on the server. Furthermore, if a client TCP is multi-homed, the client's TCP connections will probably go through different address sharing devices and thus have different externally-visible IP addresses and different USER_HINT values. Thus, it is NOT RECOMMENDED to include the USER_HINT option if the TCP SYN contains the MP_JOIN option.

5.3. Authentication Option (TCP-AO)

The USER_HINT option is incompatible with the Authentication Option (TCP-AO) [RFC5925], because TCP-AO provides integrity protection of the TCP SYN, including TCP options. However, TCP-AO is already incompatible with address sharing, because TCP-AO provides integrity protection of the source IP address.

6. Interaction with TCP SYN Cookies

TCP SYN cookies [RFC4987] are commonly deployed to mitigate TCP SYN attacks, which have some side effects. The USER_HINT information in the TCP SYN provides the TCP server with additional information it can use when deciding if this TCP connection attempt should be answered with a SYN cookie or should be answered normally. In the event the TCP server does not (or cannot) store the USER_HINT data, the USER_HINT data can be re-established on the TCP server when the client's first ACK is sent. There is a slight risk, however, that the client's first ACK, as seen by the middlebox, might contain data. If it does contain data, adding another 4 bytes to the packet could cause MTU to be exceeded.



The procedure is as follows:

1. Upon receiving a TCP SYN containing the USER_HINT option, the TCP server MAY respond to a SYN containing USER_HINT with an ACK packet containing its own USER_HINT value. (Note: this ACK response will typically have the SYN bit set.) If the server does not include the USER_HINT in its ACK packet, processing stops.

2. The middlebox, upon seeing the USER_HINT in the ACK, records those 2 bytes, which are used in a later step.

3. The middlebox strips the USER_HINT from the ACK, so it is not received by the TCP client. The middlebox sends the TCP ACK, without its USER_HINT option, to the TCP client.

4. The TCP client responds normally, generating a TCP ACK.

5. The middlebox receives an ACK from the TCP client. This ACK will either contain:

   A. no data, which causes the middlebox to add the USER_HINT value (from step 2) to the TCP ACK

   B. data, which causes the middlebox to simply forward the ACK packet. This is done to avoid MTU problems between the middlebox and the TCP server.

State is required in the address sharing device to perform the steps described in this section. This isn't a disaster with stateful address sharing (e.g., NAPT). However, in an A+P-like system (e.g., [I-D.ymbk-aplusp], [I-D.despres-intarea-4rd]), the CPE would need to perform the USER_HINT function, which introduces additional security considerations (not yet discussed in this version of the document).

7. Security Considerations

An attacker might use this functionality to appear as if IP address sharing is occurring, in the hopes that a naive server will allow additional attack traffic. TCP servers and applications SHOULD NOT assume the mere presence of the functionality described in this paper indicates there are other (benign) users sharing the same IP address.

8. Acknowledgements

Thanks to Anantha Ramaiah for the discussion. Thanks to Senthil Sivakumar for his review.

9. IANA Considerations

Assign a new TCP option number (kind value), USER_HINT.

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.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

10.2. Informative References

[I-D.boucadair-intarea-nat-reveal-analysis] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments", draft-boucadair-intarea-nat-reveal-analysis-04 (work in progress), September 2011.

[I-D.despres-intarea-4rd]

Despres, R., Matsushima, S., Murakami, T., and O. Troan, "IPv4 Residual Deployment across IPv6-Service networks (4rd) ISP-NAT's made optional", draft-despres-intarea-4rd-01 (work in progress), March 2011.

[I-D.ietf-intarea-shared-addressing-issues] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", draft-ietf-intarea-shared-addressing-issues-05 (work in progress), March 2011.

[I-D.ietf-mptcp-multiaddressed] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", draft-ietf-mptcp-multiaddressed-04 (work in progress), July 2011.

[I-D.ymbk-aplusp] Bush, R., "The A+P Approach to the IPv4 Address Shortage", draft-ymbk-aplusp-10 (work in progress), May 2011.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

Appendix A. Change History

[Note to RFC Editor: Please remove this section prior to publication.]

A.1. Changes from draft-wing-nat-reveal-option-01 to -02

- Limit option value to 16 bits (which becomes 32 bits total with the 8 bit option number and 8 bit length)
- described how USER_HINT can work successfully with Multipath TCP (MPTCP)'s options.
- Better described operation with TCP SYN Cookies.
- Renamed option from CX-ID to USER_HINT

Authors' Addresses

Andrew Yourtchenko
Cisco Systems, Inc.
6a de Kleetlaan
Diegem 1831
BE

Phone: +32 2 704 5494
Email: ayourtch@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose CA 95134
USA

Email: dwing@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 20, 2018

C. Bao
X. Li
Y. Zhai
W. Shang
CERNET Center/Tsinghua University
October 17, 2017

dIVI: Dual-Stateless IPv4/IPv6 Translation
draft-xli-behave-divi-08

Abstract

This document presents the concepts and the implementations of address-sharing dual-stateless IPv4/IPv6 translation (dIVI).

The dIVI is an extension of 1:1 stateless IPv4/IPv6 translation with features of IPv4 address sharing and dual translation. The major benefits of those extensions are using public IPv4 addresses more effectively and removing the requirements of DNS64 and ALG, without loosing the stateless, end-to-end address transparency and bidirectional-initiated communications.

Table of Contents

1. Introduction

The experiences for the IPv6 deployment in the past 10 years strongly indicate that for a successful transition, the communication between IPv4 and IPv6 address families should be supported.

Recently, the stateless and stateful IPv4/IPv6 translation methods are developed and became the IETF standards. The original stateless IPv4/IPv6 translation (stateless 1:1 IVI) is scalable, maintains the end-to-end address transparency and support both IPv6 initiated and IPv4 initiated communications [RFC6052], [RFC6144], [RFC6145], [RFC6147], [RFC6219]. But it can not use the IPv4 addresses effectively. The stateful IPv4/IPv6 translation can share the IPv4 addresses among IPv6 hosts, but it only supports IPv6 initiated communication [RFC6052], [RFC6144], [RFC6145], [RFC6146], [RFC6147]. In addition, both stateless and stateful IPv4/IPv6 translation technologies require the application layer gateway (ALG) for the applications which embed IP address literals.

In this document, we present concepts and the implementations of the dual-stateless IPv4/IPv6 translation (dIVI). The dIVI can solve the IPv4 address sharing and the ALG problems mentioned above, though still keeps the stateless, end-to-end address transparency and supporting of both IPv6 initiated and IPv4 initiated communications. The dIVI is in the family of stateless IPv4/IPv6 translation, along with IVI and dIVI-PD [I-D.xli-behave-divi-pd]. These techniques are used for the following scenarios with second translation extension defined in [RFC6144].

- Scenario 1: IPv4 hosts via an IPv6 network to the IPv4 Internet.
- Scenario 2: The IPv4 Internet via an IPv6 network to IPv4 hosts.
- Scenario 5: An IPv6 network via an IPv4 network to IPv4 hosts.
- Scenario 6: IPv4 hosts via an IPv4 network to an IPv6 network.

2. Applicability

The address sharing dual stateless IPv4/IPv6 translation (dIVI) is shown in the following figure.

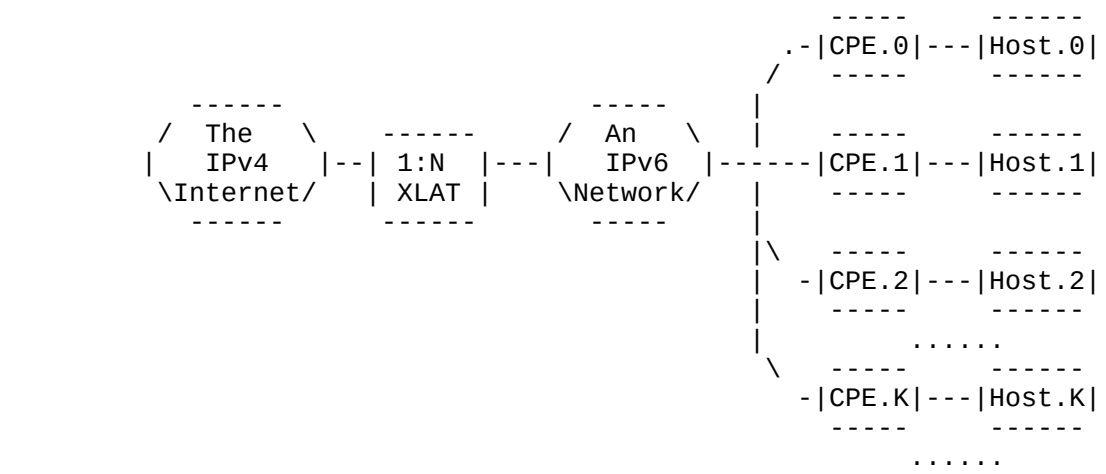


Figure 1: dIVI

Where

- The 1:N XLAT (first translator) is the IPv4/IPv6 translator perform stateless 1:N translation between the IPv4 Internet and an IPv6 network.

- The CPE.0 to CPE.K (second translators) are 1:1 IPv6/IPv4 translators/IPv6 routers which also perform port mapping function for Host.x when it is communicating with the IPv4 Internet with IPv4 address sharing.

- The Host.1 to Host.N are the dual-stack hosts.

The IPv6 network routes the IPv6 more specific routes of the IPv4-translatable addresses (longer than /64) defined in Section 5 of this document to corresponding CPEs. In the case of prefix delegation (shorter than /64), the dual stateless IPv4/IPv6 translation with prefix delegation (dIVI-pd) defined in [I-D.xli-behave-divi-pd] can be used.

3. Terminologies

This document uses the terminologies defined in [RFC6144].

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

4. Port Mapping Algorithm and Address Format Extension

4.1. Port Mapping Algorithm

The dual stateless IPv4/IPv6 translation (dIVI) uses the port mapping algorithm defined in [I-D.bcx-behave-address-fmt-extension].

For given sharing ratio (R) and the maximum number of continue ports (M), the generalized modulus algorithm is defined as

1. The port number (P) of a given PSID (K) is composed of

   P = R*M*j + M*K + i

   Where
   o PSID: K=0 to R-1
   o Port range index: j = (1024/M)/R to ((65536/M)/R)-1, if the well-known port numbers (0-1023) are excluded.
   o Port continue index: i=0 to M-1

2. The PSID (K) of a given port number (P) is determined by

   K = (floor(P/M)) % R

   Where
   o % is modular operator
   o floor(arg) is a function returns the largest integer not greater than arg

3. The well-known port number (0-1023) can be used, if additional port mapping rule is defined.

For example, for R=128, M=4

	Port range-1	Port rang-2	Port
PSID=0	1024, 1025, 1026, 1027,	1536, 1537, 1538, 1539,	2048
PSID=1	1028, 1029, 1030, 1031,	1540, 1541, 1542, 1543,	....
PSID=2	1032, 1033, 1034, 1035,	1544, 1545, 1546, 1547,	....
PSID=3	1036, 1037, 1038, 1039,	1548, 1549, 1550, 1551,	....
...			
PSID=127	1532, 1533, 1534, 1535,	2044, 2045, 2046, 2047,	....

Figure 2: Example

4.2. Address Format Extensions

The dual stateless IPv4/IPv6 translation (dIVI) uses the address format extension defined in [I-D.bcx-behave-address-fmt-extension].

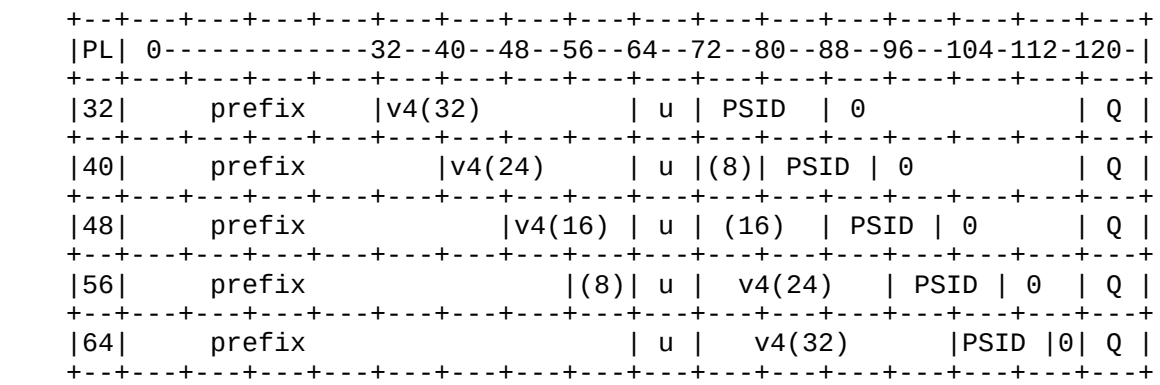


Figure 3: Address Format

Where PL designates the prefix length.

The PSID prefix length (Q) is encoded in the last octet (bits 120-127) to indicate the number of ports can be used. When Q=0, the extended address format will become the address format defined in [RFC6052]. The relations between Q, the sharing ratio (R), the maximum continue port range (M) and the number of ports can be shown in the following figure.

Q	Ratio	Maximum M	# of Ports
0	1:1	65,536	65,536
1	1:2	32,786	32,786
2	1:4	16,384	16,384
3	1:8	8,192	8,192
4	1:16	4,096	4,096
5	1:32	2,048	2,048
6	1:64	1,024	1,024
7	1:128	512	512
8	1:256	256	256
9	1:512	128	128
10	1:1,024	64	64
11	1:2,048	32	32
12	1:4,096	16	16

Figure 4: Port Range

4.3. Stateless Translation Algorithm with Address Sharing

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

- If the packets are from the IPv4 Internet to an IPv6 network, the IPv4 source addresses are translated to the IPv4-converted addresses and the source port numbers are unchanged before and after translation; the IPv4 destination addresses are translated to the extended IPv4-translatable addresses based on the destination port number and the destination port numbers are unchanged before and after translation. Note that this means that only a specific IPv6 node can receive the packets for a specific port number.

- If the packets are from an IPv6 network to the IPv4 Internet, the IPv6 source addresses and the source port numbers are checked, if the source port number matches the port number range defined by the extended IPv4-translatable address format, the IPv6 source addresses (which are the IPv4-translatable addresses) are translated to the IPv4 addresses and the source port numbers are unchanged before and after translation; the destination IPv6 addresses (which are the IPv4-converted addresses) are translated to the IPv4 destination addresses and the destination port numbers are unchanged before and after translation. However, if the source port number does not match the port number range defined by the extended IPv4-translatable address format, the packets will be dropped.

4.4. Partial-state Translation Algorithm with Address Sharing

Stateless translation requires that IPv6 nodes generate source port number in the range defined by the extended IPv4-translatable address. If this condition does not hold, the partial-state translation algorithm can be used, which can map the random source port number to the extended IPv4-translatable address defined source port number range. Due to the requirement of the states in the translator for the port mapping (no states required for the address mapping), it is named partial state. The technical details of the port mapping state maintaining algorithm is an implementation issue (see Section 6.2).

5. Dual Stateless Translation

In general dual stateful IPv4/IPv6 translations (IPv4-IPv6-IPv4) are

considered harmful, since the two translators cannot synchronize the parameter to translate the IPv4 address to its original format. However, the dual stateless IPv4/IPv6 translation (IPv4-IPv6-IPv4) can translate the IPv4 address to its original format based on algorithm. There are several reasons for using dual stateless IPv4/IPv6 translation.

- The second translator (CPE) performs the port-set mapping algorithm discussed in Section 4.1, therefore, there is no need to modify the host with IPv4 address sharing.
- ALG cannot be developed for some applications, or has not been developed during the early transition stage.
- DNS64 is not allowed (due to DNSSec, etc.).
- All the IPv6 packet processing tools (routing, policy based routing, packet filtering, traffic shaping based on layer 3 and layer 4 can be used for dual stateless translation, while new tools are required for encapsulation and tunneling.

5.1. Header Translation and MTU Handling

The general header and ICMP translation specifications are defined in [RFC6145].

Special MTU and fragmentation actions must be taken in the case of dual translation.

5.2. Port Mapping Algorithm in CPE

The port mapping algorithm is straightforward. The port mapping device maintains a database of allowed port numbers defined by the extended IPv4-translatable address format. If the packets from the host contains the source port number which do not match the port number range defined by the extended IPv4-translatable address format, the CPE will map the source port number to an allowed one and keep the record in the database for mapping back the returning packets and all the packets in the same session.

The port number database can be refreshed via the corresponding transport layer flags for TCP or via timeout for UDP sessions.

6. Implementation Considerations

6.1. Host implementation

For the wireless mobile Internet environment, it is not difficult to modify the operating system of the mobile device, therefore it possible to integrate the port mapping algorithm and the second IPv4/IPv6 translation function in the mobile device, which is an IPv6-only host to the network and has a dual-stack socket API for the applications running on this host.

6.2. Port Mapping in the Core Translator

The port mapping can be performed in the core translator, in this case, there is no need to have the second translator (CPE.x) and no need to modify the IPv6 host for the port restriction requirements.

7. Security Considerations

There are no security considerations in this document.

8. IANA Considerations

This memo adds no new IANA considerations.

Note to RFC Editor: This section will have served its purpose if it correctly tells IANA that no new assignments or registries are required, or if those assignments or registries are created during the RFC publication process. From the author’s perspective, it may therefore be removed upon publication as an RFC at the RFC Editor’s discretion.

9. Acknowledgments

The authors would like to acknowledge the following contributors in the different phases of the address-sharing IVI and dIVI development: Maoke Chen, Hong Zhang, Weifeng Jiang, Yuncehng Zhu and Guoliang Han.

The authors would like to acknowledge the following contributors who provided helpful inputs: Dan Wing, Fred Baker, Dave Thaler, Randy Bush, Kevin Yin, Wojciech Dec and Rajiv Asati.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <https://www.rfc-editor.org/info/rfc6052>.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, DOI 10.17487/RFC6144, April 2011, <https://www.rfc-editor.org/info/rfc6144>.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, DOI 10.17487/RFC6145, April 2011, <https://www.rfc-editor.org/info/rfc6145>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[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, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC6219] Li, X., Bao, C., Chen, M., Zhang, H., and J. Wu, "The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition", RFC 6219, DOI 10.17487/RFC6219, May 2011, <https://www.rfc-editor.org/info/rfc6219>.

10.2. Informative References

[I-D.bcx-behave-address-fmt-extension] Bao, C. and X. Li, "Extended IPv6 Addressing for Encoding Port Range", draft-bcx-behave-address-fmt-extension-10 (work in progress), January 2017.

[I-D.xli-behave-divi-pd]
Sun, Q., Asati, R., Xie, C., Li, X., Dec, W., and C. Bao, "dIVI-pd: Dual-Stateless IPv4/IPv6 Translation with Prefix Delegation", draft-xli-behave-divi-pd-01 (work in progress), September 2011.

Appendix A. Address Format and Workflow Examples

The prefix we selected is 2001:da8:a4a6::/48. The formats of the IPv4-converted address and the IPv4-translatable address are:

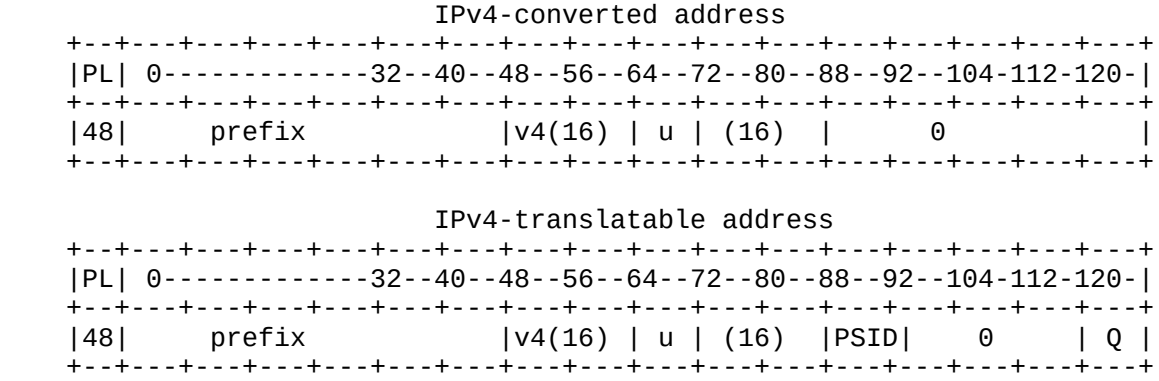


Figure 5: Address format

The sharing ratio R=16, so the PSID has 4 bits (Q=4) and each PSID can have (4096-1024/16)= 4032 concurrent port numbers to use. The continue port range M=4, so each PSID will have 1008 available port range.

The host in the IPv4 Internet is 202.112.35.254 and the corresponding IPv4-converted address is 2001:da8:a4a6:ca70:23:fe00::

The shared IPv4 address is 202.38.117.1 with sharing ratio 16, the corresponding IPv4-translatable addresses are

```
PSID=0    2001:da8:a4a6:ca26:75:100::4
PSID=1    2001:da8:a4a6:ca26:75:110::4
PSID=2    2001:da8:a4a6:ca26:75:120::4
PSID=3    2001:da8:a4a6:ca26:75:130::4
PSID=4    2001:da8:a4a6:ca26:75:140::4
PSID=5    2001:da8:a4a6:ca26:75:150::4
PSID=6    2001:da8:a4a6:ca26:75:160::4
PSID=7    2001:da8:a4a6:ca26:75:170::4
PSID=8    2001:da8:a4a6:ca26:75:180::4
```

```
PSID=9    2001:da8:a4a6:ca26:75:190::4
PSID=10   2001:da8:a4a6:ca26:75:1a0::4
PSID=11   2001:da8:a4a6:ca26:75:1b0::4
PSID=12   2001:da8:a4a6:ca26:75:1c0::4
PSID=13   2001:da8:a4a6:ca26:75:1d0::4
PSID=14   2001:da8:a4a6:ca26:75:1e0::4
PSID=15   2001:da8:a4a6:ca26:75:1f0::4
```

A.1. The address-sharing host on an IPv6 network initiates communication

An address-sharing Host.0 (202.38.117.1) in an IPv6 network behind CPE initiates communication with Host 202.112.35.254:80 in the IPv4 Internet

```
On the Host.0
Src#p= 202.38.117.1#1881 (random port)
Dst#p= 202.112.35.254#80 (server port)

On an IPv6 network
Src#p= [2001:da8:a4a6:ca26:75:0100::4]#8192 (CPE mapped port)
Dst#p= [2001:da8:a4a6:ca70:23:fe00::]#80 (server port)

On the IPv4 Internet
Src#p= 202.38.117.1#8192 (CPE mapped port)
Dst#p= 202.112.35.254#80 (server port)
```

Figure 6: Workflow Example 1

The returning packets reverse the Src and Dst, the CPE maps the "CPE mapped port (8192)" back to the original "random port (1881)".

A.2. A host in the IPv4 Internet initiates communication

A host 202.112.35.254 in the IPv4 Internet initiates communication to the address-sharing Host.0 (202.38.117.1#4096)

```
      On the IPv4 Internet
      Src#p= 202.112.35.254#36567 (random port)
      Dst#p= 202.38.117.1#4096 (server port)

      On an IPv6 network
      Src#p= [2001:da8:a4a6:ca70:23:fe00::]#36567 (random port)
      Dst#p= [2001:da8:a4a6:ca26:75:0100::4]#4096 (server port)

      On the Host.0
      Src#p= 202.112.35.254#36567 (random port)
      Dst#p= 202.38.117.1#4096 (server port)
```

Figure 7: Workflow Example 2

The returning packets reverse the Src and Dst.

Authors' Addresses

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

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

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

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

Yu Zhai
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing 100084
CN

Phone: +86 10-62785983
Email: jacky.zhai@gmail.com

Wentao Shang
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing 100084
CN

Phone: +86 10-62785983
Email: wentaoshang@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 15, 2011

X. Li
C. Bao
CERNET Center/Tsinghua University
D. Wing
R. Vaithianathan
Cisco
G. Huston
APNIC

May 14, 2011

Stateless Source Address Mapping for ICMPv6 Packets
draft-xli-behave-icmp-address-04

Abstract

A stateless IPv4/IPv6 translator may receive ICMPv6 packets containing non IPv4-translatable addresses as the source that should be passed across the translator as an ICMP packet directed to the the IPv4-translatable destination. This document discusses the considerations and the stateless address mapping algorithms for source address translation in ICMPv6 headers for such cases.

Status of this Memo

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

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

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

This Internet-Draft will expire on November 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

The IP/ICMP translation document of IPv4/IPv6 translation [RFC6145] states that "the IPv6 addresses in the ICMPv6 header may not be IPv4-translatable addresses and there will be no corresponding IPv4 addresses represented of this IPv6 address. In this case, the translator can do stateful translation. A mechanism by which the translator can instead do stateless translation is left for future work." This document defines such a stateless translation mechanism.

2. Notational Conventions

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

3. Problem Statement and Considerations

When a stateless IPv4/IPv6 translator receives an ICMPv6 message (for example "Packet Too Big") sourced from an non-IPv4-translatable IPv6 address, directed to an IPv4-translatable IPv6 address, it needs to generate an ICMP message. For the reasons discussed below, choosing the source IPv4 address of this ICMP message is problematic.

The address used should not cause the ICMP packet to be a candidate for discarding, particularly in the contest of uRPF filters [RFC3704]. This consideration precludes the use of private IPv4 address space [RFC1918] in this context.

It is also a consideration that the IPv4/IPv6 translation is intended for use in contexts where IPv4 addresses may not be readily available, so it is not considered to be appropriate to use IPv4-translatable IPv6 addresses for all internal points in the IPv6 network that may originate ICMPv6 messages.

It is also an objective that it is possible for the IPv4 recipient of the ICMP message be able to distinguish between different IPv6 ICMPv6 originations (for example, to support a traceroute diagnostic utility that provides some limited network level visibility across the IPv4/IPv6 translator). This implies that a IPv4/IPv6 translator needs to have a pool of IPv4 addresses to be used for mapping the source address of ICMPv6 packets generated from different originations.

These addresses are for use in the source address of ICMP packets, and therefore are not intended to be used as a destination address for any packet. It is therefore possible to use a common address

pool for the IPv4/IPv6 translation protocol, and, considering an objective of constraining the use of these IPv4 addresses in this application, it is feasible to use a common address pool for mapping the source addresses of non-translatable ICMPv6 packets as a part of the protocol specification.

These considerations leads to the recommendation of drawing an IPv4 /24 prefix from the IANA Special Purpose Address Registry as a "Well-Known Prefix" for use by IPv4/IPv6 translators for the purpose of mapping otherwise untranslatable IPv6 source addresses of ICMPv6 messages to IPv4 ICMP messages.

4. Routing Considerations

Addresses from the assigned address prefix are intended to be used as source addresses and not as destination addresses in the context of the public network. As packets passing through the public network need to pass through conventional packet filters, including uRPF filters [RFC3704], this implies that the assigned address may be used in routing advertisements. Such routing advertisements are non-exclusive and should be accepted from any originating AS in an anycast fashion.

5. Possible Stateless Address Mapping Algorithms

When an IPv4 /24 prefix is allocated to represent the source address of ICMP, the Last Octet can be generated using one of the following algorithms.

- The translator can randomly generates the Last Octet of the /24 prefix for different non IPv4-translatable addresses. However, in this case the translator may need to maintain states to ensure same non IPv4-translatable IPv6 address maps to same IPv4 address.

- The translator can copy the "Hop Count" in the IPv6 header of the ICMPv6 to the Last Octet. Routers typically emit ICMPv6 packets with the same hop count, thus as the ICMPv6 packet is routed through the network its hop count is decreased. However, if the routers emit ICMPv6 packets with different hop counts, it may give the appearance of a routing loop to tools such as traceroute. That minor side-effect in that particular case cannot be avoided while still being stateless.

- Hashing of the IPv6 address to generate a 8 bit value which will be used to generate the last octet. In this case, there is no need to maintain expensive states, except hashing table which

should be simple with minimal memory usage and consume minimal CPU cycles. If the hashing function is good and there are limited number of IPv6 routers (< 256) on the IPv6 side of the network, we will get unique IPv4 addresses to map the addresses of the IPv6 routers with O(1) lookup.

The selection of the algorithm SHOULD be a configuration function in the IPv4/IPv6 translator.

6. Security Considerations

The use of an address for source addresses in ICMP packets is considered "safe" in so far as ICMP packets are not intended to generate responses directed to the source address.

However it is possible to use this address as a means of gaining anonymity when launching a denial of service attacks by using this address as the source address for other forms of malicious traffic. Packet firewall filters should be configured to treat addresses in the IANA-assigned /24 network as martian addresses by discarding all non-ICMP packets that use the IANA-assigned /24 network as a source address, and all packets that use the IANA-assigned /24 network as a destination address.

7. IANA Considerations

IANA is requested to make a permanent assignment of a /24 from the IPv4 Special Purpose Address Registry [RFC5736]. The assigned address is to be used in the context of generating an IPv4 source address for mapped ICMPv6 packets being passed through a stateless IPv4/IPv6 translator. The assignment is under the category of a specialized use of a designated address block in an anycast context associated with an Internet Standards Track protocol.

The IANA IPv4 Special Purpose Address Registry records are:

- Prefix 192.70.192.0/24
- Description: To be used in the context of generating an IPv4 source address for mapped ICMPv6 packets being passed through a stateless IPv4/IPv6 translator.
- Begin: 2011-06-01
- End: Never

- Purpose: Stateless ICMPv6/ICMP translation
- Contact: See RFC
- Scope: Addresses from the assigned address prefix are intended to be used as source addresses and not as destination addresses in the context of the public network.
- RFC: This draft.

8. Acknowledgments

The authors would like to acknowledge the following contributors of this document: Kevin Yin, Chris Metz and Neeraj Gupta.

9. References

9.1. 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.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

9.2. Informative References

[RFC5736] Huston, G., Cotton, M., and L. Vegoda, "IANA IPv4 Special Purpose Address Registry", RFC 5736, January 2010.

Authors' Addresses

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

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

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

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

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Email: dwing@cisco.com

Ramji Vaithianathan
Cisco Systems, Inc.
A 5-2, BGL 12-4, SEZ Unit,
Cessna Business Park, Varthur Hobli
Sarjapur Outer Ring Road
BANGALORE KARNATAKA 560 103
INDIA

Phone: +91 80 4426 0895
Email: rvaithia@cisco.com

Geoff Huston
APNIC

Email: gih@apnic.net

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