Abstract

This specification defines a NAT that allows IPv6-only hosts that are inside the NAT to operate with IPv4 hosts that are outside the NAT. It requires no modifications to the v4 hosts or applications, or to the operating system of v6 hosts, but it does require some changes to v6 applications. Typically this specification would be used to allow the hosts inside a NAT to connect to hosts outside it; but under some limitations, it can also allow hosts outside to connect to hosts inside.

The goal of this draft is to consider what is the minimal feasible approach to this problem. The current intention is to merge this with the nat64 draft. This draft is being discussed on the
behave@ietf.org list.

Table of Contents

1. Introduction ............................................. 3
2. Architecture ............................................. 3
  2.1. NAT from 6 to 4 ...................................... 3
  2.2. NAT from 4 to 6 ...................................... 5
3. Terminology ............................................... 6
4. Mapping ...................................................... 6
5. Packet Forwarding .......................................... 7
  5.1. UDP and TCP 6 to 4 .................................... 7
  5.2. UDP and TCP 4 to 6 .................................... 7
  5.3. IP Fragmentation ...................................... 8
  5.4. TTL .................................................... 8
  5.5. DSCP ................................................... 9
  5.6. ICMP .................................................. 9
    5.6.1. Option Yes ICMP .................................. 9
    5.6.2. Option No ICMP ................................... 9
  5.7. IPsec ................................................. 9
  5.8. DCCP, SCTP, etc ...................................... 10
6. ALGs ......................................................... 10
  6.1. FTP .................................................... 10
  6.2. SIP .................................................... 10
  6.3. Others ............................................... 10
7. Helper Services ............................................ 10
  7.1. STUN ................................................... 11
  7.2. DNS ................................................... 11
  7.3. DHCP .................................................. 11
  7.4. V6 Routing / Tunnel .................................. 11
8. Requirement Conformance .................................... 11
  8.1. RFC 4787 Requirements ................................ 11
  8.2. draft-ietf-behave-tcp Requirements .................... 12
  8.3. draft-bagnulo-v6ops-6man-nat64-pb-statement
    Requirements .......................................... 12
  8.4. draft-nishitani-cgn Requirements ....................... 12
  8.5. Multicast ............................................. 12
9. IANA Considerations ....................................... 13
10. Security Considerations .................................. 13
11. Acknowledgements ........................................ 13
12. References ............................................... 14
  12.1. Normative References ................................ 14
  12.2. Informative References .............................. 14
Author’s Address ............................................ 14
13. Intellectual Property and Copyright Statements .......... 16
1. Introduction

A lot of thought has gone into the question of how to connect v6-only and v4-only hosts. This draft tries to identify the simplest approach to what would just "barely work," since what barely works is what is most likely to get deployed. The basic approach is to ask what works for v4 to v4 NATs and to pick a design that requires as few changes as possible from that. The assumption is that it is hard to deploy changes in existing v4 applications and in both v4 and v6 operating systems. Of course, this specification relies on subjective judgements about the relative complexity of deploying changes into various parts of the network.

This specification therefore amounts to a rough straw man meant to encourage discussion of what is the minimum that can be done.

2. Architecture

2.1. NAT from 6 to 4

The deployment topology that this work addresses has many v6-only hosts behind some large, carrier-grade NAT, and these hosts wish to be able to form connections to the v4 internet. This is the classic NAT64 case. The harder NAT46 case is discussed later. As shown in Figure 1, when a new connection is initiated by the host inside the NAT using internal address X1 and port x1, the NAT creates a mapping from the inside address port combination X1:x1 to some unused address and port on the outside. The external v4 address that X1:x1 was mapped to is referred to as X1':x1, and any packets that are sent to this port on the NAT are forwarded to X1:x1.
The client application needs to know how to create a v6 destination address that will be routed to the v4 server. First the client application finds the v4 address of the server. One way to find the server is using a DNS query. The DNS query will go through a device that both synthesizes a DNS AAAA record for the A record and returns the A record. If the end application knows how to look at the A record directly and synthesize the v6 address to use, it can do that and continue to support features such as DNSSEC. If the application is unaware of the fact it is behind a NAT, it can use the AAAA record as a normal v6 only application would. The v6 address is formed by putting a ::FFFE:0:0/96 prefix before the v4 address. This prefix range is routed to the NAT, which can extract the v4 destination address.

The mapping from X1:x1/X1’:x1’ is maintained by the NAT as long as it sees periodic traffic being sent from X1:x1. This specification only defines the NAT for UDP and TCP but could be extended for other protocols that have a port multiplexing scheme. When a mapping is created for a particular port, that mapping exists for all protocols, not just the protocol that created the mapping. This greatly simplifies generating the keep alive traffic that is necessary to
maintain the mapping.

2.2.  NAT from 4 to 6

Making it possible for a v4 host to connect to a server on the v6 side requires more complex changes to the v6 applications than the trivial changes that were required in the 6 to 4 direction. However, many applications that usefully have a server running behind a NAT have already changed to work behind v4 to v4 NATs. The approach here is the same.

```
+-------------------------------+
|    +----------+ External v4    |
|    |Client App|               |
|    +----------+               |
|    |Host OS   |               |
|    +----------+               |
|      Y1:y1                    |
+-------------------------------+
|    +--------+   +----+        |
|    |STUN|   |X1':x1'|        |
|    +----+   +--------+        |
+----+--------+-----------------+
| NAT    |                  |
+--------+-----------------+
|    +--------+   Internal v6 |
|    |Server |                 |
|    +--------+                 |
|      X1:x1                    |
|    +--------+                 |
|    |Host OS|                 |
|    +--------+                 |
+-------------------------------+
```

Figure 2

The server starts by creating a mapping in the NAT to a v4 address that it can use. The server does this by creating a connection to some server in the outside v4 space and having that server tell it what address and port the request came from, which reveals the external X1’:x1’ address port that has been mapped to the port the server is using. Typically a STUN server would be used. Note that a UDP transaction to a STUN server will allocate a mapping that can be used for incoming TCP connections. The NAT is required to run a STUN server on the external side at the address ::FFFE:127.0.0.1 on the default STUN port so the server will always be able to find a STUN
server if it is behind a NAT. [[ Note - I have no idea if this address hack would work - but we can skin the cat with some other potato peeler if it does not ]]. The server needs to send periodic keep alive traffic to make sure the NAT does not remove the mapping. This can also be done with STUN.

Next the server lets the client know that it can be reached at the address port of X1’:x1’. This might be done simply, such as by creating a URL that points to that location and providing it by whatever means the URL is found (email, IM, bathroom walls, whatever); or through a complex approach, such as by using a rendezvous server such as a SIP registrar or by using DNS SRV records as rendezvous servers that point at the correct address and port. At this point, a client in the v4 space can initiate a connection to the X1’:x1’, and this will form a connection to the server.

The applications that tend to be popular to run behind NATs are mostly P2P applications, such as file sharing and VoIP. Many of these types of applications have already undergone the changes that would be necessary to enable them to work behind a NAT such as the one described here, because these changes let them work behind v4 to v4 NATs. HTTP servers may also turn out to be valuable to run behind NATs. The use cases would likely not involve direct web page serving so much as places where a web application, such as Facebook, could make an RPC call to an application that was running on the v6 server behind the NAT. The URL that Facebook uses for the RPC calls can easily be updated by the application. This type of architecture is already becoming more common as people use virtual servers in elastic computing environments. These environments are often behind NATs to facilitate migration of virtual servers. It is worth noting, particularly for future protocol development, that if HTTP did a DNS SRV lookup, it would be possible to use DNS as the rendezvous server for a generic web browser on the v4 internet to reach a v6-only server behind the NAT.

3. 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 RFC 2119 [RFC2119].

4. Mapping

When the NAT receives a packet with a source of X1:x1 from a host on the inside, the NAT first checks to see if it already has a mapping for it. If so, it resets the timer for the mapping; otherwise, it
creates a new mapping. If the NAT already has any mapping from X1 to an external address X1’, the external address used for this new mapping MUST be the same as the external address used for previous mappings of X1. The external address/port combination (X1’:x1’) allocated for the new mapping MUST NOT be in use by any other mapping. When choosing a port number for the mapping, well known port numbers MUST NOT be used.

The mappings timer MUST keep mappings for at least 10 minutes after any packet is sent from the internal host through that mapping. Note that applications should not assume that the mapping will be promptly removed after 10 minutes of inactivity, since NAT implementations often do not use a timer per mapping but instead use a periodic sweep approach to deleting mapping.

5. Packet Forwarding

5.1. UDP and TCP 6 to 4

When the NAT receives a UDP or TCP packet from the inside, it updates the mapping as described in Section 4, and then it extracts the external v4 address from the v6 address by striping the ::FFFE:0:0/96 prefix. Next it takes the payload section of the UDP or TCP packet and constructs a v4 UDP or TCP payload using this destination and source from the associated mapping. Then it sends the packet following the procedures defined for a host to send a UDP or TCP packet. Note that any IP options (other than fragmentation) are lost in this process; also, as defined in UDP and TCP, new checksums will be updated.

It is worth noting that the v4 destination address of the packet may be one of the external addresses of this NAT, in which case the packet is forwarded to that address and processed just like any other packet arriving at this address. Packets can therefore “hairpin” though the NAT. There is no good way to avoid this without some modifications to the applications to allow them to be aware of this and optimize around it. ICE is an example of a way applications can optimize around this hairpinning.

5.2. UDP and TCP 4 to 6

When the NAT receives a UDP or TCP packet from the outside, it checks to see if it has a mapping for the address port that the packet was received on. If it does, it uses the internal address and port associated with the mapping as the destination. It constructs a v6 UDP or TCP packet from the payload of the received packet and forwards that to the internal address. Note that checksums are
updated when this new packet is created and sent.

If the NAT receives a TCP SYN packet for which there is no mapping it SHOULD silently discard it; otherwise TCP hole punching techniques using simultaneous opens will not work.

5.3. IP Fragmentation

The interaction of v4 and v6 fragmentation has some thorny issues.

The NAT MUST support reassembly of fragmented packets when the packets are received in order, but support for out of order fragments is OPTIONAL.

If the NAT receives a v4 packet with the do not fragment bit set, it MUST NOT forward it if the MTU of the v6 link would require the packet to be fragmented, and the NAT MUST NOT include a fragment header in the v6 packet. If the NAT receives a v4 packet that does not have the do not fragment bit set, then the packet, when forwarded, MUST include a fragment header; and if the packet is larger than the MTU, the packet MUST be appropriately fragmented. When the NAT receives a v6 packet without a fragment header, it MUST set the do not fragment bit in any v4 packets, and if the resulting v4 packet is larger than the MTU on the v4 link that will be used, the NAT MUST NOT forward the packet. When the NAT receives a v6 packet with a fragment header, it can send the v4 packet without the do not fragment bit set and can fragment the packet appropriately.

The problem with this approach is that the v6 host is likely to send packets less than 1280 octets with no fragmentation header. The v4 side will interpret these packets as being unable to be fragmented, and if they are destined for a host in which some element of the path has a shorter MTU, the packet will be discarded. The only practical way to mitigate this situation is to have the application send most of it packets with a fragmentation header, even if they are smaller than 1280 octets.

Application that support this specification, when sending to a v6 address that starts with the prefix ::FFFE:0:0/96 SHOULD include a fragment header regardless of the size of the packet.

5.4. TTL

When a packet is forwarded, the TTL is decremented by one. If it is zero, the packet is not forwarded.
5.5. DSCP

When packets pass from one side to the other, the DSCP needs to be copied. If the NAT also includes diffserv classifier and marker functionality it MAY change the DSCP values. See RFC 2983[RFC2983] for more information.

5.6. ICMP

[[ Note - this is going to be controversial. I suspect it actually goes too far but I’m deliberately presenting a pretty far out there side of the argument, in the hope that it will drive a discussion about what we really need, in practice, if we ignore IETF dogma. ]]

5.6.1. Option Yes ICMP

ICMP packets are translated as described in [RFC2765].

5.6.2. Option No ICMP

ICMP can be split into two parts, the part that reports errors for other transports and the part that supports exactly two applications, ping and traceroute. The problem with ICMP for reporting transport errors is that on today’s internet ICMP is so often blocked that no application can rely on ICMP working. Applications tend to be designed to work without ICMP. NAT processing of ICMP packets is complicated because ICMP packets may contain embedded IP packets or fragments thereof. The security is further complicated by the fact that a UDP or TCP packet may cause an ICMP error to be generated by a host other than the host for which the original packet was destined. This possibility makes it difficult to determine which ICMP packets are valid and which are not. Furthermore, the way the APIs for UDP are typically used makes it hard for operating systems to notify applications of ICMP errors.

NAT processing of ICMP errors is complicated and of very limited value; for that reason forwarding ICMP error messages is OPTIONAL. Processing to allow ping and traceroute through the NAT is also OPTIONAL

5.7. IPsec

IPsec over UDP will work, but other forms of IPsec will generally not work in a reliable way.
5.8. DCCP, SCTP, etc

This specification can be extended by future specifications to support other transports but, given the immediate needs, this specification only requires support for UDP and TCP. This allows vendors that have not implemented other protocols to be compliant with this specification. No significant problems are seen with using the same basic approach for DCCP.

SCTP may be more complicated. Often one of the reasons for using SCTP is to take advantage of multi-homing for reliability reasons but this may require that the two SCTP sessions try and use different NATs. The requirements and deployments topologies are not clear.

6. ALGs

6.1. FTP

The deployment of personal firewalls and the misconfiguration of other firewalls has resulted in widespread breakage of active mode FTP. The general guess is that an FTP ALG will be needed to take PASV responses and turn them into EPAS responses. However, more experimentation is needed with what happens today with existing FTP clients running on v6 only hosts behind NATs to determine what is the best approach to this problem.

6.2. SIP

Many widely deployed SIP implementations work fine through NATs without requiring any ALGs. SIP was not designed to work with ALGs. More importantly, ALGs are not considered when designing new extensions to SIP and the combination of the extensions and the ALG often break badly. It is NOT RECOMMENDED for the NAT to have SIP ALGs, but if SIP ALGs are insisted upon, the best approach is to implement a dual homed proxy in the NAT that does v6 on one side and v4 on the other. This approach will be compatible with future SIP extensions and is significantly easier to correctly implement as SIP was designed so this would work.

6.3. Others

All other ALGs are NOT RECOMMENDED.

7. Helper Services

There are several services that, while not actually part of NATs,
greatly facilitate being able to build applications that reliably work through NATs and should be logically, if not physically, associated with the NAT.

7.1. STUN

The NAT must run a Basic Standalone STUN server as defined in section 13 of [I-D.ietf-behave-rfc3489bis], with the exception that it is NOT REQUIRED that it support TCP and it is not necessary to provide a DNS entry for this server. The server MUST run on the address ::FFFFE:127.0.0.1 with default port of 3478. [[Note, if this address hack is inappropriate, this could be resolved by just defining a well known v4 anycast address for STUN and using that]]

7.2. DNS

The NAT MUST provide a recursive DNS resolver that is capable of taking a DNS request received from the inside and resolving it on the outside.

The resolver SHOULD try to take DNS A records results from the outside and synthesize synthetic AAAA records that would be routed to the using the v6 prefix. This SHOULD NOT be done if a record exists that does not use the NAT prefix address.

7.3. DHCP

DHCP is very useful for the automated configuration of many things beyond IP addresses. [[ TODO should any particular DHCP things be required]]

7.4. V6 Routing / Tunnel

If a packet arrives from the inside with a v6 destination that is not in the ::FFFFE:0:0/96 range, the NAT could/(should/may) forward this to the v6 internet. This could be via a direct v6 connection or some 646 tunnel. [[ Note not sure what to require / recommend here ]]

8. Requirement Conformance

8.1. RFC 4787 Requirements

NATs meeting this specification are compliant with the specification defined for UDP NAT behavior in RFC 4787 [RFC4787], with the exception that RFC 4787 requires reassembly of out of order packets while this does not.
8.2. draft-ietf-behave-tcp Requirements

NATs meeting this specification are compliant with the specification defined for [I-D.ietf-behave-tcp], with the exception of Req-9, which requires ICMP, and Req-5, which requires that mapping of established TCP connections with no traffic to stay valid for 2 hours and 4 minutes, while this requires only 10 minutes.

8.3. draft-bagnulo-v6ops-6man-nat64-pb-statement Requirements

Meets all the MUST support items in [I-D.bagnulo-v6ops-6man-nat64-pb-statement] except the requirement in R7 for ICMP support.

Meets all the SHOULD support items except:

- I4 requires support for out of order fragmented packets. See security consideration section in this document for more discussion on this.
- I6 - not clear whether or not all of MIPv6 works with this.
- I7 & I8 require support for DCCP and SCTP which could be done as an extension to this.
- I9 - not clear when this does and does not work with multicast.

8.4. draft-nishitani-cgn Requirements

Meets all the requirements of [I-D.nishitani-cgn] except the following:

- R4 & R9 - require support of ICMP.
- R5 & R6 are additional constraints on reserving ports which are not mandated by this specification; but a device that was fully compliant with this specification could still support these.
- R7 & R8 are analyzed in Section 8.1 and Section 8.2.

8.5. Multicast

More analysis is needed - your mileage may vary. Some important multicast applications such as an IP TV-like system that used an SSM sender in the v4 space with clients in the v6 space could probably be made to work fine, with little modification for v6-only clients. Some other multicast scenarios with senders in both the v4 and v6 space probably would not work.
9. IANA Considerations

This document makes no request of IANA.

Open Issues: What prefix to use. We will want to allocate a different prefix than the ::FFFF:0:0/96. This draft makes no argument about which prefix would be best, it just requires that the specification define some fixed prefix to use.

Note to RFC Editor: this section may be removed on publication as an RFC.

10. Security Considerations

Often NATs are combined with firewalls that perform address-dependent filtering (as defined in [RFC4787]) to improve security. The type of NAT envisioned here is a large, carrier-grade NAT with many clients behind it. Performing firewall operations at this location in the topology is not particularly effective because the attacker may well be on the "inside". The firewall capabilities should be provided much closer to the host being protected. The benefit of having mappings with address-independent filtering is that it make it possible to easily run servers inside the NAT with no modification of the clients outside the NAT. For this reason, this specification adopts a NAT design with address-independent filtering.

One of the concerns about a large scale NAT that a single host inside the NAT might be able to perform a DOS attack by using up a large portion of the available external ports simply by creating many mappings. To mitigate this, the NAT SHOULD allow a configurable limit to the number of mappings that can be created by a single host inside the NAT.

TODO - reassembly of out of order packets

11. Acknowledgements

Thanks to Dave Ward who pointed out that I and others were spending too much time making this complicated and should stop talking and get on with writing some drafts. Thanks for helpful comments from Mangus Westerlund, Iljitsch van Beijnum, and .

12. References
12.1. Normative References

[I-D.ietf-behave-rfc3489bis]
Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)",
draft-ietf-behave-rfc3489bis-16 (work in progress), July 2008.


12.2. Informative References

[I-D.bagnulo-v6ops-6man-nat64-pb-statement]

[I-D.ietf-behave-tcp]
Guha, S., "NAT Behavioral Requirements for TCP",

[I-D.nishitani-cgn]
nishitani, T. and S. Miyakawa, "Carrier Grade Network Address Translator (NAT) Behavioral Requirements for Unicast UDP, TCP and ICMP",
draft-nishitani-cgn-00 (work in progress), July 2008.


Author’s Address

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA  95134
USA

Phone:  +1 408 902-3341
Email:  fluffy@cisco.com