

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: March 13, 2014

R. Stewart
Adara Networks
M. Tuexen
I. Ruengeler
Muenster Univ. of Appl. Sciences
September 09, 2013

Stream Control Transmission Protocol (SCTP) Network Address Translation
draft-ietf-behave-sctpnat-09.txt

Abstract

Stream Control Transmission Protocol [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT for TCP that allows multiple hosts to reside behind a NAT and yet use only a single globally unique IPv4 address, even when two hosts (behind a NAT) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation or NAPT. To date, specialized code for SCTP has NOT yet been added to most NATs so that only pure NAT is available. The end result of this is that only one SCTP capable host can be behind a NAT.

This document describes an SCTP specific variant of NAT which provides similar features of NAPT in the single point and multi-point traversal scenario.

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 13, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	3
3. Terminology	3
4. SCTP NAT Traversal Scenarios	4
4.1. Single Point Traversal	4
4.2. Multi Point Traversal	5
5. Limitations of Classical NAPT for SCTP	6
6. The SCTP Specific Variant of NAT	6
7. NAT to SCTP	10
8. Handling of Fragmented SCTP Packets	10
9. Various Examples of NAT Traversals	10
9.1. Single-homed Client to Single-homed Server	10
9.2. Single-homed Client to Multi-homed Server	12
9.3. Multihomed Client and Server	15
9.4. NAT Loses Its State	18
9.5. Peer-to-Peer Communication	20
10. IANA Considerations	24
11. Security Considerations	24
12. Acknowledgments	24
13. References	24
13.1. Normative References	24
13.2. Informative References	25
Authors' Addresses	25

1. Introduction

Stream Control Transmission Protocol [RFC4960] provides a reliable communications channel between two end-hosts in many ways similar to TCP [RFC0793]. With the widespread deployment of Network Address Translators (NAT), specialized code has been added to NAT for TCP that allows multiple hosts to reside behind a NAT and use private

addresses (see [RFC5735]) and yet use only a single globally unique IPv4 address, even when two hosts (behind a NAT) choose the same port numbers for their connection. This additional code is sometimes classified as Network Address and Port Translation or NAPT. To date, specialized code for SCTP has not yet been added to most NATs so that only true NAT is available. The end result of this is that only one SCTP capable host can be behind a NAT.

This document proposes an SCTP specific variant NAT that provides the NAPT functionality without changing SCTP port numbers. The authors feel it is possible and desirable to make these changes for a number of reasons.

- o It is desirable for SCTP internal end-hosts on multiple platforms to be able to share a NAT's public IP address, much as TCP does today.
- o If a NAT does not need to change any data within an SCTP packet it will reduce the processing burden of NAT'ing SCTP by NOT needing to execute the CRC32c checksum required by SCTP.
- o Not having to touch the IP payload makes the processing of ICMP messages in NATs easier.

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].

3. Terminology

For this discussion we will use several terms, which we will define and point out in Figure 1.

Private-Address (Priv-Addr): The private address that is known to the internal host.

Internal-Port (Int-Port): The port number that is in use by the host holding the Private-Address.

Internal-VTag (Int-VTag): The Verification Tag that the internal host has chosen for its communication. The VTag is a unique 32 bit tag that must accompany any incoming SCTP packet for this association to the Private-Address.

External-Address (Ext-Addr): The address that an internal host is attempting to contact.

External-Port (Ext-Port): The port number of the peer process at the External-Address.

External-VTag (Ext-VTag): The Verification Tag that the host holding the External-Address has chosen for its communication. The VTag is a unique 32 bit tag that must accompany any incoming SCTP packet for this association to the External-Address.

Public-Address (Pub-Addr): The public address assigned to the NAT box which it uses as a source address when sending packets towards the External-Address.

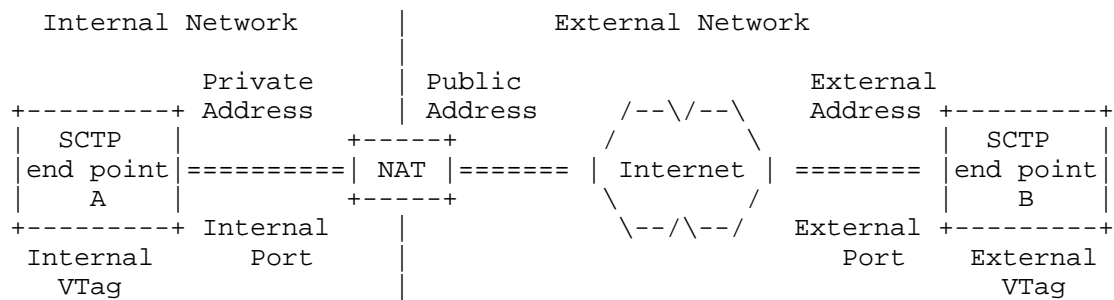


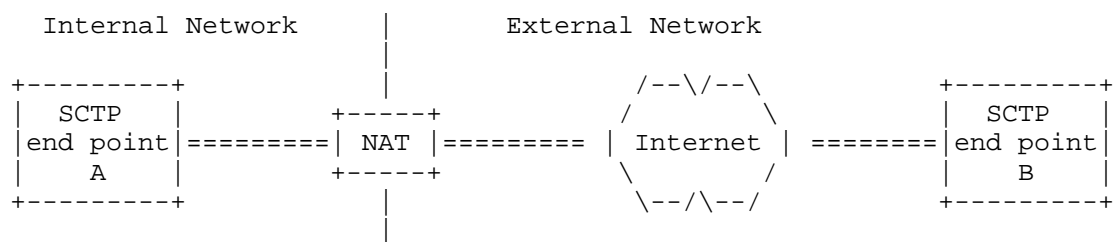
Figure 1: Architecture

4. SCTP NAT Traversal Scenarios

This section defines the notion of single and multi-point NAT traversal.

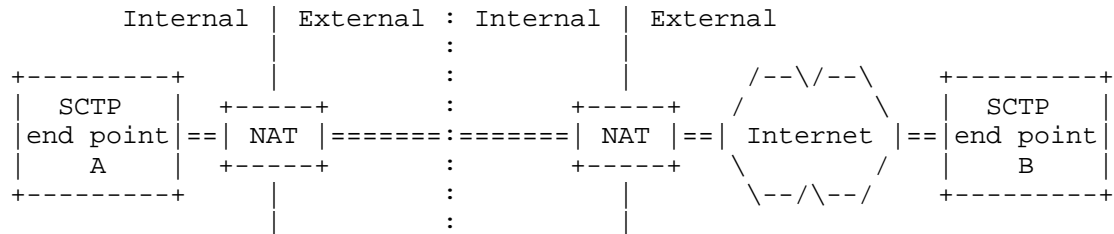
4.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT, as shown below:



Single NAT scenario

A variation of this case is shown below, i.e., multiple NATs in a single path:



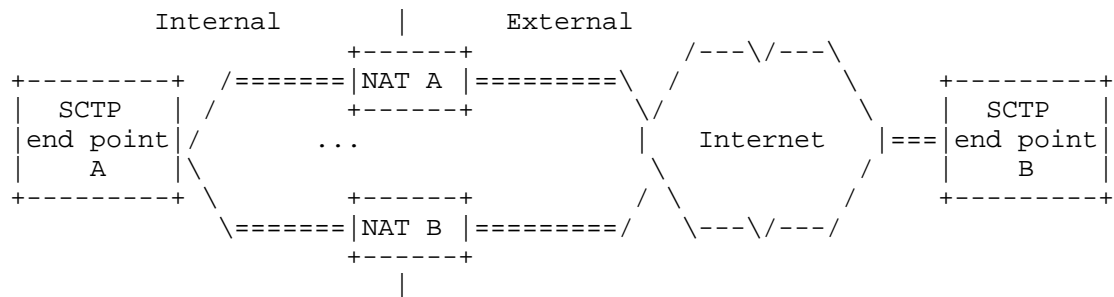
Serial NATs scenario

In this single point traversal scenario, we must acknowledge that while one of the main benefits of SCTP multi-homing is redundant paths, the NAT function represents a single point of failure in the path of the SCTP multi-home association. However, the rest of the path may still benefit from path diversity provided by SCTP multi-homing.

The two SCTP endpoints in this case can be either single-homed or multi-homed. However, the important thing is that the NAT (or NATs) in this case sees all the packets of the SCTP association.

4.2. Multi Point Traversal

This case involves multiple NATs and each NAT only sees some of the packets in the SCTP association. An example is shown below:



Parallel NATs scenario

This case does NOT apply to a single-homed SCTP association (i.e., BOTH endpoints in the association use only one IP address). The advantage here is that the existence of multiple NAT traversal points can preserve the path diversity of a multi-homed association for the

entire path. This in turn can improve the robustness of the communication.

5. Limitations of Classical NAT for SCTP

Using classical NAT may result in changing one of the SCTP port numbers during the processing which requires the recomputation of the transport layer checksum. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed. This would add considerable to the NAT computational burden, however hardware support may mitigate this in some implementations.

An SCTP endpoint may have multiple addresses but only has a single port number. To make multipoint traversal work, all the NATs involved must recognize the packets they see as belonging to the same SCTP association and perform port number translation in a consistent way. One possible way of doing this is to use pre-defined table of ports and addresses configured within each NAT. Other mechanisms could make use of NAT to NAT communication. Such mechanisms are considered by the authors not to be deployable on a wide scale base and thus not a recommended solution. Therefore the SCTP variant of NAT has been developed.

6. The SCTP Specific Variant of NAT

In this section we assume that we have multiple SCTP capable hosts behind a NAT which has one Public-Address. Furthermore we are focusing in this section on the single point traversal scenario.

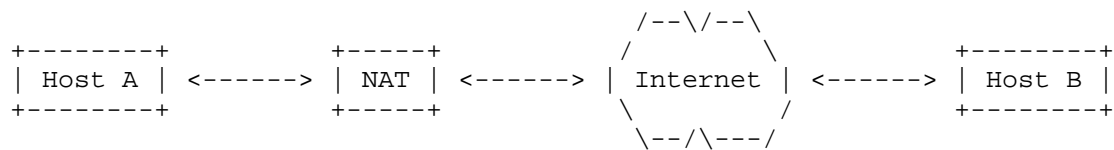
The modification of SCTP packets sent to the public Internet is easy. The source address of the packet has to be replaced with the Public-Address. It may also be necessary to establish some state in the NAT box to handle incoming packets, which is discussed later.

For SCTP packets coming from the public Internet the destination address of the packets has to be replaced with the Private-Address of the host the packet has to be delivered to. The lookup of the Private-Address is based on the External-VTag, External-Port, External-Address, Internal-VTag and the Internal-Port.

For the SCTP NAT processing the NAT box has to maintain a table of Internal-VTag, Internal-Port, Private-Address, External-VTag, External-Port and whether the restart procedure is disabled or not. An entry in that table is called a NAT state control block. The function Create() obtains the just mentioned parameters and returns a NAT-State control block.

The entries in this table fulfill some uniqueness conditions. There must not be more than one entry with the same pair of Internal-Port and External-Port. This rule can be relaxed, if all entries with the same Internal-Port and External-Port have the support for the restart procedure enabled. In this case there must be no more than one entry with the same Internal-Port, External-Port and Ext-VTag and no more than one entry with the same Internal-Port, External-Port and Int-VTag.

The processing of outgoing SCTP packets containing an INIT-chunk is described in the following figure. The scenario shown is valid for all message flows in this section.



```

                INIT[Initiate-Tag]
Priv-Addr:Int-Port -----> Ext-Addr:Ext-Port
                        Ext-VTag=0

                Create(Initiate-Tag, Int-Port, Priv-Addr, 0)
                Returns(NAT-State control block)

Translate To:

                INIT[Initiate-Tag]
Pub-Addr:Int-Port -----> Ext-Addr:Ext-Port
                        Ext-VTag=0

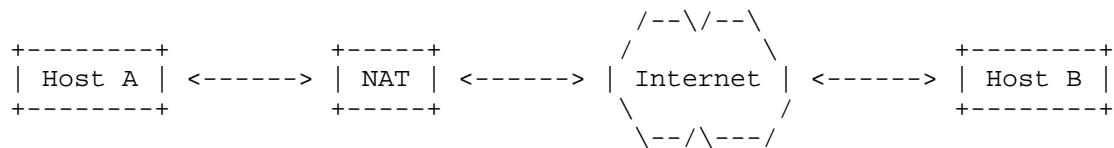
```

It should be noted that normally a NAT control block will be created. However, it is possible that there is already a NAT control block with the same External-Address, External-Port, Internal-Port, and Internal-VTag but different Private-Address. In this case the INIT SHOULD be dropped by the NAT and an ABORT SHOULD be sent back to the SCTP host with the M-Bit set and an appropriate error cause (see [I-D.ietf-tsvwg-natsupp] for the format). The source address of the packet containing the ABORT chunk MUST be the destination address of the packet containing the INIT chunk.

It is also possible that a connection to External-Address and External-Port exists without an Internal-VTag conflict but the

External-Address does not support the DISABLE_RESTART feature (noted in the NAT control block when the prior connection was established). In such a case the INIT SHOULD be dropped by the NAT and an ABORT SHOULD be sent back to the SCTP host with the M-Bit set and an appropriate error cause (see [I-D.ietf-tsvwg-natsupp] for the format).

The processing of outgoing SCTP packets containing no INIT-chunk is described in the following figure.

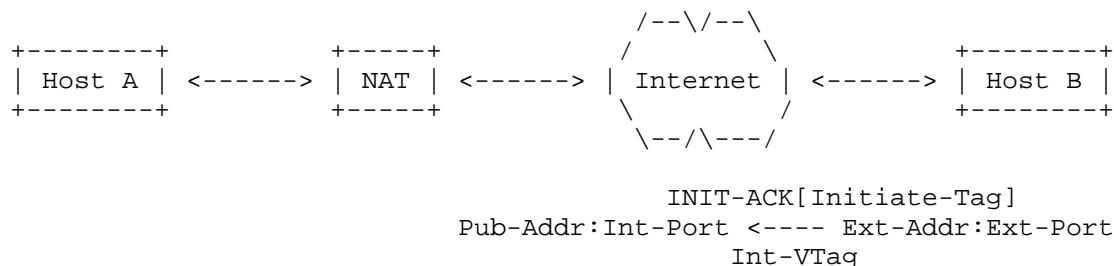


Priv-Addr:Int-Port -----> Ext-Addr:Ext-Port
 Ext-VTag

Translate To:

Pub-Addr:Int-Port -----> Ext-Addr:Ext-Port
 Ext-VTag

The processing of incoming SCTP packets containing INIT-ACK chunks is described in the following figure. The Lookup() function getting as input the Internal-VTag, Internal-Port, External-VTag (=0), External-Port, and External-Address, returns the corresponding entry of the NAT table and updates the External-VTag by substituting it with the value of the Initiate-Tag of the INIT-ACK chunk. The wildcard character signifies that the parameter's value is not considered in the Lookup() function or changed in the Update() function, respectively.




```

Lookup(Int-VTag, Int-Port, *, 0, Ext-Port)
Update(*, *, *, Initiate-Tag, *)

```

```

Returns(NAT-State control block containing Private-Address)

```

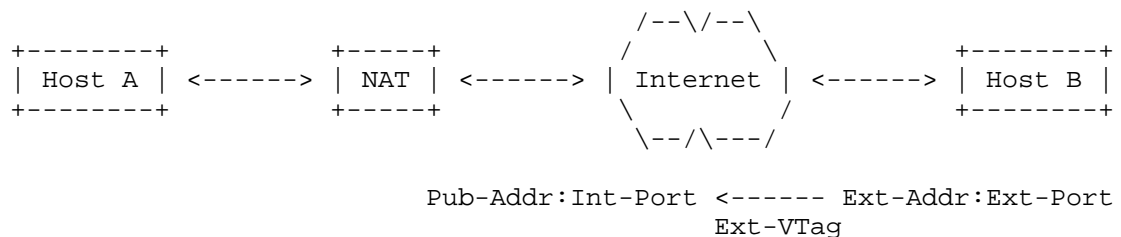
```

INIT-ACK[Initiate-Tag]
Priv-Addr:Int-Port <----- Ext-Addr:Ext-Port
Int-VTag

```

In the case Lookup fails, the SCTP packet is dropped. The Update routine inserts the External-VTag (the Initiate-Tag of the INIT-ACK chunk) in the NAT state control block.

The processing of incoming SCTP packets containing an ABORT or SHUTDOWN-COMPLETE chunk with the T-Bit set is described in the following figure.



```

Lookup(0, Int-Port, *, Ext-VTag, Ext-Port)

```

```

Returns(NAT-State control block containing Private-Address)

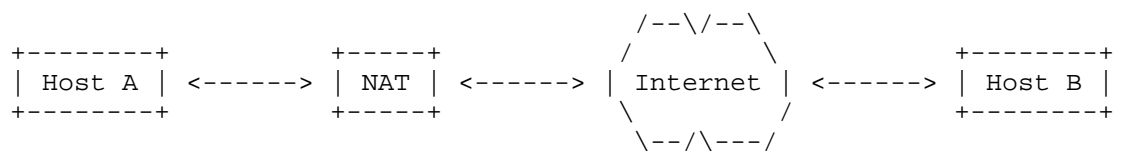
```

```

Priv-Addr:Int-Port <----- Ext-Addr:Ext-Port
Ext-VTag

```

The processing of other incoming SCTP packets is described in the following figure.



```

Pub-Addr: Int-Port <----- Ext-Addr: Ext-Port
                          Int-VTag

```

```

Lookup(Int-VTag, Int-Port, *, *, Ext-Port)

```

```

Returns(NAT-State control block containing Local-Address)

```

```

Priv-Addr: Int-Port <----- Ext-Addr: Ext-Port
                          Int-VTag

```

For an incoming packet containing an INIT-chunk a table lookup is made only based on the addresses and port numbers. If an entry with an External-VTag of zero is found, it is considered a match and the External-VTag is updated.

This allows the handling of INIT-collision through NAT.

7. NAT to SCTP

This document at various places discusses the sending of specialized SCTP chunks (e.g. an ABORT with M-Bit set). These chunks and procedures are not defined in this document, but instead are defined in [I-D.ietf-tsvwg-natsupp]. The NAT implementer should refer to [I-D.ietf-tsvwg-natsupp] for detailed descriptions of packet formats and procedures.

8. Handling of Fragmented SCTP Packets

A NAT box MUST support IP reassembly of received fragmented SCTP packets. The fragments may arrive in any order.

When an SCTP packet has to be fragmented by the NAT box and the IP header forbids fragmentation a corresponding ICMP packet SHOULD be sent.

9. Various Examples of NAT Traversals

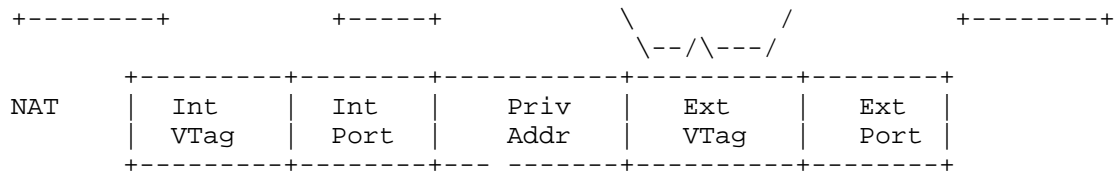
9.1. Single-homed Client to Single-homed Server

The internal client starts the association with the external server via a four-way-handshake. Host A starts by sending an INIT chunk.

```

+-----+          +-----+          /--\ /--\          +-----+
| Host A | <-----> | NAT   | <-----> | Internet | <-----> | Host B |

```

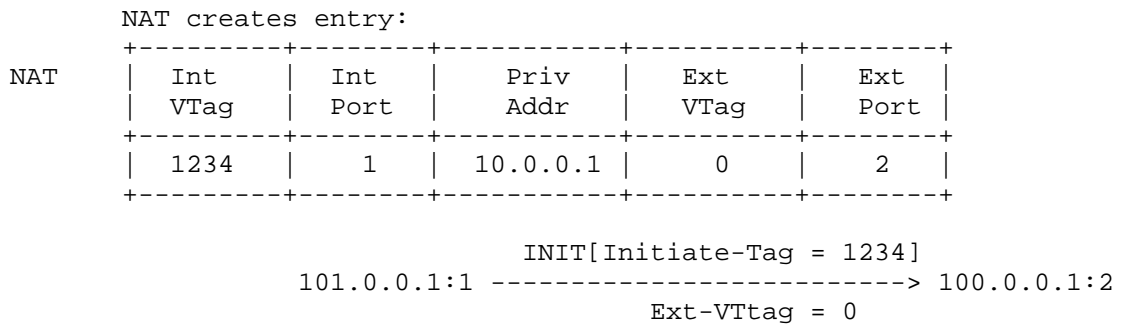


```

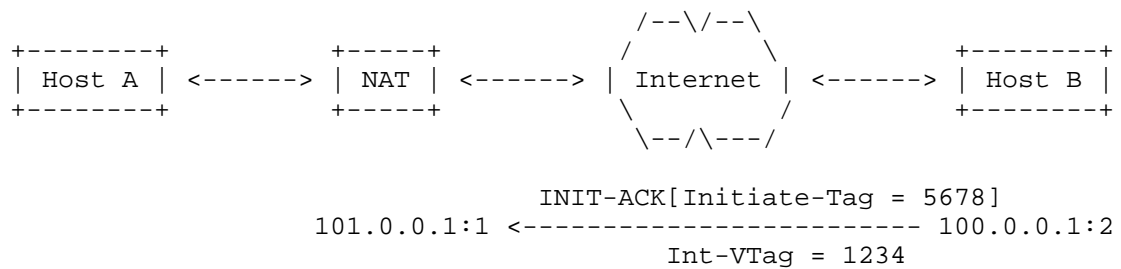
INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 100.0.0.1:2
    Ext-VTtag = 0

```

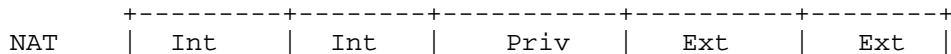
A NAT entry is created, the source address is substituted and the packet is sent on:



Host B receives the INIT and sends an INIT-ACK with the NAT's external address as destination address.



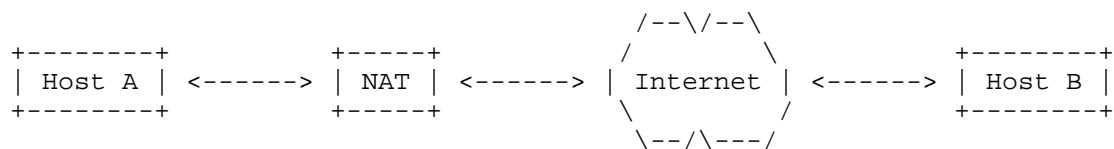
NAT updates entry:



VTag	Port	Addr	VTag	Port
1234	1	10.0.0.1	5678	2

```
INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



```
      COOKIE-ECHO
10.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

```

                                COOKIE-ECHO
101.0.0.1:1 -----> 100.0.0.1:2
                                Ext-VTag = 5678
```

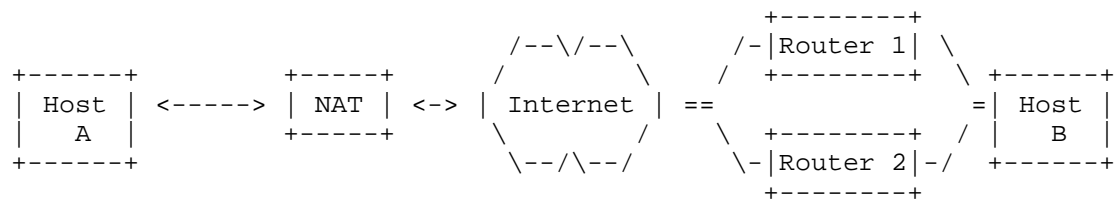
```

                                COOKIE-ACK
101.0.0.1:1 <----- 100.0.0.1:2
                                Int-VTag = 1234
```

```
      COOKIE-ACK
10.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

9.2. Single-homed Client to Multi-homed Server

The internal client is single-homed whereas the external server is multi-homed. The client (Host A) sends an INIT like in the single-homed case.



NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port

```

INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 100.0.0.1:2
      Ext-VTag = 0

```

NAT creates entry:

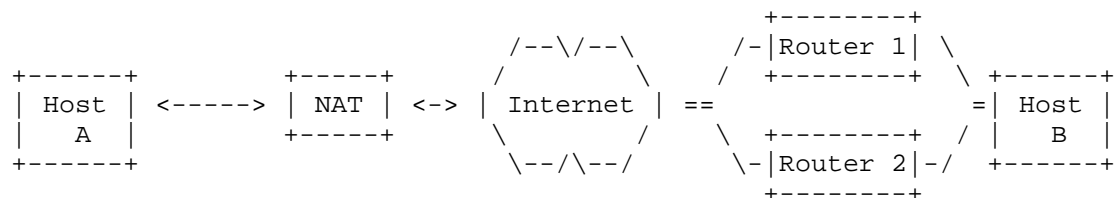
NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	0	2

```

                                INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
                                Ext-VTag = 0

```

The server (Host B) includes its two addresses in the INIT-ACK chunk, which results in two NAT entries.



```

      INIT-ACK[Initiate-tag = 5678, IP-Addr = 100.1.0.1]
101.0.0.1:1 <----- 100.0.0.1:2
                  Int-VTag = 1234

```

NAT does need to change the table for second address:

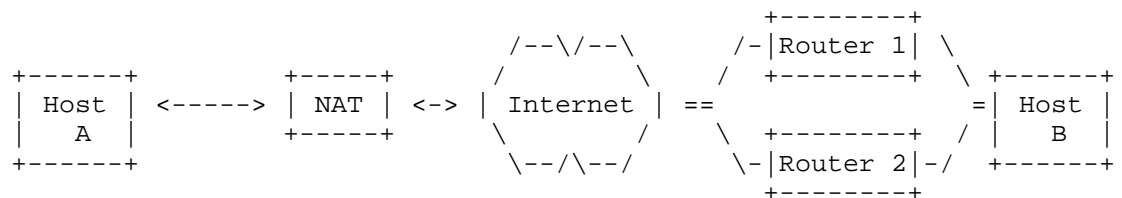
NAT						
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	
	1234	1	10.0.0.1	5678	2	

```

INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <--- 100.0.0.1:2
      Int-VTag = 1234

```

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



```

      COOKIE-ECHO
10.0.0.1:1 ---> 100.0.0.1:2
      ExtVTag = 5678

```

```

                                     COOKIE-ECHO
101.0.0.1:1 -----> 100.0.0.1:2
                                     Ext-VTag = 5678

```

```

                                     COOKIE-ACK
101.0.0.1:1 <----- 100.0.0.1:2
                                     Int-VTag = 1234

```

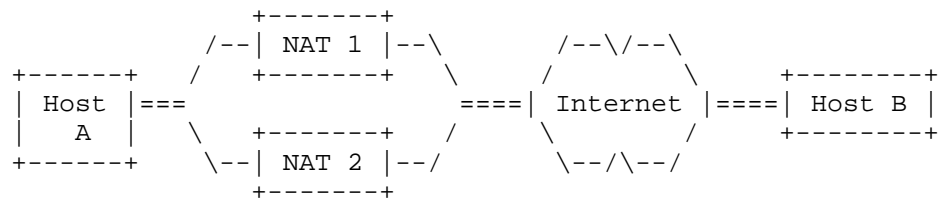
```

      COOKIE-ACK
10.0.0.1:1 <--- 100.0.0.1:2
      Int-VTag = 1234

```

9.3. Multihomed Client and Server

The client (Host A) sends an INIT to the server (Host B), but does not include the second address.



NAT 1	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port

```

      INIT[Initiate-Tag = 1234]
10.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 0

```

NAT 1 creates entry:

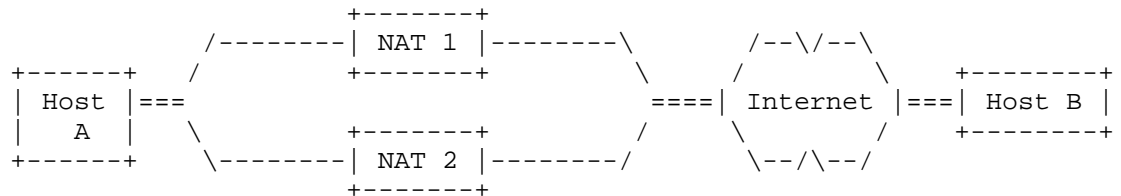
NAT 1	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	0	2

```

      INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
      ExtVTag = 0

```

Host B includes its second address in the INIT-ACK, which results in two NAT entries in NAT 1.



```

INIT-ACK[Initiate-Tag = 5678, IP-Addr = 100.1.0.1]
101.0.0.1:1 <----- 100.0.0.1:2
                    Int-VTag = 1234
  
```

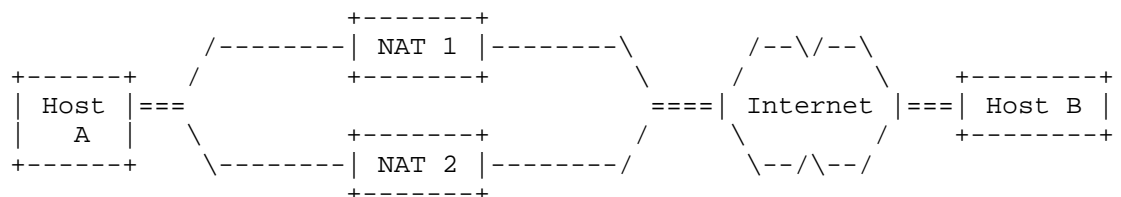
NAT 1 does not need to update the table for second address:

NAT 1	Int	Int	Priv	Ext	Ext
	VTag	Port	Addr	VTag	Port
	1234	1	10.0.0.1	5678	2

```

INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <-----100.0.0.1:2
                    Int-VTag = 1234
  
```

The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



COOKIE-ECHO


```
10.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

```

                                COOKIE-ECHO
101.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

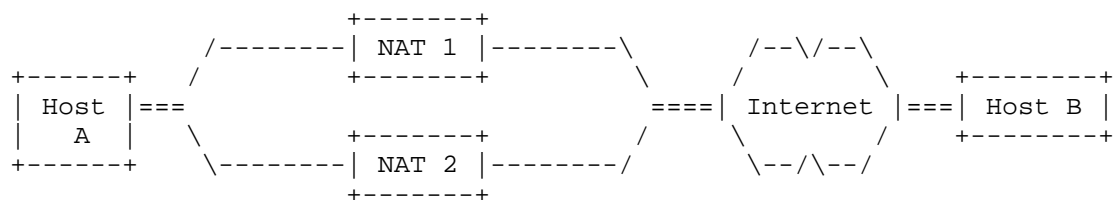
```

                                COOKIE-ACK
101.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

```

                                COOKIE-ACK
10.0.0.1:1 <----- 100.0.0.1:2
      Int-VTag = 1234
```

Host A announces its second address in an ASCONF chunk. The address parameter contains an undefined address (0) to indicate that the source address should be added. The lookup address parameter within the ASCONF chunk will also contain the pair of VTags (external and internal) so that the NAT may populate its table completely with this single packet.



```

ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Ext-VTag = 5678]
10.1.0.1:1 -----> 100.1.0.1:2
      Ext-VTag = 5678
```

NAT 2 creates complete entry:

NAT 2					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.1.0.1	5678	2

```

+-----+-----+-----+-----+-----+
ASCONF [ADD-IP,Int-VTag=1234, Ext-VTag = 5678]
101.1.0.1:1 -----> 100.1.0.1:2
                        Ext-VTag = 5678

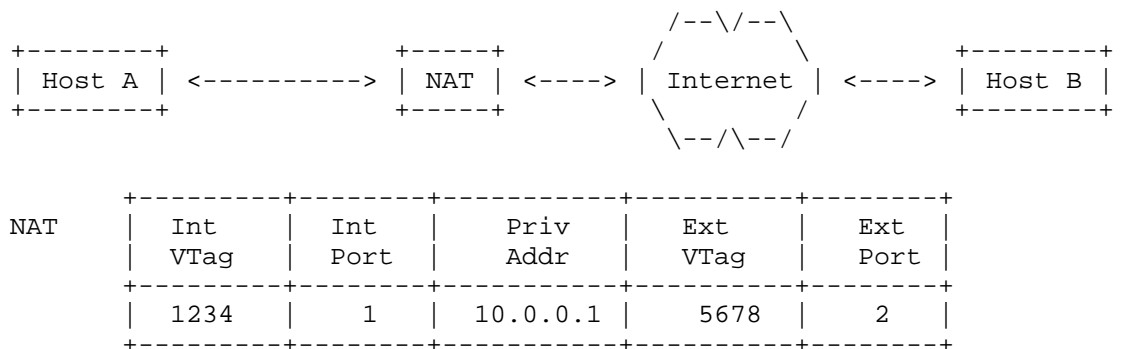
                        ASCONF-ACK
101.1.0.1:1 <----- 100.1.0.1:2
                        Int-VTag = 1234

ASCONF-ACK
10.1.0.1:1 <----- 100.1.0.1:2
                        Int-VTag = 1234

```

9.4. NAT Loses Its State

Association is already established between Host A and Host B, when the NAT loses its state and obtains a new public address. Host A sends a DATA chunk to Host B.



```

DATA
10.0.0.1:1 -----> 100.0.0.1:2
                        Ext-VTag = 5678

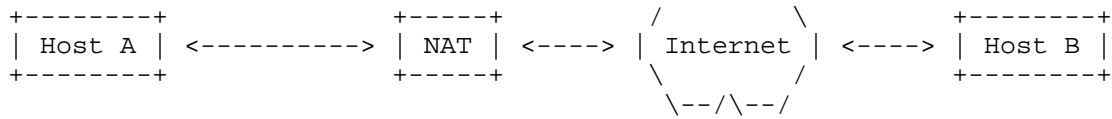
```

The NAT box cannot find entry for the association. It sends ERROR message with the M-Bit set and the cause "NAT state missing".

```

/--\ /--\

```

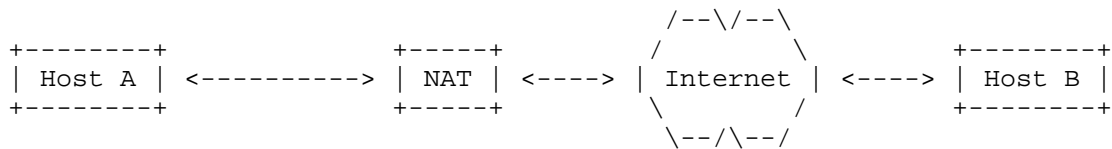


```

ERROR [M-Bit, NAT state missing]
10.0.0.1:1 <----- 100.0.0.1:2
      Ext-VTag = 5678

```

On reception of the ERROR message, Host A sends an ASCONF chunk indicating that the former information has to be deleted and the source address of the actual packet added.



```

ASCONF [ADD-IP,DELETE-IP,Int-VTag=1234, Ext-VTag = 5678]
10.0.0.1:1 -----> 100.1.0.1:2
      Ext-VTag = 5678

```

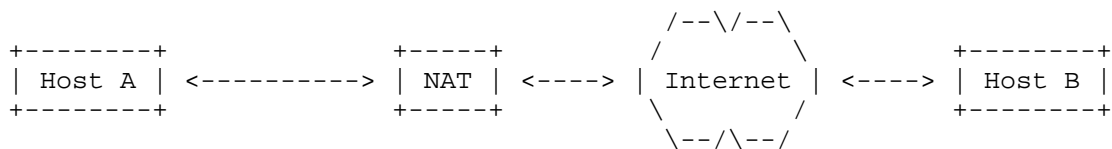
NAT	+-----+-----+-----+-----+-----+					
	Int	Int	Priv	Ext	Ext	
	VTag	Port	Addr	VTag	Port	
	+-----+-----+-----+-----+-----+					
	1234	1	10.0.0.1	5678	2	
	+-----+-----+-----+-----+-----+					

```

ASCONF [ADD-IP,DELETE-IP,Int-VTag=1234, Ext-VTag = 5678]
      102.1.0.1:1 -----> 100.1.0.1:2
                        Ext-VTag = 5678

```

Host B adds the new source address and deletes all former entries.



```

                                ASCONF-ACK
102.1.0.1:1 <----- 100.1.0.1:2
                                Int-VTag = 1234

                                ASCONF-ACK
10.1.0.1:1 <----- 100.1.0.1:2
                                Int-VTag = 1234

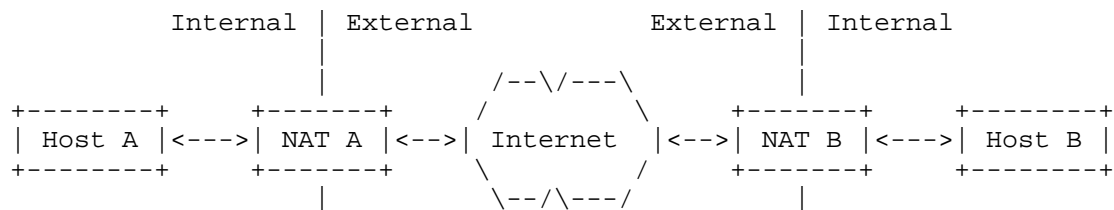
                                DATA
10.0.0.1:1 -----> 100.0.0.1:2
                                Ext-VTag = 5678

                                DATA
102.1.0.1:1 -----> 100.1.0.1:2
                                Ext-VTag = 5678

```

9.5. Peer-to-Peer Communication

If two hosts are behind NATs, they have to get knowledge of the peer's public address. This can be achieved with a so-called rendezvous server. Afterwards the destination addresses are public, and the association is set up with the help of the INIT collision. The NAT boxes create their entries according to their internal peer's point of view. Therefore, NAT A's Internal-VTag and Internal-Port are NAT B's External-VTag and External-Port, respectively. The naming of the verification tag in the packet flow is done from the sending peer's point of view.



NAT-Tables

NAT A						
	Int	Int	Priv	Ext	Ext	
	VTag	Port	Addr	VTag	Port	
NAT B						
	Int	Int	Priv	Ext	Ext	
	v-tag	port	addr	v-tag	port	

```

+-----+-----+--- +-----+-----+
INIT[Initiate-Tag = 1234]
10.0.0.1:1 --> 100.0.0.1:2
      Ext-VTag = 0

```

NAT A creates entry:

NAT A					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	0	2

```

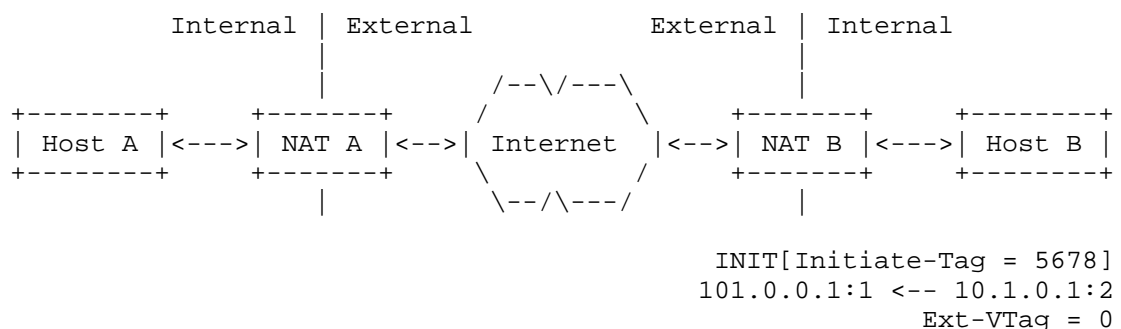
              INIT[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
              Ext-VTag = 0

```

NAT B processes INIT, but cannot find an entry. The SCTP packet is silently discarded and leaves the NAT table of NAT B unchanged.

NAT B					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port

Now Host B sends INIT, which is processed by NAT B. Its parameters are used to create an entry.



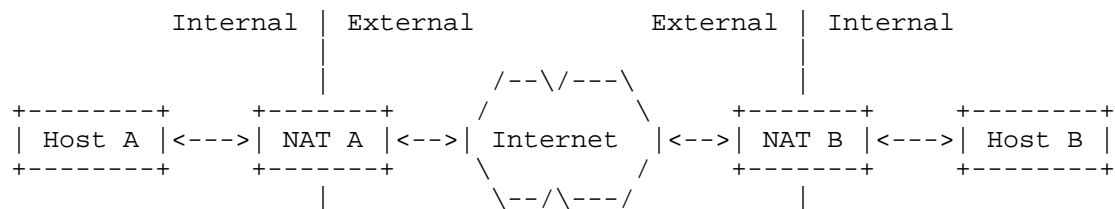
NAT B	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	5678	2	10.1.0.1	0	1

```

INIT[Initiate-Tag = 5678]
101.0.0.1:1 <----- 100.0.0.1:2
                Ext-VTag = 0

```

NAT A processes INIT. As the outgoing INIT of Host A has already created an entry, the entry is found and updated:



VTag != Int-VTag, but Ext-VTag == 0, find entry.

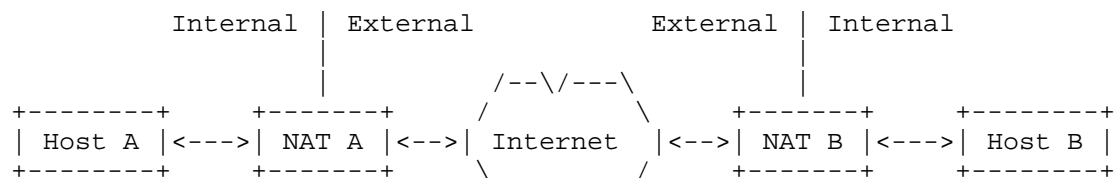
NAT A	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port
	1234	1	10.0.0.1	5678	2

```

INIT[Initiate-tag = 5678]
10.0.0.1:1 <-- 100.0.0.1:2
      Ext-VTag = 0

```

Host A send INIT-ACK, which can pass through NAT B:



```

|                               \--/\---/                               |
INIT-ACK[Initiate-Tag = 1234]
10.0.0.1:1 --> 100.0.0.1:2
    Ext-VTag = 5678

```

```

INIT-ACK[Initiate-Tag = 1234]
101.0.0.1:1 -----> 100.0.0.1:2
    Ext-VTag = 5678

```

NAT B updates entry:

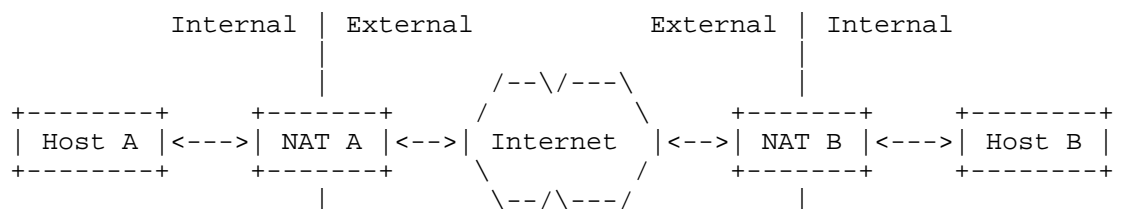
NAT B	+-----+-----+-----+-----+-----+					
	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	
	5678	2	10.1.0.1	1234	1	

```

INIT-ACK[Initiate-Tag = 1234]
101.0.0.1:1 --> 10.1.0.1:2
    Ext-VTag = 5678

```

The lookup for COOKIE-ECHO and COOKIE-ACK is successful.



```

COOKIE-ECHO
101.0.0.1:1 <-- 10.1.0.1:2
    Ext-VTag = 1234

```

```

COOKIE-ECHO
101.0.0.1:1 <----- 100.0.0.1:2
    Ext-VTag = 1234

```

```

COOKIE-ECHO
10.0.0.1:1 <-- 100.0.0.1:2
    Ext-VTag = 1234

```

```
      COOKIE-ACK
10.0.0.1:1 --> 100.0.0.1:2
      Ext-VTag = 5678
```

```
      COOKIE-ACK
101.0.0.1:1 -----> 100.0.0.1:2
      Ext-VTag = 5678
```

```
      COOKIE-ACK
101.0.0.1:1 --> 10.1.0.1:2
      Ext-VTag = 5678
```

10. IANA Considerations

This document requires no actions from IANA.

11. Security Considerations

State maintenance within a NAT is always a subject of possible Denial Of Service attacks. This document recommends that at a minimum a NAT runs a timer on any SCTP state so that old association state can be cleaned up.

12. Acknowledgments

The authors wish to thank Jason But Bryan Ford, David Hayes, Alfred Hines, Henning Peters, Timo Voelker, Dan Wing, and Qiaobing Xie for their invaluable comments.

13. References

13.1. Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [I-D.ietf-tsvwg-natsupp]

Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation Support", draft-ietf-tsvwg-natsupp-05 (work in progress), February 2013.

13.2. Informative References

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

Authors' Addresses

Randall R. Stewart
Adara Networks
Chapin, SC 29036
US

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Irene Ruengeler
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: i.ruengeler@fh-muenster.de

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 6, 2013

J. Yamaguchi
Fiber 26 Network
Y. Shirasaki
S. Miyakawa
NTT Communications
A. Nakagawa
Japan Internet Exchange (JPIX)
H. Ashida
IS Consulting G.K.
July 5, 2012

NAT444 addressing models
draft-shirasaki-nat444-isp-shared-addr-08

Abstract

This document describes addressing models of NAT444. There are some addressing models of NAT444. The addressing models have some issues of network behaviors, operations, and addressing. This document helps network architects to use NAT444 after IPv4 address exhaustion.

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 6, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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

Table of Contents

1. Introduction	4
2. Addressing Models	4
2.1. Global Address	4
2.2. Private Address	4
2.2.1. Policy Based Routing Issue	4
2.2.2. Address Block Duplication Issue	4
2.2.3. Class-E Address (240/4)	4
2.2.4. ISP Shared Address	5
3. Example Architectures	5
3.1. Direct Routing inside CGN	5
3.2. CGN Bypassing	6
3.3. Global Address Customers inside CGN	7
4. Acknowledgements	7
5. IANA Considerations	8
6. Security Considerations	8
7. Normative References	8
Authors' Addresses	8

1. Introduction

NAT444 [I-D.shirasaki-nat444] is one of solutions after IPv4 address exhaustion. ISP can select some addressing models of NAT444. The addressing models have some issues of network behaviors, operations, and addressing. This document describes these issues and solutions. It boosts up to deploy the IPv6 Internet.

2. Addressing Models

The key of addressing model is the address block between Customer Premises Equipment (CPE) and Carrier Grade NAT (CGN) [I-D.ietf-behave-lsn-requirements]. It's mentioned in this section. The best addressing model is "ISP Shared Address" which is defined in [I-D.shirasaki-isp-shared-addr] and briefly described in this section.

2.1. Global Address

ISP cannot assign IPv4 Global Address any more after the exhaustion.

2.2. Private Address

It has two major problems.

2.2.1. Policy Based Routing Issue

If both source and destination address of the packet are inside CGN, it has to go through CGN. The reason is that some servers reject receiving packets when the source address of receiving packet is Private Address. Therefore packets have to go through the CGN for rewriting the source address from Private Address to Global Address. Additionally, if Private Address and Global Address co-exist inside CGN, the ISP has to use Policy Based Routing (PBR).

2.2.2. Address Block Duplication Issue

The Private Address in ISP's network could conflict with its customer's network address. Many CPEs between customer's network and ISP's network cannot route the packet under this situation. To avoid this, ISP has to negotiate with its all customers not to use the reserved Private Address block.

2.2.3. Class-E Address (240/4)

It is known that some equipment such as routers and servers reject packets from or to this address block. So, to use this address block

in ISP's network, ISP has to request its customers to replace their equipment. In addition to that, ISP might have to replace their equipment when it doesn't handle Class-E address packets properly.

2.2.4. ISP Shared Address

ISP Shared Address is the newly defined IPv4 address block that is to be allocated from IANA free pool. It doesn't have any problem. Spending some blocks from the exhausting IANA free pool could be regarded as a problem, but from long view, this problem is much smaller than its great merit. ISP Shared Address is defined in [I-D.shirasaki-isp-shared-addr].

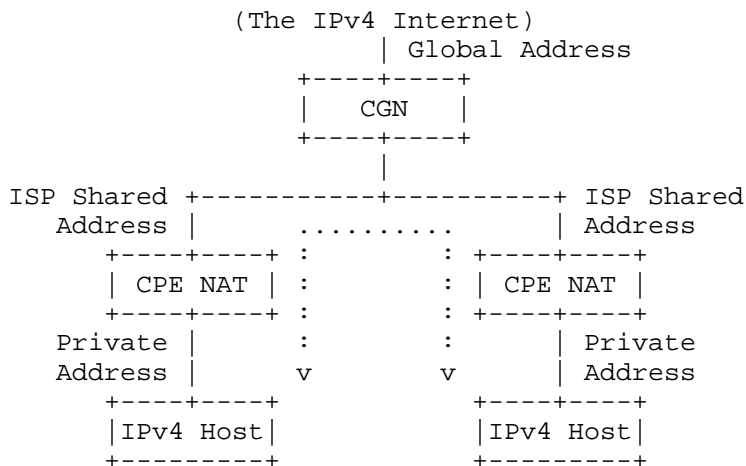
3. Example Architectures

This section explains example architectures how to design NAT444 with ISP Shared Address.

3.1. Direct Routing inside CGN

This architecture enables direct communication between customers inside same CGN. It has the following advantages.

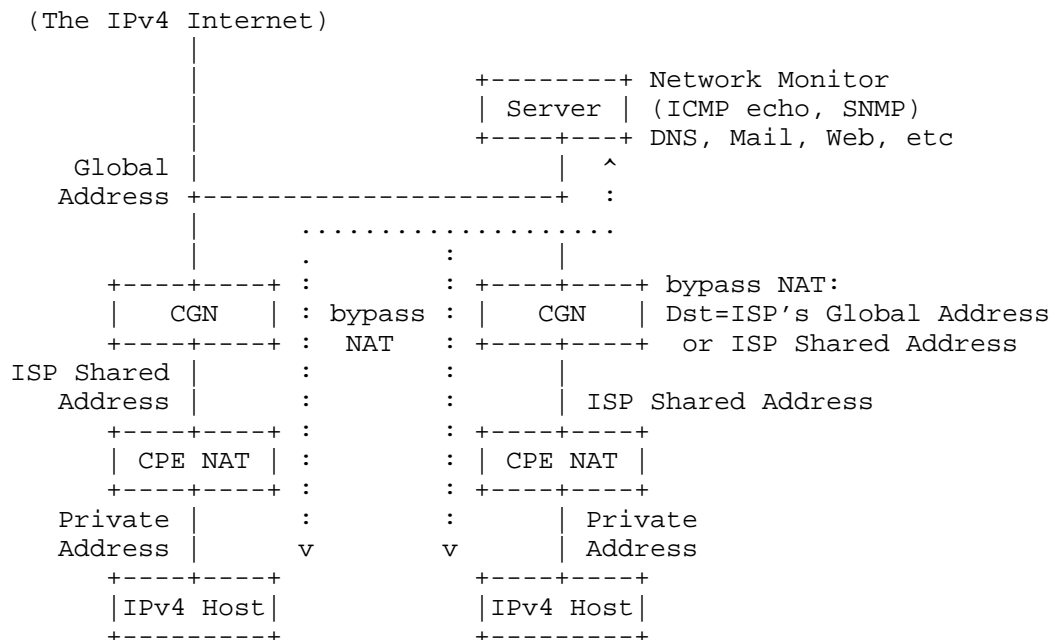
- o The packets don't go through CGN. (No hairpinning)
- o The customers inside CGN can use bidirectional applications (e.g. TV Conference, VPN).
- o No need to use Policy Based Routing.



3.2. CGN Bypassing

This architecture is bypassing the NAT function of CGN. It has the following advantage.

- o The customers inside an ISP can use bidirectional applications (e.g. TV Conference, VPN).
- o Any communication in single ISP doesn't consume CGN external port.
- o ISP's servers outside CGN can access CPE. (e.g. ICMP echo, SNMP, remote access)
- o ISP's servers outside CGN can distinguish which customer's connection it receives. (e.g. DNS, Mail)



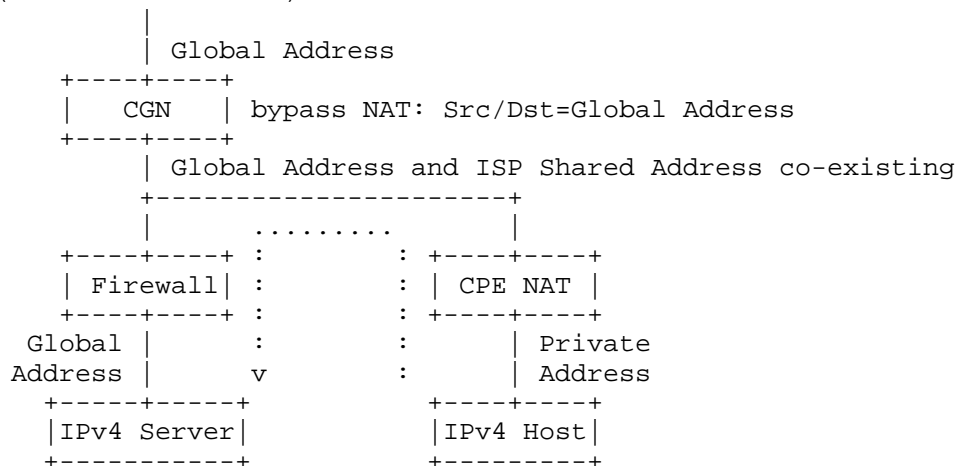
3.3. Global Address Customers inside CGN

This architecture enables co-existing Global Address and ISP Shared Address inside CGN.

It enables direct communications from ISP Shared Address customer to Global Address customer inside same CGN. It has the following advantage.

- o The ISP can put ISP Shared Address customer and Global Address customer in the same concentrator.
- o The customers inside CGN can use bidirectional applications (e.g. TV Conference, VPN).
- o No need to use Policy Based Routing.

(The IPv4 Internet)



4. Acknowledgements

Thanks for the input and review by Shirou Niinobe, Takeshi Tomochika, Tomohiro Fujisaki, Dai Nishino, JP address community members, AP address community members and JPNIC members.

5. IANA Considerations

IANA is to allocate a certain size of address block from IANA free pool. The size of it is described in [I-D.shirasaki-isp-shared-addr]

6. Security Considerations

There are no security considerations.

7. Normative References

[I-D.shirasaki-isp-shared-addr]

Yamagata, I., Miyakawa, S., Nakagawa, A., Yamaguchi, J.,
and H. Ashida, "ISP Shared Address",
draft-shirasaki-isp-shared-addr-07 (work in progress),
January 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-07 (work in
progress), June 2012.

[I-D.shirasaki-nat444]

Yamagata, I., Shirasaki, Y., Nakagawa, A., Yamaguchi, J.,
and H. Ashida, "NAT444", draft-shirasaki-nat444-05 (work
in progress), January 2012.

Authors' Addresses

Jiro Yamaguchi
Fiber 26 Network Inc.
Haraguchi bldg., 5F, 3-11-4 Kanda Jinbo-cho, Chiyoda-ku
Tokyo 101-0051
Japan

Phone: +81 50 3463 6109
Email: jiro-y@f26n.jp

Yasuhiro Shirasaki
NTT Communications Corporation
NTT Hibiya Bldg. 7F, 1-1-6 Uchisaiwai-cho, Chiyoda-ku
Tokyo 100-8019
Japan

Phone: +81 3 6700 8530
Email: yasuihiro@nttv6.jp

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

Phone: +81 3 6733 8671
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

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: December 2, 2011

R. Bush, Ed.
Internet Initiative Japan
May 31, 2011

The A+P Approach to the IPv4 Address Shortage
draft-ymbk-aplusp-10

Abstract

We are facing the exhaustion of the IANA IPv4 free IP address pool. Unfortunately, IPv6 is not yet deployed widely enough to fully replace IPv4, and it is unrealistic to expect that this is going to change before the depletion of IPv4 addresses. Letting hosts seamlessly communicate in an IPv4-world without assigning a unique globally routable IPv4 address to each of them is a challenging problem.

This draft proposes an IPv4 address sharing scheme, treating some of the port number bits as part of an extended IPv4 address (Address plus Port, or A+P). Instead of assigning a single IPv4 address to a single customer device, we propose to extend the address field by using bits from the port number range in the TCP/UDP header as additional end point identifiers, thus leaving a reduced range of ports available to applications. This means assigning the same IPv4 address to multiple clients (e.g., CPE, mobile phones), each with its assigned port-range. In the face of IPv4 address exhaustion, the need for addresses is stronger than the need to be able to address thousands of applications on a single host. If address translation is needed, the end-user should be in control of the translation process - not some smart boxes in the core.

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 December 2, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Problems with Carrier Grade NATs	4
2. Terminology	5
3. Design Constraints and Functions	6
3.1. Design Constraints	6
3.2. A+P Functions	7
3.3. Overview of the A+P Solution	8
3.3.1. Signaling	9
3.3.2. Address Realm	11
3.3.3. Reasons for Allowing Multiple A+P Gateways	15
3.3.4. Overall A+P Architecture	17
3.4. A+P experiments	17
4. Stateless A+P Mapping Function	18
4.1. Stateless A+P Mapping gateway (SMAP) Function description	18
4.2. Implementation Mode	20
4.3. Towards IPv6-only Networks	22
4.4. PRR: On Stateless and Binding Table Modes	22
4.5. General recommendations on SMAP	23
5. Deployment Scenarios	24
5.1. A+P Deployment Models	24
5.1.1. A+P for Broadband Providers	24
5.1.2. A+P for Mobile Providers	24
5.1.3. A+P from the Provider Network Perspective	25
5.2. Dynamic Allocation of Port Ranges	27
5.3. Example of A+P-forwarded Packets	28
5.3.1. Forwarding of Standard Packets	33
5.3.2. Handling ICMP	33
5.3.3. Fragmentation	34
5.3.4. Limitations of the A+P approach	34
5.3.5. Port allocation strategy agnostic	35
6. IANA Considerations	35
7. Security Considerations	35
8. Authors	36
9. Acknowledgments	37
10. References	38
10.1. Normative References	38
10.2. Informative References	38
Author's Address	40

1. Introduction

This document describes a technique to deal with the imminent IPv4 address space exhaustion. Many large Internet Service Providers (ISPs) face the problem that their networks' customer edges are so large that it will soon not be possible to provide each customer with a unique public IPv4 address. Therefore, although undesirable, address sharing, in the same molds as NAT, is inevitable.

To allow end-to-end connectivity between IPv4 speaking applications we propose to extend the semantics of the IPv4 address with bits from the UDP/TCP header. Assuming we could limit the applications' port addressing to any number of bits lower than 16, we can increase the effective size of an IPv4 address by remaining additional bits of up to 16. In this scenario, 1 to 65536 customers could be multiplexed on the same IPv4 address, while allowing them a fixed or dynamic range of 1 to 65536 ports. Customers could for example receive initial fixed port range, defined by operator and dynamically request additional blocks, depending on their contract. We call this "extended addressing" or "A+P" (Address plus Port) addressing. The main advantage of A+P is that it preserves the Internet "end-to-end" paradigm by not requiring translation (at least for some ports) of an IP address.

1.1. Problems with Carrier Grade NATs

Various forms of NATs will be installed at various levels and places in the IPv4-Internet to achieve address compression. This document argues for mechanisms where this happens as close to the edge of the network as possible, thereby minimizing damage to the End-to-End Principle and allowing end-customers to retain control over the address and port translation. Therefore it is essential to create mechanisms to "bypass" NATs in the core when applicable and keep the control at the end-user.

With Carrier Grade NATs in the core of the network the user is trapped behind unchangeable application policies, and the deployment of new applications is hindered by the need to implement the corresponding Application Level Gateways (ALGs) on the CGNs. This is the opposite of the "end-to-end" model of the Internet.

With the smarts at the edges, one can easily deploy new applications between consenting end-points by merely tweaking the NATs at the corresponding Customer Premises Equipment (CPE), or even upgrading them to a new version that supports a specific ALG.

Today's NATs are typically mitigated by offering the customers limited control over them, e.g. port forwarding or UPnP/NAT-PMP.

However, this is not expected to work with CGNs. CGN proposals - other than DS-Lite [I-D.ietf-softwire-dual-stack-lite] with A+P or PCP [I-D.ietf-pcp-base]- admit that it is not expected that applications that require specific port assignment or port mapping from the NAT box will keep working.

Another issue with CGN is the trade-off between session state and network placement. The furthest from the edge the CGN placed, the more session state needs to be kept due to larger subscriber aggregation, and more disruption in the case of a failure. In order to reduce the state, CGNs would end up somewhere closer to the edge. The CGN hence trades scalability for the amount of state that needs to be kept, which makes optimally placing a CGN a hard engineering problem

In some deployment scenarios, CGN can be seen as single point of failure and therefore the availability of delivered services are impacted by the ones of CGN s devices. Means to ensure state synchronisation and failover would be required to allow for service continuity whenever a failure occurs.

Intra-domain paths may not be optimal for communications between two nodes connected to the same domain deploying CGNs, hence leading to path stretches.

2. Terminology

This document makes use of the following terms:

Public Realm: This realm contains only public routable IPv4 addresses. Packets in this realm are forwarded based on the destination IPv4 address

A+P Realm: This realm contains both public routable IPv4 and also A+P addresses.

A+P Packet: A regular IPv4 packet is forwarded based on the destination IPv4 address and the TCP/UDP port numbers.

Private Realm: This realm contains IPv4 addresses that are not globally routed. They may be taken from the [RFC1918] range. However, this document does not make such an assumption. We regard as private address space any IPv4 address, which needs to be translated in order to gain global connectivity, irrespective of whether it falls in [RFC1918] space or not.

Port Range Router (PRR): A device that forwards A+P packets.

Customer Premises Equipment (CPE): cable/DSL modem.

Provider Edge Router (PE): Customer aggregation router

Provider Border Router (BR): Providers edge to other providers

Network Core Routers (Core): Provider routers which are not at the edges.

3. Design Constraints and Functions

The problem of address space shortage is first felt by providers with a very large end-user customer base, such as broadband providers and mobile service providers. Though the cases and requirements are slightly different, they share many commonalities. In the following we develop a set of overall design constraints for solutions addressing the IPv4 address shortage problem.

3.1. Design Constraints

We regard several constraints as important for our design:

- 1) End-to-End is under customer control: Customers shall have the ability to deploy new application protocols at will. IPv4 address shortage should not be a license to break the Internet's end-to-end paradigm.
- 2) Backward compatibility: Approaches should be transparent to unaware users. Devices or existing applications should be able to work without modification. Emergence of new applications should not be limited.
- 3) Highly-scalable and minimal state core: Minimal state should be kept inside the ISP's network. If the operator is rolling out A+P incrementally, it is understood there may be state in the core in the non-A+P part of such a roll-out.
- 4) Efficiency vs. complexity: Operators should have the flexibility to trade off port multiplexing efficiency and scalability and end-to-end transparency.
- 5) "Double-NAT" should be avoided: Multiple gateway devices might be present in a path, and once one has done some translation, those packets should not be re-translated.

- 6) Legal traceability: ISPs must be able to provide the identity of a customer from the knowledge of the IPv4 public address and the port. This should have as low an impact as is reasonable on storage by the ISP. We assume that NATs on customer premises do not pose much of a problem, while provider NATs need to keep additional logs.
- 7) IPv6 deployment should be encouraged. NAT444 strongly biases the users to the deployment of RFC 1918 addressing.

Constraint 5 is important: while many techniques have been deployed to allow applications to work through a NAT, traversing cascaded NATs is crucial if NATs are being deployed in the core of a provider network.

3.2. A+P Functions

The A+P architecture can be split into three distinct functions: encaps/decaps, NAT, and signaling.

Encaps/decaps function: is used to forward port-restricted A+P-packets over intermediate legacy devices. The encapsulation function takes an IPv4 packet, looks up the IP and TCP/UDP headers, and puts the packet into the appropriate tunnel. The state needed to perform this action is comparable to a forwarding table. The decapsulation device SHOULD check if the source address and port of packets coming out of the tunnel are legitimate (e.g., see [BCP38]). Based on the result of such a check, the packet MAY be forwarded untranslated, it MAY be discarded or MAY be NATed. In this document we refer to a device that provides this encaps/decaps functionality as Port-Range-Router (PRR).

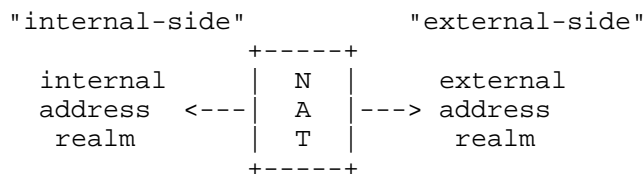
Network Address Translation (NAT) function: is used to connect legacy end-hosts. Unless upgraded, end-hosts or end-systems are not aware of A+P restrictions and therefore assume a full IP address. The NAT function performs any address or port translation, including Application Level Gateways (ALGs) whenever required. The state that has to be kept to implement this function is the mapping for which external addresses and ports have been mapped to which internal addresses and ports, just as in CPEs embedding NAT today. A subtle, but very important, difference should be noted here: the customer has control over the NATing process or might choose to "bypass" the NAT. If this is done, we call the NAT a large scale NAT (LSN). However, if the NAT that does NOT allow the customer to control the translation process, we refer to as a CGN.

Signaling function: is used in order to allow A+P-aware devices get to know which ports are assigned to be passed through untranslated

and what will happen to packets outside the assigned port-range (e.g., could be NATed or discarded). Signaling may also be used to learn the encapsulation method and any endpoint information needed. In addition, the signaling function may be used to dynamically assign the requested port-range.

3.3. Overview of the A+P Solution

As mentioned above, the core architectural elements of the A+P solution are three separated and independent functions: the NAT function, the encaps/decaps function, and the signaling function. The NAT function is similar to a NAT as we know it today: it performs a translation between two different address realms. When the external realm is public IPv4 address space, we assume that the translation is many-to-one, in order to multiplex many customers on a single public IPv4 address. The only difference with a traditional NAT (Figure 1) is that the translator might only be able to use a restricted range of ports when mapping multiple internal addresses onto an external one, e.g., the external address realm might be port-restricted.



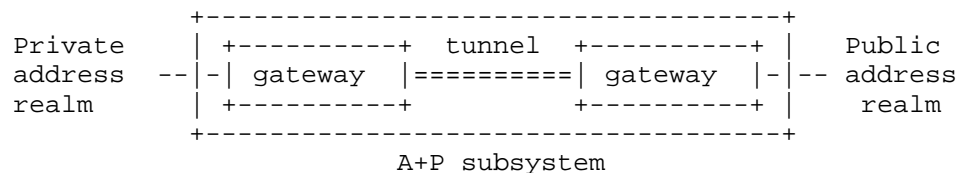
Traditional NAT

Figure 1

The encaps/decaps function, on the other hand, is the ability to establish a tunnel with another end-point providing the same function. This implies some form of signaling to establish a tunnel. Such signaling can be viewed as integrated with DHCP or as a separate service. Section 3.3.1 discusses the constraints of this signaling function. The tunnel can be an IPv6 or IPv4 encapsulation, a layer-2 tunnel, or some other form of software. Note that the presence of a tunnel allows unmodified, naive, or even legacy devices between the two endpoints.

Two or more devices which provide the encaps/decaps function and are linked by tunnels to form an A+P subsystem. The function of each gateway is to encapsulate and decapsulate respectively. Figure 2 depicts the simplest possible A+P subsystem, that is, two devices

providing the encaps/decaps function.



A simple A+P subsystem

Figure 2

Within an A+P subsystem, the public address realm is extended by using bits from the port number when forwarding packets. Each device is assigned one address from the external realm and a range of port numbers. Hence, devices which are part of an A+P subsystem can communicate with the public realm without the need for address translation (i.e., preserving end-to-end packet integrity): an A+P packet originated from within the A+P subsystem can be simply forwarded over tunnels up to the endpoint, where it gets decapsulated and routed in the external realm.

3.3.1. Signaling

The following information needs to be available on all the gateways in the A+P subsystem. It is expected that there will be a signaling protocols such as [I-D.bajko-pripaddrassign], [I-D.boucadair-dhcpv6-shared-address-option], [I-D.boucadair-pppext-portrange-option], or [I-D.ietf-pcp-base].

The information that needs to be shared is the following:

- o a set of public IPv4 addresses,
- o for each IPv4 address a starting point for the allocated port-range,
- o number of delegated ports,
- o optional key that enables partial or full preservation of entropy in port randomization - see [I-D.bajko-pripaddrassign],
- o lifetime for each IPv4 address and allocated port-set,

- o the tunneling technology to be used (e.g., "IPv6-encapsulation")
- o addresses of the tunnel endpoints (e.g., IPv6 address of tunnel endpoints)
- o whether or not NAT function is provided by the gateway
- o a device identification number and some authentication mechanisms
- o a version number and some reserved bits for future use.

Note that the functions of encapsulation and decapsulation have been separated from the NAT function. However, to accommodate legacy hosts, NATing is likely to be provided at some point in the path; therefore the availability or absence of NATing MUST be communicated in signaling, as A+P is agnostic about NAT placement.

The port-ranges can be allocated in two different ways:

- o If applications or end-hosts behind the CPE are not UPnPv2/NAT-PMP aware, then the CPE SHOULD request ports via mechanisms, e.g. as described in [I-D.bajko-pripaddrassign] and [I-D.boucadair-pppext-portrange-option]. Note that different port-ranges can have different lifetimes, and the CPE is not entitled to use them after they expire - unless it refreshes those ranges. It is up to the ISP to put mechanisms in place, that determine what percentage of already allocated port-ranges should be exhausted before a CPE may request additional ranges, how often the CPE can request additional ranges, and so on. (To prevent Denial of Service attacks.)
- o If applications behind the CPE are UPnPv2/NAT-PMP aware additional ports MAY be requested through that mechanism. In this case the CPE should forward those requests to the LSN and the LSN should reply reporting if the requested ports are available or not (and if they are not available some alternatives should be offered). Here again, to prevent potential denial of service attacks, mechanism should be in place to prevent UPnPv2/NAT-PMP packet storms and fast port allocation. Detailed description of this mechanism, called PCP is described in [I-D.ietf-pcp-base].

Whatever signaling mechanism is used inside the tunnels, DHCP, IPCP, or PCP-based, synchronization between signaling server and PRR must be established in both directions. For example, if we use DHCP as signaling mechanism, the PRR must communicate to DHCP server at least its IP range. The DHCP server then starts to allocate IPs and port-ranges to CPEs and communicates back to the PRR which IP and port range have been allocated to which CPE, so the PRR knows to which

tunnel redirect incoming traffic. In addition, DHCP MUST also communicate lifetimes of port-ranges assigned to CPE via the PRR. DHCP server may be co-located with the PRR function to ease address management and also to avoid the need to introduce a communication protocol between the PRR and DHCP.

If UPnPv2/NAT-PMP is used as dynamic port allocation mechanism, the PRR must also communicate to the DHCP (or IPCP) server to avoid those ports. The PRR must somehow (DHCP or IPCP options) communicate back to CPE that allocation of ports was successful, so CPE adds those ports to existing port ranges.

Note that operation can be even simplified if a fixed length of port ranges are assigned to all customers and no differentiation is implemented based on port range length. In such case, the binding table maintained by the PRR can be dynamically built upon the receipt of a first packet from a port-restricted device.

3.3.2. Address Realm

Each gateway within the A+P subsystem manages a certain portion of A+P address space, that is, a portion of IPv4 space which is extended by borrowing bits from the port number. This address space may be a single, port-restricted IPv4 address. The gateway MAY use its managed A+P address space for several purposes:

- o Allocation of a sub-portion of the A+P address space to other authenticated A+P gateways in the A+P subsystem (referred to as delegation). We call the allocated sub-portion delegated address space.
- o Exchange of (untranslated) packets with the external address realm. For this to work, such packets MUST use source address and port belonging to the non-delegated address space.

If the gateway is also capable of performing the NAT function, it MAY translate packets arriving on an internal interface which are outside of its managed A+P address space into non-delegated address space.

Hence, a provider may have 'islands' of A+P as they slowly deploy over time. The provider does not have to replace CPE until they want to provide the A+P function to an island of users or even to one particular user in a sea of non-A+P users.

An A+P gateway ("A"), accepts incoming connections from other A+P gateways ("B"). Upon connection establishment (provided appropriate authentication), B would "ask" A for delegation of an A+P address. In turn, A will inform B about its public IPv4 address, and will

delegate a portion of its port-range to B. In addition, A will also negotiate the encaps/decaps function with B (e.g., let B know the address of the decaps device/other-end-point of the tunnel).

This could be implemented for example via a NAT-PMP or DHCP-like solution. In general the following rule applies: A sub-portion of the managed A+P address space is delegated as long as devices below ask for it, otherwise private IPv4 is provided to support legacy hosts.

In the following example, IPv4 address reserved for documentation blocks defined in [RFC5737] are used.

```

private      +-----+           +-----+      public
address ---|  B  |=====|  A  |--- Internet
realm      +-----+           +-----+

```

Address space realm of A:
 public IPv4 address = 192.0.2.1
 port range = 0-65535

Address space realm of B:
 public IPv4 address = 192.0.2.1
 port range = 2560-3071

Configuration example

Figure 3

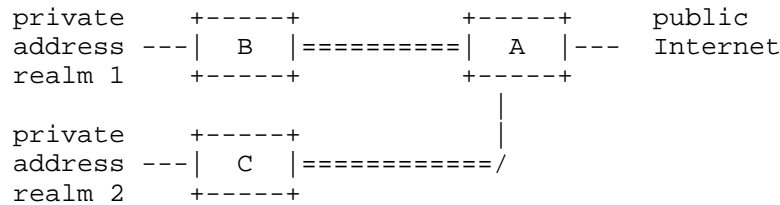
Figure 3 illustrates a sample configuration. Note that A might actually consist of three different devices: one that handles signaling requests from B; one device that performs encapsulation and decapsulation; and, if provided, one device that performs NATing function (e.g., LSN). Packet forwarding is assumed to be as follows: In the "out-bound" case, a packet arrives from the private address realm to B. As stated above, B has two options: it can either apply or not apply the NAT function. The decision depends upon the specific configuration and/or the capabilities of A and B. Note that NAT functionality is required to support legacy hosts, however, this can be done at either of the two devices A or B. The term NAT refers to translating the packet into the managed A+P address (B has address 192.0.2.1 and ports 2560-3071 in the example above). We then have two options:

- 1) B NATs the packet. The translated packet is then tunneled to A. A recognizes that the packet has already been translated, because the source address and port match the delegated space. A decapsulates the packet and releases it in the public Internet.
- 2) B does not NAT the packet. The untranslated packet is then tunneled to A. A recognizes that the packet has not been translated, so A forwards the packet to a co-located NATing device, which translates the packet and routes it in the public Internet. This device, e.g. - an LSN, has to store the mapping between the source port used to NAT and the tunnel where the packet came from, in order to correctly route the reply. Note that A cannot use a port number from the range that has been delegated to B. As a consequence A has to assign a part of its non-delegated address space to the NATing function.

"Inbound" packets are handled in the following way: a packet from the public realm arrives at A. A analyzes the destination port number to understand whether the packet needs to be NATed or not.

- 1) If the destination port number belongs to the range that A delegated to B, then A tunnels the packet to B. B NATs the packet using its stored mapping and forwards the translated packet to the private domain.
- 2) If the destination port number is from the address space of the LSN, then A passes the packet on to the co-located LSN which uses its stored mapping to NAT the packet into the private address realm of B. The appropriate tunnel is stored as well in the mapping of the initial NAT. The LSN then encapsulates the packet to B, which decapsulates it and normally routes it within its private realm.
- 3) Finally, if the destination port number neither falls in a delegated range, nor into the address range of the LSN, A discards the packet. If the packet is passed to the LSN, but no mapping can be found, the LSN discards the packet.

Observe that A must be able to receive all IPv4 packets destined to the public IPv4 address (192.0.2.1 in the example), so that it can make routing decisions according to the port number. On the other hand, B receives IPv4 packets destined to the public IPv4 address only via the established tunnel with A. In other words, B uses the public IPv4 address just for translation purposes, but it is not used to make routing decisions. This allows us to keep the routing logic at B as simple as described above, while enabling seamless communication between A+P devices sharing the same public IPv4 address.



Address space realm of A:
 public IPv4 address = 192.0.2.1
 port range = 0-65535

Address space realm of B:
 public IPv4 address = 192.0.2.1
 port range = 2560-3071

Address space realm of C:
 public IPv4 address = 192.0.2.1
 port range = 0-2559

Hierarchical A+P

Figure 4

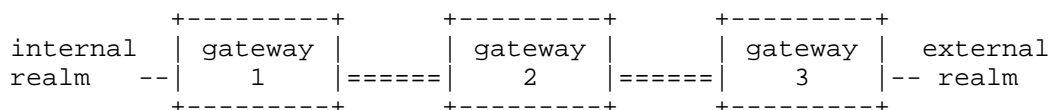
Consider the example shown in Figure 4. Here both B and C use the encaps/decaps function to establish a tunnel with A, and they are assigned the same public IPv4 address with different, non-overlapping port-ranges. Assume that a host in B's private realm sends a packet destined to address 192.0.2.1 and port 2000, and that B has been instructed to NAT all packets destined to 192.0.2.1. Under these assumptions, B receives the packet and NATs it using its own public IPv4 address (192.0.2.1) and a port selected from its configured port-range (e.g., 3000). B then tunnels the translated packet to A. When A receives the packet via the tunnel, it looks at the destination address and port, recognizes C's delegated range, and then tunnels the packet to C. Observe that, apart from stripping the tunnel header, A handles the packet as if it came from the public Internet. When C receives the packet, it NATs the destination address into one address chosen from its private address realm, while keeping the source address (192.0.2.1) and port (3000) untranslated. Return traffic is handled the same way. Such a mechanism allows hosts behind A+P devices to communicate seamlessly even when they share the same public IPv4 address.

Please refer to Section 4 for a discussion of an alternative A+P mechanism that does not incur in path stretches penalties for intra-domain communication.

3.3.3. Reasons for Allowing Multiple A+P Gateways

Since each device in an A+P subsystem provides the encaps/decaps function, new devices can establish tunnels and become in turn part of an A+P subsystem. As noted above, being part of an A+P subsystem implies the capability of talking to the external address realm without any translation. In particular, as described in the previous section, a device X in an A+P subsystem can be reached from the external domain by simply using the public IPv4 address and a port which has been delegated to X. Figure 5 shows an example where three devices are connected in a chain. In other words, A+P signaling can be used to extend end-to-end connectivity to the devices which are in an A+P subsystem. This allows A+P-aware applications (or OSes) running on end hosts to enter an A+P subsystem and exploit untranslated connectivity.

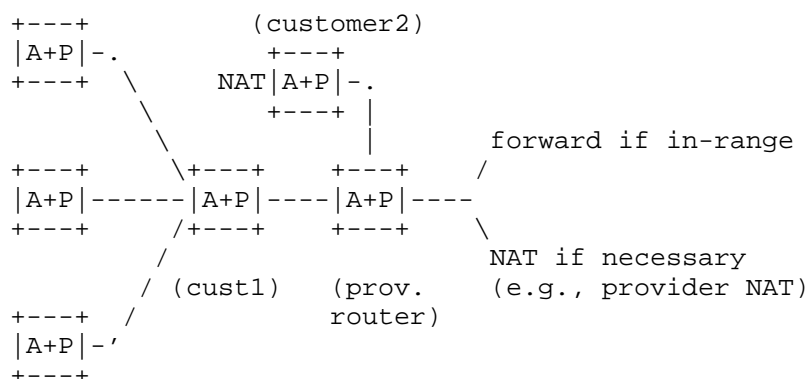
There are two modes for end-hosts to gain fine-grained control of end-to-end connectivity. The first is where actual end-hosts perform the NAT function and the encaps/decaps function which is required to join the A+P subsystem. This option works in a similar way to the NAT-in-the-host trick employed by virtualization software such as VMware, where the guest operating system is connected via a NAT to the host operating system. The second mode is applications which autonomously ask for an A+P address and use it to join the A+P subsystem. This capability is necessary for some applications that require end-to-end connectivity (e.g., applications that need to be contacted from outside).



An A+P subsystem with multiple devices

Figure 5

Whatever the reasons might be, the Internet was built on a paradigm that end-to-end connectivity is important. A+P makes this still possible in a time where address shortage forces ISPs to use NATs at various levels. In such sense, A+P can be regarded as a way to bypass NATs.



A complex A+P subsystem

Figure 6

Figure 6 depicts a complex scenario, where the A+P subsystem is composed by multiple devices organized in a hierarchy. Each A+P gateway decapsulates the packet and then re-encapsulates it again to the next tunnel.

A packet can either be NATed when it enters the A+P subsystem, or at intermediate devices, or when it exits the A+P subsystem. This could be for example a gateway installed within the provider's network, together with a LSN. Then each customer operates its own CPE. However, behind the CPE applications might also be A+P-aware and run their own A+P-gateways, which enables them to have end-to-end connectivity.

One limitation applies, if "delayed translation" is used (e.g., translation at the LSN instead of the CPE). If devices using "delayed translation" want to talk to each other they SHOULD use A+P addresses or out-of-band addressing.

3.3.4. Overall A+P Architecture

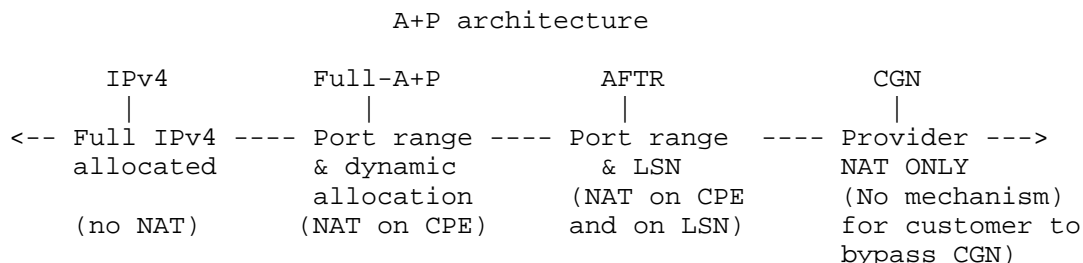


Figure 7: A+P overall architecture

The A+P architecture defines various deployment options within an ISP. Figure 7 shows the spectrum of deployment options. On the far left is the common deployment method for broadband subscribers today, an IPv4 address unrestricted with full port-range. Full-A+P refers to a port-range allocation from the ISP. The customer must operate an A+P-aware CPE device and no NATing functionality is provided by the ISP. AFTR, such as DS-Lite [I-D.ietf-softwire-dual-stack-lite], is a hybrid. There is NAT present in the core (in this document referred to as LSN), but the user has the option to "bypass" that NAT in one form or another, for example via A+P, NAT-PMP, etc... Finally, a service provider which only deploys CGN, will place a NAT in the providers core and does not allow the customer to "bypass" the translation process or modify ALGs on the NAT. The customer is provider-locked. Notice that all options (besides full IPv4) require some form of tunneling mechanism (e.g., 4in6) and a signaling mechanism (see Section 3.3.1).

3.4. A+P experiments

There are implementations of A+P as well as documented experiments. France Telecom did experiments, that are described in [I-D.deng-aplusp-experiment-results]. As seen in that experiment, most tested applications are unaffected. There are problems with torrent protocol and applications, as listening port is out of A+P port range and some UPnP may be required to make it work with A+P

Problems with BitTorrent have already been experienced in the wild by users trapped behind a non-UPnP-capable CPE. The current workaround for the end user is to statically map ports, which can be done in the A+P scenario as well.

Bittorrent tests and experiments in shared IP and port range

environments are well described in [I-D.boucadair-behave-bittorrent-portrange]. Conclusions in that document tell us that two limitations were experienced. The first occurred when two clients sharing the same IP address tried to simultaneously retrieve the SAME file located in a SINGLE remote peer. The second limitation occurred when a client tried to download a file located on several seeders, when those seeders shared the same IP address. Mutual file sharing between hosts having the same IP address has been checked. Indeed, machines having the same IP address can share files with no alteration compared to current IP architectures.

Working implementations of A+P can be found in ISC AFTR (<http://www.isc.org/software/aftr>), FT Orange opensource A+P (<http://opensourceapluspl.us/weebly.com/>) and 4RD from ipinfusion.com (stateless A+P).

4. Stateless A+P Mapping Function

4.1. Stateless A+P Mapping gateway (SMAP) Function description

SMAP stands for Stateless A+P Mapping. This function is responsible to encapsulate (Resp., decapsulate), in a stateless scheme, IPv4 packets in (Resp. from) IPv6 ones. A SMAP function may be hosted in a PRR, end-user device, etc.

As mentioned in Section 4.1 of [RFC6052], the suffix part may enclose the port.

Stateless A+P Mapping gateway (SMAP) consists in two basic functions as described in Figure 8.

1. SMAP encapsulates an IPv4 packet, destined to a shared IPv4 address, in IPv6 one. The IPv6 source address is constructed using an IPv4-Embedded IPv6 address [RFC6052] from the IPv4 source address and port number plus the IPv6 prefix which has been provisioned to the node performing the SMAP function. The destination IPv6 address is constructed using the shared IPv4 destination address and port number plus the IPv6 prefix which has been provisioned to the SMAP function and which is dedicated to IPv4 destination addresses.

2. SMAP extracts IPv4 incoming packets from IPv6 incoming ones which have IPv6 source addresses belonging to the prefix of the node performing the SMAP function. Extracted IPv4 packets are then forwarded to the point identified by the IPv4 destination address and port number.

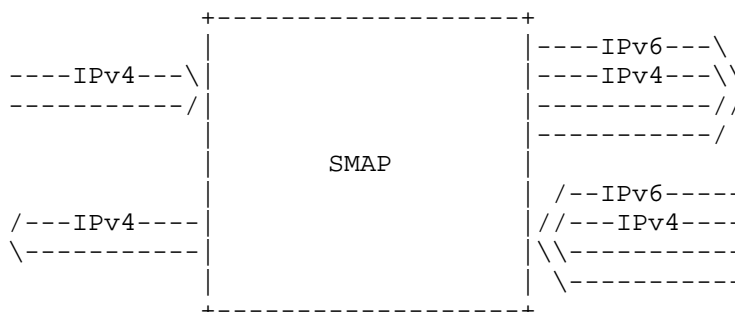


Figure 8: Stateless A+P Mapping Gateway Function

A SMAP-enabled node will perform the stateless 6/4 mapping function for all public shared IPv4 addresses for which it was designated as a stateless 6/4 mapping gateway.

To perform stateless 6/4 mapping function a SMAP gateway must:

- o be provided with an IPv6 prefix (i.e., Pref6). The SMAP gateway uses this prefix to construct IPv6 source addresses for all IPv4 shared addresses for which it was designated as a SMAP gateway. The IPv6 prefix may be provisioned statically or dynamically (e.g., DHCP)

- o be able to know the IPv6 prefix of the node serving as another SMAP gateway for IPv4 destination addresses. This prefix may be known in various ways:

- * Default or Well Known Prefix (i.e., 64:ff9b::/96) which was provisioned statically or dynamically;

- * Retained at the reception of incoming IPv4-in-IPv6 encapsulated packets;

- * Discovered at the communication starting thanks to mechanisms as DNS resolution for example.

When the SMAP-enabled node receives IPv4 packets with IPv4 source addresses for which it was not designated as a SMAP gateway, it will not perform stateless 6/4 mapping function for those packets. Those packets will be handled in a classical way (i.e., forwarded, dropped or locally processed).

When the SMAP-enabled node receives IPv6 packets with IPv6 addresses which do not match with its IPv6 prefix, it will not perform the

stateless 6/4 mapping function for those packets. Those packets will be handled in a classical way (i.e., forwarded, dropped or locally processed).

4.2. Implementation Mode

In this configuration, the node A performs the stateless mapping function on the received IPv4 traffic (encapsulated in IPv6 packets) before forwarding to the node B. The node B performs the stateless mapping function on the received IPv6 traffics (extracting IPv4 packets) before forwarding the IPv4 traffic to the destination identified by the IPv4 destination address and port number. In the opposite direction and as previously, the node B performs the stateless mapping function on the received IPv4 traffics (encapsulating in IPv6 packets) before forwarding to the node A. The node A performs the stateless mapping function on the received IPv6 traffic (extracting IPv4 packets) before forwarding the IPv4 traffic to the point identified by the IPv4 destination address and port number. In this case, only IPv6 traffic is managed in the network segment between the nodes A and B.

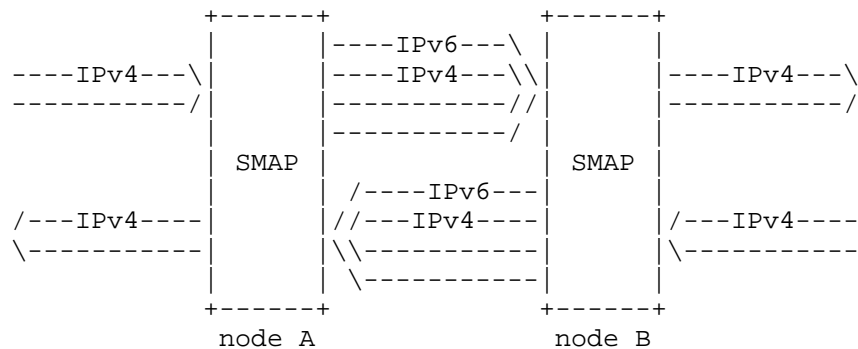


Figure 9

Several deployment scenarios of the SMAP function may be envisaged in the context of Port Range based solutions:

- o A SMAP function is embedded in a port-restricted device. Other SMAP-enabled nodes are deployed in the boundaries between IPv6-enabled realms and IPv4 ones. This scenario may be particularly deployed for intra-domain communications so as to interconnect heterogeneous realms (i.e., IPv6/IPv4) within the same AS.
- o A SMAP function is embedded in a port-restricted device. Other

SMAP-enabled nodes are deployed in the interconnection segment (with adjacent IPv4-only ones) of a given AS. This deployment scenario is more suitable for service providers targeting the deployment of IPv6 since it eases the migration to full IPv6. Core nodes are not required to activate anymore both IPv4 and IPv6 transfer capabilities.

Other considerations regarding the interconnection of SMAP-enabled domains should be elaborated. The following provides a non exhaustive list of interconnection schemes.

o The interconnection of two domains implementing the SMAP function may be deployed via IPv4 Internet (Figure 10): This means that IPv4 packets encapsulated in IPv6 one are transferred using IPv6 until reaching the first SMAP-node. Then these packets are de-encapsulated and are forwarded using IPv4 transfer capabilities. A remote SMAP-enabled node will receive those packets and proceeds to an IPv4-in-IPv6 encapsulation. These packets are then routed normally until reaching the port-restricted devices which de-encapsulates the packets.

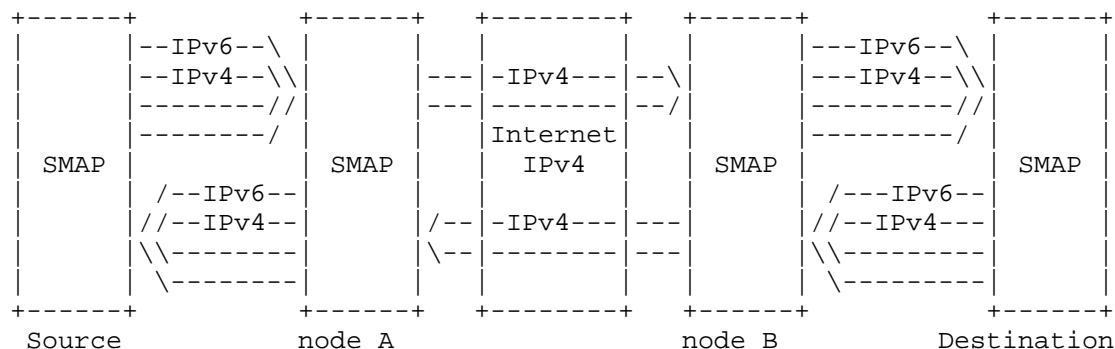


Figure 10: Interconnection Scenario 1

o A second scheme is to interconnect two realms implementing the SMAP function using IPv6 (Figure 11). An IPv6 prefix (i.e., Pref6) assigned by IANA is used for this service. If appropriate routing configuration have been enforced, then the IPv6 encapsulated packets will be routed until the final destination. In order to implement this model, IPv4-inferred IPv6 prefixes are required to be injected in the IPv6 inter-domain routing tables.

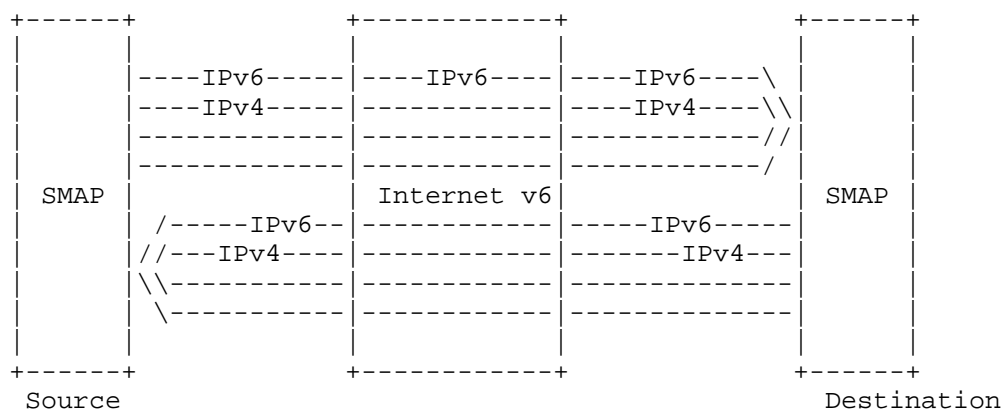


Figure 11: Interconnection Scenario 2

4.3. Towards IPv6-only Networks

The deployment of SMAP function allows for smooth migration of networks to IPv6-only scheme while maintaining the delivery of IPv4 connectivity services to customers. The delivery of IPv4 connectivity services over an IPv6-only network does not require any stateful function to be deployed on the core network. Owing to this A+P mode, both the IPv4 service continuity and migration to an IPv6-only deployment model are facilitated.

4.4. PRR: On Stateless and Binding Table Modes

SMAP section discusses two modes: the binding and the stateless modes. Dynamic port allocation is not a feature of the stateless mode but it is supported in the binding mode. In the binding mode, distinct external IPv4 addresses may be used but this is not recommended.

o Stateless Mode

Complete stateless mapping implies that the IPv4 address and the significant bits coding the port range are reflected inside the IPv6 prefix assigned to the port-restricted device. This can be achieved either by embedding the full IPv4 address and the significant bits in the IPv6 prefix or by applying an algorithmic approach. Two alternatives are offered when such a stateless mapping is to be enabled:

- either using the IPv6 prefix already used for native IPv6 traffic,

- or provide two prefixes to the port-restricted device: one for the native IPv6 traffic and one for the IPv4 traffic.

Note that:

- Providing two IPv6 prefixes has the advantages of allowing a /64 prefix for the port-restricted device along with another prefix (e.g., a /56 or /64) for native IPv6 traffic. This alternative spares the service provider to relate the native IPv6 traffic addressing plan to the IPv4 addressing plan. The drawback is the burden to allocate two prefixes to each port-restricted device and to route them. In addition, an address selection issue may be encountered.
- Providing one prefix for both needs (e.g., a /56 or a /64) spares the service provider to handle two types of IPv6 prefix for the port-restricted device and in routing tables. But the drawback is that it somewhat links strongly the IPv4 addressing plan to the allocated IPv6 prefixes.

As mentioned in Section 4.1 of [RFC6052], the suffix part may enclose the port.

o Binding Table Mode

Another alternative is to assign a "normal" IPv6 prefix to the port-restricted device and to use a binding table, which can be hosted by a service node, to correlate the (shared IPv4 address, Port Range) with an IPv6 address part of the assigned IPv6 prefix. For scalability reasons, this table should be instantiated within PRR-enabled nodes which are close to the port-restricted devices. The number of required entries if hosted at interconnection segment would be equal to the amount of subscribed users (one per port-restricted device).

4.5. General recommendations on SMAP

If Stateless A+P Mapping (SMAP) type of implementation is deployed over intermediate IPv6-ONLY-capable devices, it is recommended that default-routes are configured and IPv4 routing table is not "leaked" into IPv6 routing table in terms to have reachability for the packets going towards the internet.

One of stateless A+P variants is 4RD [I-D.despres-intarea-4rd]

5. Deployment Scenarios

5.1. A+P Deployment Models

5.1.1. A+P for Broadband Providers

Some large broadband providers will not have enough public IPv4 address space to provide every customer with a single IP. The natural solution is sharing a single IP address among many customers. Multiplexing customers is usually accomplished by allocating different port numbers to different customers somewhere within the network of the provider.

It is expected that, when the provider wishes to enable A+P for a customer or a range of customers, the CPE can be upgraded or replaced to support A+P encaps/decaps functionality. Ideally the CPE also provides NATing functionality. Further, it is expected that at least another component in the ISP network provides the corresponding A+P functionality, and hence is able to establish an A+P subsystem with the CPE. This device is referred to as A+P router or port-range router (PRR), and could be located close to PE routers. The core of the network MUST support the tunneling protocol (which SHOULD be IPv6, as per Constraint 7) but MAY be another tunneling technology when necessary. In addition, we do not wish to restrict any initiative of customers who might want to run an A+P-capable network on or behind their CPE. To satisfy both Constraints 1 and 2 unmodified legacy hosts should keep working seamlessly, while upgraded/new end-systems should be given the opportunity to exploit enhanced features.

5.1.2. A+P for Mobile Providers

In the case of mobile service provider the situation is slightly different. The A+P border is assumed to be the gateway (e.g., GGSN/PDN GW of 3GPP, or ASN GW of WiMAX). The need to extend the address is not within the provider network, but on the edge between the mobile phone devices and the gateway. While desirable, IPv6 connectivity may or may not be provided.

For mobile providers we use the following terms and assumptions:

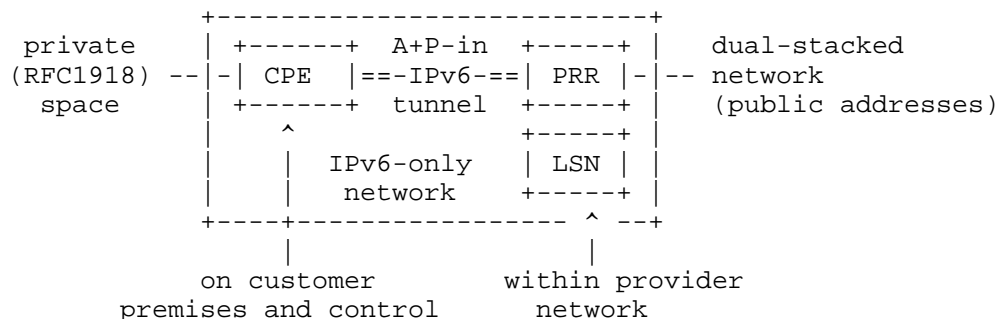
1. Provider Network (PN)
2. Gateway (GW)
3. Mobile Phone device (phone)

4. Devices behind phone, e.g., laptop computer connecting via phone to Internet.

We expect that the gateway has a pool of IPv4 addresses and is always in the data-path of the packets. Transport between the gateway and phone devices is assumed to be an end-to-end layer-2 tunnel. We assume that phone as well as gateway can be upgraded to support A+P. However, some applications running on the phone or devices behind the phone (such as laptop computers connecting via the phone), are not expected to be upgraded. Again, while we do not expect that devices behind the phone will be A+P aware/upgraded we also do not want to hinder their evolution. In this sense the mobile phone would be comparable to the CPE in the broadband provider case; the gateway to the PRR/LSN box in the network of the broadband provider.

5.1.3. A+P from the Provider Network Perspective

ISPs suffering from IPv4 address space exhaustion are interested in achieving a high address space compression ratio. In this respect, an A+P subsystem allows much more flexibility than traditional NATs: the NAT can be placed at the customer, and/or in the provider network. In addition hosts or applications can request ports and thus have untranslated end-to-end connectivity.



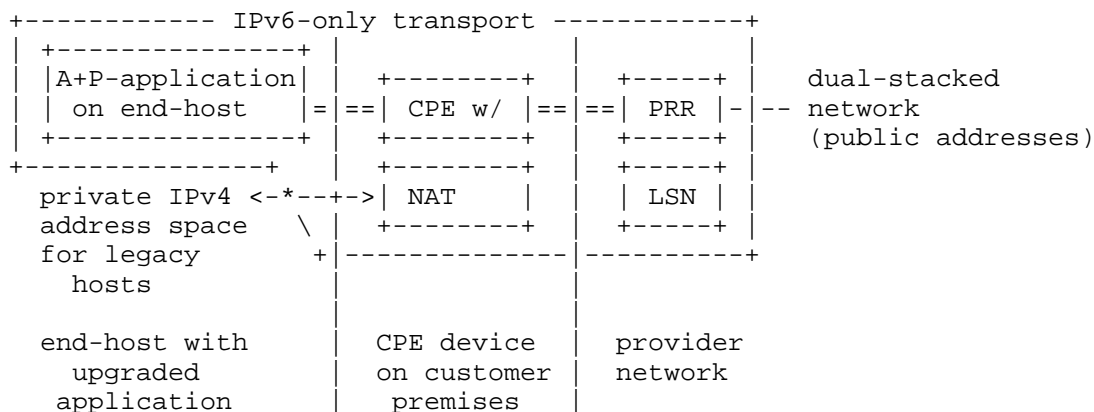
A simple A+P subsystem example

Figure 12

Consider the deployment scenario in Figure 12, where an A+P subsystem is formed by the CPE and a PRR within the ISP core network, preferably close to the customer edge, and represents the border from where on packets are forwarded based on address and port. The provider MAY deploy a LSN co-located with the PRR to handle packets that have not been translated by the CPE. In such a configuration,

the ISP allows the customer to freely decide whether the translation is done at the CPE or at the LSN. In order to establish the A+P subsystem, the CPE will be configured automatically (e.g. via a signaling protocol, that conforms to the requirements stated above).

Note that the CPE in the example above is only provisioned with an IPv6 address on the external interface.



An extended A+P subsystem with end-host running A+P-aware applications

Figure 13

Figure 13 shows an example of how an upgraded application running on a legacy end-host can connect to another host in the public realm. The legacy host is provisioned with a private IPv4 address allocated by the CPE. Any packet sent from the legacy host will be NATed either at the CPE (if configured to do so), or at the LSN (if available).

An A+P-aware application running on the end-host MAY use the signaling described in Section 3.3.1 to connect to the A+P-subsystem. In this case, the application will be delegated some space in the A+P address realm, and will be able to contact the public realm (i.e., the public Internet) without the need for translation.

Note that part of A+P signaling is that the NATs are optional. However, if neither the CPE nor the PRR provides NATing functionality, then it will not be possible to connect legacy end-hosts.

To enable packet forwarding with A+P, the ISP MUST install at its A+P border a PRR which encapsulates/decapsulates packets. However, to achieve a higher address space compression ratio and/or to support CPEs without NATing functionality, the ISP MAY decide to provide an LSN as well. If no LSN is installed in some part of the ISP's topology, all CPE in that part of the topology MUST support NAT functionality. For reasons of scalability, it is assumed that the PRR is located within the access-portion of the network. The CPE would be configured automatically (e.g. via an extended DHCP or NAT-PMP, which has the signaling requirements stated above) with the address of the PRR, and if a LSN is being provided or not. Figure 12 illustrates a possible deployment scenario.

5.2. Dynamic Allocation of Port Ranges

Allocating a fixed number of ports to all CPEs may lead to exhaustion of ports for high usage customers. This is a perfect recipe for upsetting more demanding customers. On the other hand, allocating to all customers ports sufficient to match the needs of peak users will not be very efficient. A mechanism for dynamic allocation of port ranges allows the ISP to achieve two goals; a more efficient compression ratio of number of customers on one IPv4 address and, on the other hand, not limiting the more demanding customers' communication.

Additional allocation of ports, or port ranges may be made after an initial static allocation of ports.

The mechanism would prefer allocations of port ranges from the same IP address as the initial allocation. If it is not possible to allocate an additional port range from the same IP, then mechanism can allocate a port range from another IP within the same subnet. With every additional port range allocation, the PRR updates its routing table. The mechanism for allocating additional port ranges may be part of normal signaling that is used to authenticate CPE to ISP.

The ISP controls the dynamic allocation of port ranges by the PRR by setting the initial allocation size and maximum number of allocations per CPE, or the maximum allocations per subscription, depending on subscription level. There is a general observation that the more demanding customer uses around 1024 ports when heavily communicating. So, for example, a first suggestion might be 128 ports initially and then dynamic allocations of ranges of 128 ports up to 511 more allocations maximum. A configured maximum number of allocations could be used to prevent one customer acting in destructive manner should they become infected. The maximum number of allocations might also be more finely grained, with parameters of how many allocations

a user may request per some time frame. If this is used, evasive applications may need to be limited in their bad behavior, for example one additional allocation per minute would considerably slow a port request storm.

There is likely no minimum request size. This is because A+P-aware applications running on end-hosts MAY request a single port (or a few ports) for the CPE to be contacted on (e.g., VoIP clients register a public IP and a single delegated port from the CPE, and accept incoming calls on that port). The implementation on the CPE or PRR will dictate how to handle such requests for smaller blocks: For example, half of available blocks might be used for "block-allocations", 1/6 for single port requests, and the rest for NATing.

Another possible mechanism to allocate additional ports is UPnP/NAT-PMP (as defined in Section 3.3.1), if applications behind CPE support it. In case of the LSN implementation (DS-Lite), as described in the A+P overall architecture section, signaling packets are simply forwarded by the CPE to the LSN and back to the host running the application which requested the ports, and PRR allocates requested port to appropriate CPE. The same behavior may be chosen with AFTR, if requested ports are outside of static initial port allocation. If a full A+P implementation is selected, than UPnPv2/NAT-PMP packets are accepted by the CPE, processed, and the requested port number is communicated through normal signaling mechanism between CPE and PRR tunnel endpoints (PCP).

5.3. Example of A+P-forwarded Packets

This section provides a detailed example of A+P setup, configuration, and packet flow from an end-host connected to A+P Service Provider to any host in the IPv4 Internet, and how the return packets flow back. The following example discusses an A+P-unaware end-host, where the NATing is done at the CPE. Figure 14 illustrates how the CPE receives an IPv4 packet from the end-user device. We first describe the case where the CPE has been configured to provide the NAT functionality (e.g., by the customer through interaction with a website, or automatic signaling). In the following, we call a packet which is translated at the CPE an A+P-forwarded packet, an analogy with the port-forwarding function employed in today's CPEs. Upon receiving a packet from the internal interface, the CPE translates, encapsulates and forwards it to the PRR. The NAT on the CPE is assumed to have a default route to the public realm through its tunnel interface.

When the PRR receives the A+P-forwarded packet, it de-capsulates the inner IPv4 packet and checks the source address. If the source address does match the range assigned to A+P enabled CPEs, then the

PRR simply forwards the decapsulated packet onward. This is always the case for A+P-forwarded packets. Otherwise, the PRR assumes that the packet is not A+P-forwarded, and passes it to the LSN function, which in-turn translates and forward the packet based on the destination address. Figure 14 shows the packet flow for an outgoing A+P-forwarded packet.

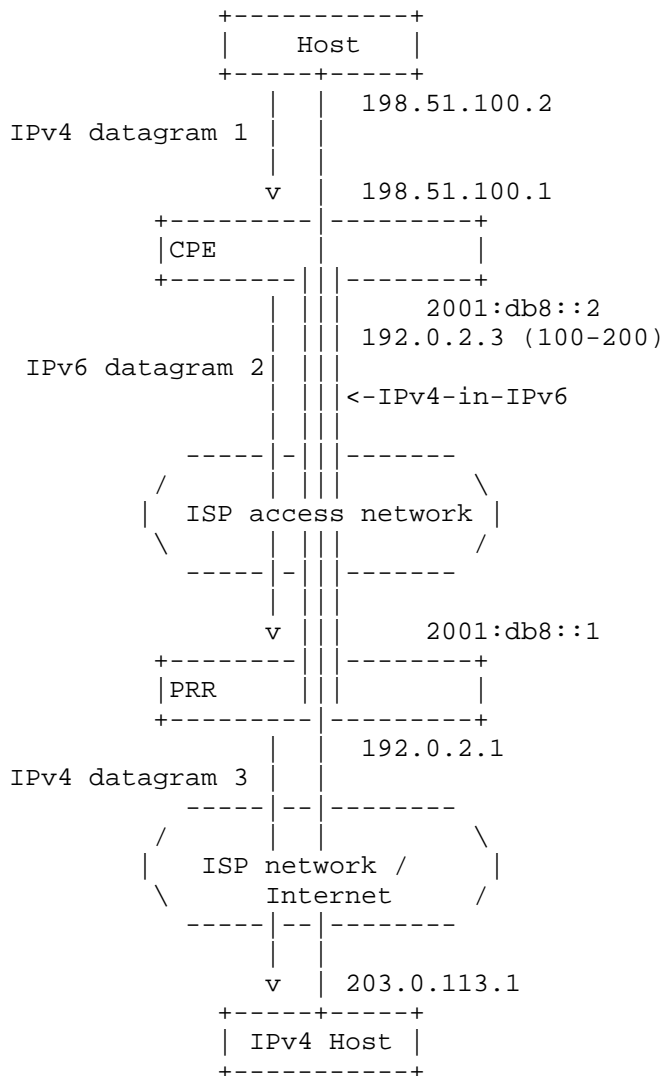


Figure 14: Forwarding of Outgoing A+P-forwarded Packets

Datagram	Header field	Contents
IPv4 datagram 1	IPv4 Dst	203.0.113.1
	IPv4 Src	198.51.100.2
	TCP Dst	80
	TCP Src	8000
IPv6 Datagram 2	IPv6 Dst	2001:db8::1
	IPv6 Src	2001:db8::2
	IPv4 Dst	203.0.113.1
	IPv4 Src	192.0.2.3
	TCP Dst	80
	TCP Src	100
IPv4 datagram 3	IPv4 Dst	203.0.113.1
	IPv4 Src	192.0.2.3
	TCP Dst	80
	TCP Src	100

Datagram header contents

An incoming packet undergoes the reverse process. When the PRR receives an IPv4 packet on an external interface, it first checks whether the destination address falls within the A+P CPE delegated range or not. If the address space was delegated, then PRR encapsulates the incoming packet and forwards it through the appropriate tunnel for that IP/port range. If the address space was not-delegated the packet would be handed to the LSN to check if a mapping is available.

Figure 15 shows how an incoming packet is forwarded, under the assumption that the port number matches the port range which was delegated to the CPE.

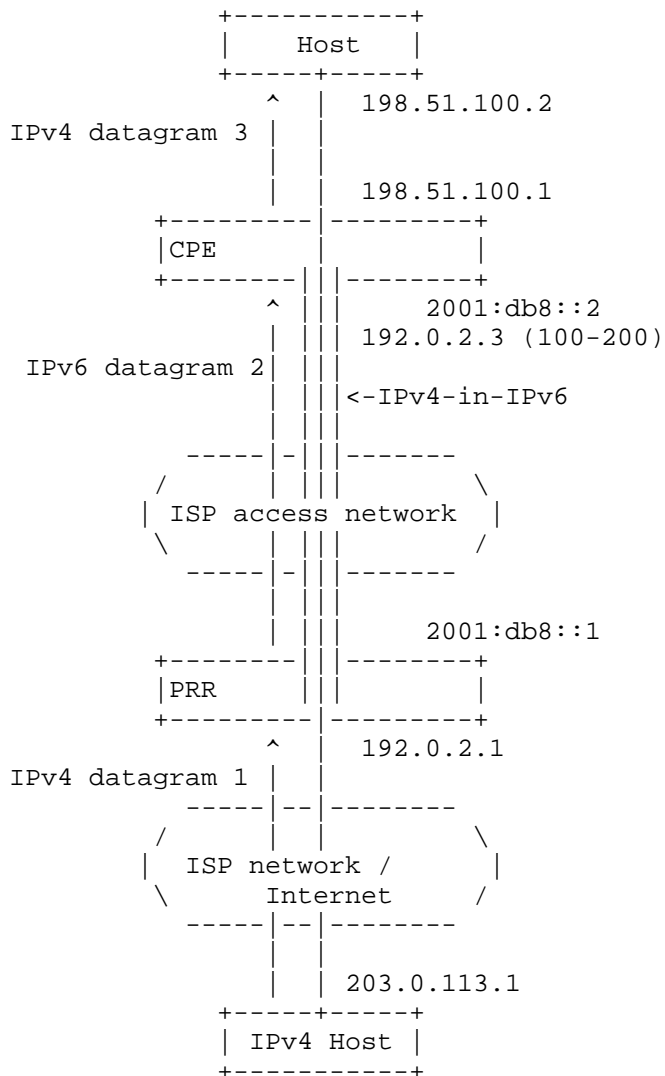


Figure 15: Forwarding of Incoming A+P-forwarded Packets

Datagram	Header field	Contents
IPv4 datagram 1	IPv4 Dst	198.51.100.3
	IPv4 Src	203.0.113.1
	TCP Dst	100
	TCP Src	80
IPv6 Datagram 2	IPv6 Dst	2001:db8::2
	IPv6 Src	2001:db8::1
	IPv4 Dst	198.51.100.3
	IP Src	203.0.113.1
	TCP Dst	100
	TCP Src	80
IPv4 datagram 3	IPv4 Dst	198.51.100.2
	IPv4 Src	203.0.113.1
	TCP Dst	8000
	TCP Src	80

Datagram header contents

Note that datagram 1 travels untranslated up to the CPE, thus the customer has the same control over the translation as it has today where s/he has an home gateway with customizable port-forwarding.

5.3.1. Forwarding of Standard Packets

Packets for which the CPE does not have a corresponding port forwarding rule are tunneled to the PRR which provides the LSN function. We underline that the LSN MUST NOT use the delegated space for NATting. See [I-D.ietf-softwire-dual-stack-lite] for network diagrams which illustrate the packet flow in this case.

5.3.2. Handling ICMP

ICMP is problematic for all NATs, because it lacks port numbers. A+P routing exacerbates the problem.

Most ICMP messages fall into one of two categories: error reports, or ECHO/ECHO reply (commonly known as "ping"). For error reports, the offending packet header is embedded within the ICMP packet; NAT devices can then rewrite that portion and route the packet to the actual destination host. This functionality will remain the same with A+P; however, the PRR will need to examine the embedded header to extract the port number, while the A+P gateway will do the necessary rewriting.

ECHO and ECHO reply are more problematic. For ECHO, the A+P gateway device must rewrite the "Identifier" and perhaps "Sequence Number" fields in the ICMP request, treating them as if they were port numbers. This way, the PRR can build the correct A+P address for the returning ECHO replies, so they can be correctly routed back to the appropriate host in the same way as TCP/UDP packets. Pings originated from the Public Realm (Internet) towards an A+P device are not supported.

5.3.3. Fragmentation

In order to deliver a fragmented IP packet to its final destination (among those having the same IP address), the PRR should activate a dedicated procedure similar to the one used by [I-D.ietf-behave-v6v4-xlate-stateful], section 3.5 in a sense that it should reassemble the fragments in order to look at the destination port number.

Note that it is recommended to use a PMTUD path discovery mechanism (e.g., [RFC1191]).

Security issues related to fragmentation are out of scope of this document. For more details, refer to [RFC1858].

5.3.4. Limitations of the A+P approach

One limitation that A+P shares with any other IP address-sharing mechanism is the availability of well-known ports. In fact, services run by customers that share the same IP address will be distinguished by the port number. As a consequence, it will be impossible for two customers who share the same IP address to run services on the same port (e.g., port 80). Unfortunately, working around this limitation usually implies application-specific hacks (e.g., HTTP and HTTPS redirection), discussion of which is out of the scope of this document. Of course, a provider might charge more for giving a customer the well-known port range, 0..1024, thus allowing the customer to provide externally available services. Many applications require the availability of well known ports. However, those applications are not expected to work in A+P environment unless they can adapt to work with different ports. However, such application do not work behind today's NATs either.

Another problem which is common to all NATs is coexistence with IPsec. In fact, a NAT which also translates port numbers prevents AH and ESP from functioning properly, both in tunnel and in transport mode. In this respect, we stress that, since an A+P subsystem exhibits the same external behavior as a NAT, well-known workarounds (such as [RFC3715]) can be employed.

A+P, as all other port sharing solutions also suffers from the issues documented in [I-D.ietf-intarea-shared-addressing-issues], but that's something we'll have to live with.

For the host-based A+P, issues related to applications conflicts trying to bind to an out-of-range port are to be further assessed. Note that extensions to the host-based model have been proposed in the past (e.g., Port Enhanced ARP extension documented in <http://software.merit.edu/pe-arp/>).

5.3.5. Port allocation strategy agnostic

Issues, raised by [I-D.thaler-port-restricted-ip-issues] have been analyzed in [I-D.dec-stateless-4v6]. As seen in that document, most of the issues apply to host based port sharing solutions. A+P is not intended to be host based port sharing solution.

Conclusion of [I-D.dec-stateless-4v6] document is, that the set of issues specifically attributed to A+P either do not apply to CPE-based flavours, or can be mitigated. A+P solution represents a reasonable trade off compared to alternatives in areas such as binding logging (for data storage purposes), ease as of deployment and operations, all of which are actually facilitated by such a solution.

6. IANA Considerations

This document makes no request of IANA.

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

7. Security Considerations

With CGN/LSNs, tracing hackers, spammers and other criminals will be difficult, requiring logging, recording, and storing of all connection based mapping information. The need for storage implies a tradeoff. On one hand, the LSNs can manage addresses and ports as dynamically as possible in order to maximize aggregation. On the other hand, the more quickly the mapping between private and public space changes, the more information needs to be recorded. This would not only cause concern for law enforcement services, but also for privacy advocates.

A+P offers a better set of tradeoffs. All that needs to be logged is the allocation of a range of port numbers to a customer. By design,

this will be done rarely, improving scalability. If the NAT functionality is moved further up the tree, the logging requirement will be as well, increasing the load on one node, but giving it more resources to allocate to a busy customer, perhaps decreasing the frequency of allocation requests.

The other extreme is A+P NAT on the customer premises. Such a node would be no different than today's NAT boxes, which do no such logging. We thus conclude that A+P is no worse than today's situation, while being considerably better than CGNs.

8. Authors

This document has 9 primary authors, which is not allowed in the header of Internet-Drafts. This is the list of actual authors of this document.

Gabor Bajko
Nokia
Email: gabor(dot)bajko(at)nokia(dot)com

Mohamed Boucadair
France Telecom
3, Av Francois Chateaux
Rennes 35000
France
Email: mohamed.boucadair@orange-ftgroup.com

Steven M. Bellovin
Columbia University
1214 Amsterdam Avenue
MC 0401
New York, NY 10027
US
Phone: +1 212 939 7149
Email: bellovin@acm.org

Randy Bush
Internet Initiative Japan
5147 Crystal Springs
Bainbridge Island, Washington 98110
US
Phone: +1 206 780 0431 x1
Email: randy@psg.com

Luca Cittadini
Universita' Roma Tre

via della Vasca Navale, 79
Rome, 00146
Italy
Phone: +39 06 5733 3215
Email: luca.cittadini@gmail.com

Olaf Maennel
Loughborough University
Department of Computer Science - N.2.03
Loughborough
United Kingdom
Phone: +44 115 714 0042
Email: o@maennel.net

Reinaldo Penno
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
Email: rpenno@juniper.net

Teemu Savolainen
Nokia
Hermiankatu 12 D
TAMPERE, FI-33720
Finland
Email: teemu.savolainen@nokia.com

Jan Zorz
Go6 Institute Slovenia
Frankovo naselje 165
Skofja Loka, 4220
Slovenia
Email: jan@go6.si

9. Acknowledgments

The authors wish to especially thank Remi Despres, and Pierre Levis for their help on the development of the A+P approach. We also thank David Ward for review, constructive criticism, and interminable questions, and Dave Thaler for useful criticism on "stackable" A+P gateways. We would also like to thank the following persons for their feedback on earlier versions of this work: Rob Austein, Gert Doering, Dino Farinacci, Russ Housley, Ruediger Volk, Tina Tsou and Pasi Sarolahti.

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.

10.2. Informative References

- [BCP38] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, May 2000.
- [I-D.bajko-pripaddrassign]
Bajko, G., Savolainen, T., Boucadair, M., and P. Levis, "Port Restricted IP Address Assignment", draft-bajko-pripaddrassign-03 (work in progress), September 2010.
- [I-D.boucadair-behave-bittorrent-portrange]
Boucadair, M., Grimault, J., Levis, P., and A. Villefranque, "Behaviour of BitTorrent service in an IP Shared Address Environment", draft-boucadair-behave-bittorrent-portrange-02 (work in progress), January 2009.
- [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.boucadair-pppext-portrange-option]
Boucadair, M., Levis, P., and T. Savolainen, "Port Range Configuration Options for PPP IPCP", draft-boucadair-pppext-portrange-option-04 (work in progress), September 2010.
- [I-D.dec-stateless-4v6]
Dec, W., "Stateless 4Via6 Address Sharing", draft-dec-stateless-4v6-01 (work in progress), March 2011.
- [I-D.deng-aplusp-experiment-results]
Deng, X., Boucadair, M., and F. Telecom, "Implementing A+P in the provider's IPv6-only network", draft-deng-aplusp-experiment-results-00 (work in progress), March 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-behave-v6v4-xlate-stateful]
Bagmulo, M., Matthews, P., and I. Beijnum, "Stateful
NAT64: Network Address and Protocol Translation from IPv6
Clients to IPv4 Servers",
draft-ietf-behave-v6v4-xlate-stateful-12 (work in
progress), July 2010.
- [I-D.ietf-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-pcp-base]
Wing, D., Cheshire, S., Boucadair, M., and R. Penno, "Port
Control Protocol (PCP)", draft-ietf-pcp-base-08 (work in
progress), April 2011.
- [I-D.ietf-softwire-dual-stack-lite]
Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-
Stack Lite Broadband Deployments Following IPv4
Exhaustion", draft-ietf-softwire-dual-stack-lite-07 (work
in progress), March 2011.
- [I-D.thaler-port-restricted-ip-issues]
Thaler, D., "Issues With Port-Restricted IP Addresses",
draft-thaler-port-restricted-ip-issues-00 (work in
progress), February 2010.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
November 1990.
- [RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security
Considerations for IP Fragment Filtering", RFC 1858,
October 1995.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and
E. Lear, "Address Allocation for Private Internets",
BCP 5, RFC 1918, February 1996.
- [RFC3715] Aboba, B. and W. Dixon, "IPsec-Network Address Translation

(NAT) Compatibility Requirements", RFC 3715, March 2004.

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

Author's Address

Randy Bush (editor)
Internet Initiative Japan
5147 Crystal Springs
Bainbridge Island, Washington 98110
US

Phone: +1 206 780 0431 x1
Email: randy@psg.com

