Network Working Group
Internet-Draft
Intended status: BCP
Expires: July 3, 2011

R. Stewart
Huawei
M. Tuexen
I. Ruengeler
Muenster Univ. of Applied Sciences
December 30, 2010

Stream Control Transmission Protocol (SCTP) Network Address Translation

draft-ietf-behave-sctpnat-04.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 July 3, 2011.

Copyright Notice

Table of Contents

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 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.

- 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.

- 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.

- 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 a figure.

- 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.

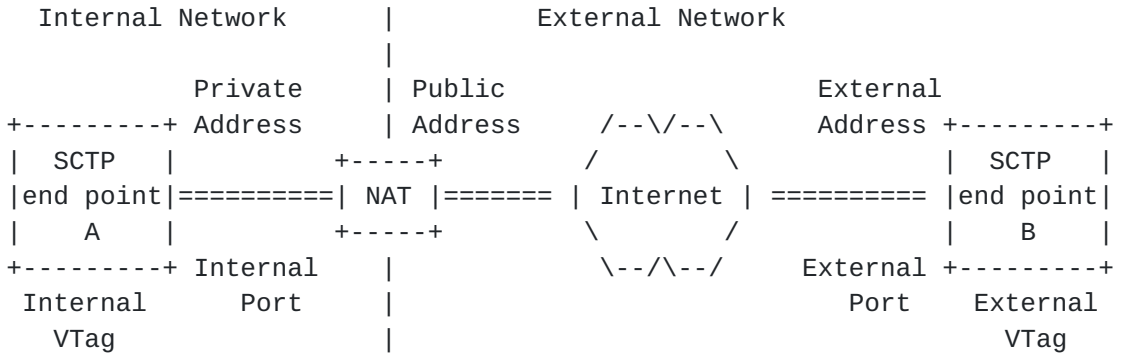
- Internal-VTag (Int-VTag) - The Verification Tag that the internal host has chosen for its communication. The VTag is a unique 32 bit tag that must accompany any incoming SCTP packet for this association to the Private-Address.

- 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.

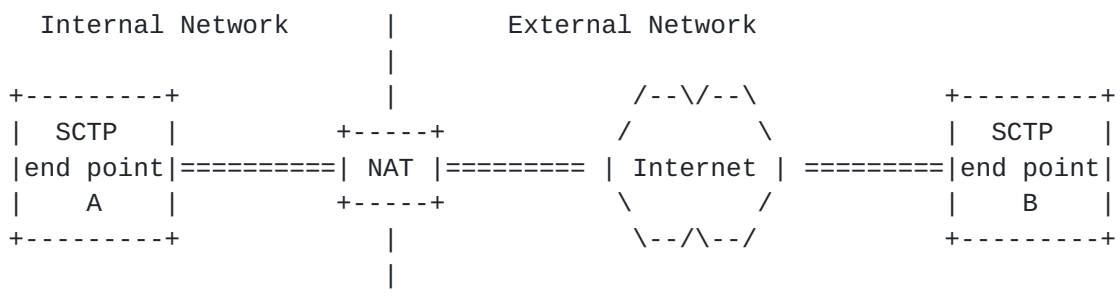


4. SCTP NAT Traversal Scenarios

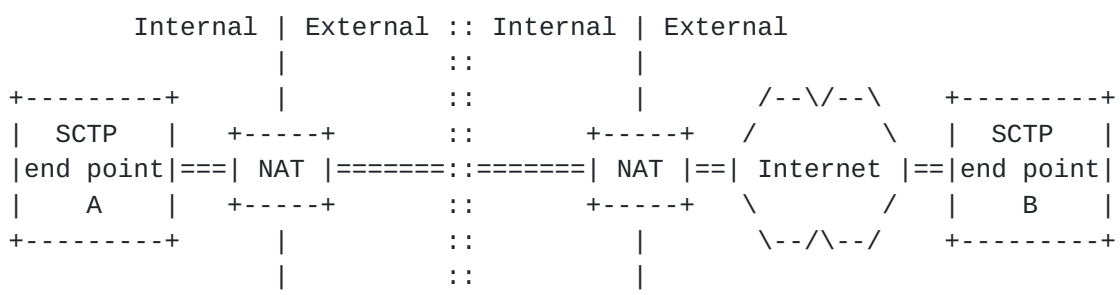
This section defines the notion of single and multi-point NAT traversal.

4.1. Single Point Traversal

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



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

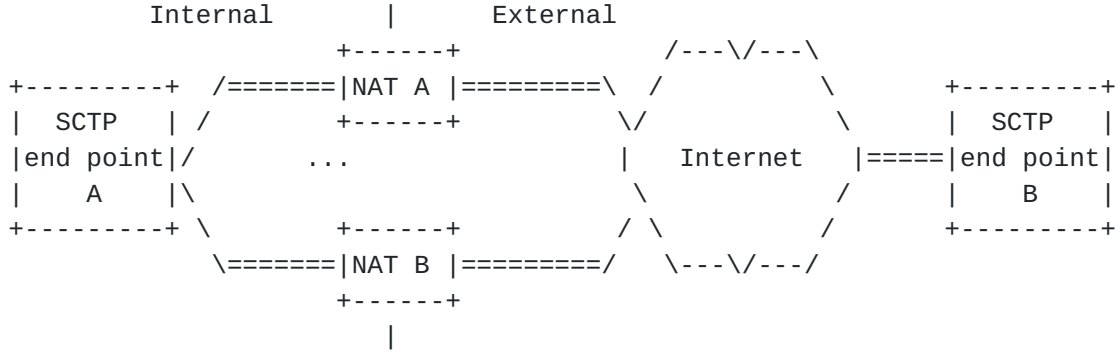


In this single point traversal scenario, we must acknowledge that while one of the main benefits of SCTP multi-homing is redundant paths, the NAT function represents a single point of failure in the path of the SCTP multi-home association. However, the rest of the path may still benefit from path diversity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



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 NAPT for SCTP

Using classical NAPT 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 efficently, 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 travesal 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 communcation. Such mechanisms are considered by the authors to be undeployable 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.

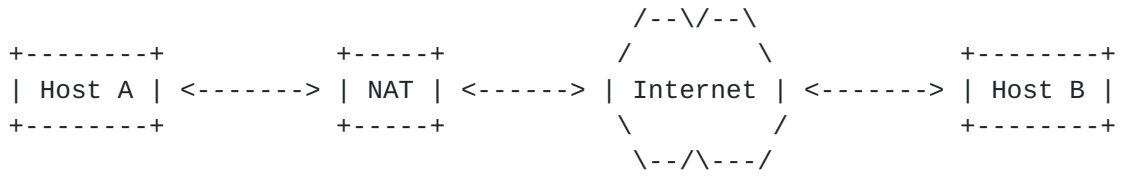
The modification of SCTP packets sent to the public Internet is easy. The source address of the packet has to be replaced with the Public-Address. It may also be necessary to establish some state in the NAT box to handle incoming packets, which is discussed later.

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 External-Address. 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 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.



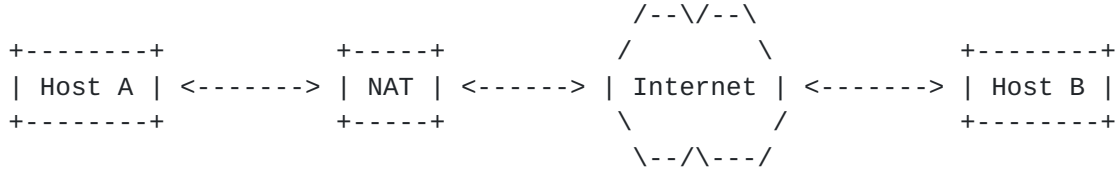

```
              INIT[Initiate-Tag]
Priv-Addr:Int-Port ------> Ext-Addr:Ext-Port
                  Ext-VTag=0

                   Create(Initiate-Tag,Internal-Port,Private-Address,
                          0,External-Port,External-Address)
                   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 [please see xxx-tsvwg-document-xx for the format].

It is also possible that a connection to External-Address and External-Port exists without anInternal-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 [please see xxx-tsvwg-document-xx for the format].

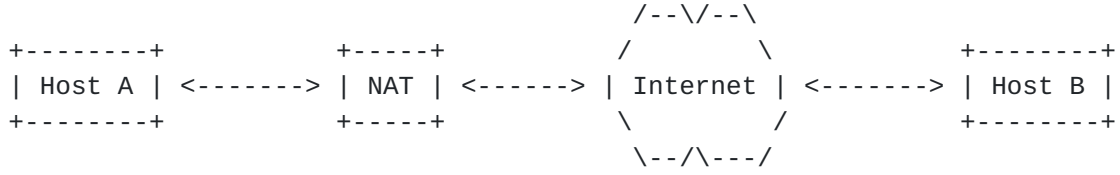
The processing of outgoing SCTP packets containing no INIT-chunk is

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.

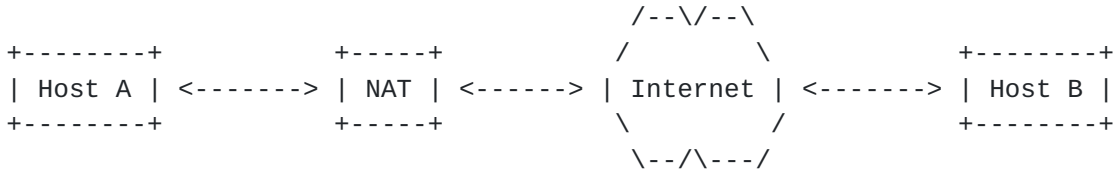



```
                                      INIT-ACK[Initiate-Tag]
                           Pub-Addr:Int-Port <------ Ext-Addr:Ext-Port
                                                     Int-VTag

           Lookup(Internal-VTag,Internal-Port,*,
                  0,External-Port,External-Address)
           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
```

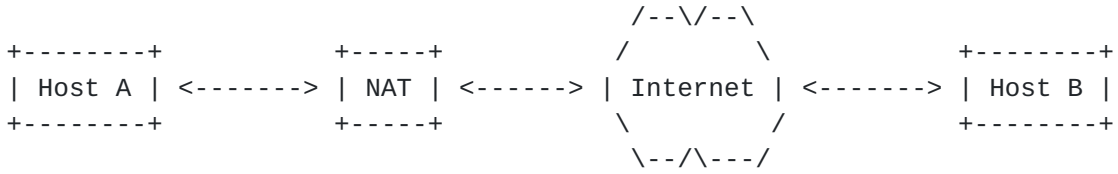
In the case Lookup fails, the SCTP packet is dropped. The Update

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.



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



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 proceedures are not defined in this document, but instead are defined in [xxxxx]. The NAT implementor should refer to [xxxx] for detailed descriptions of packet format 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. Simplification for small NATs

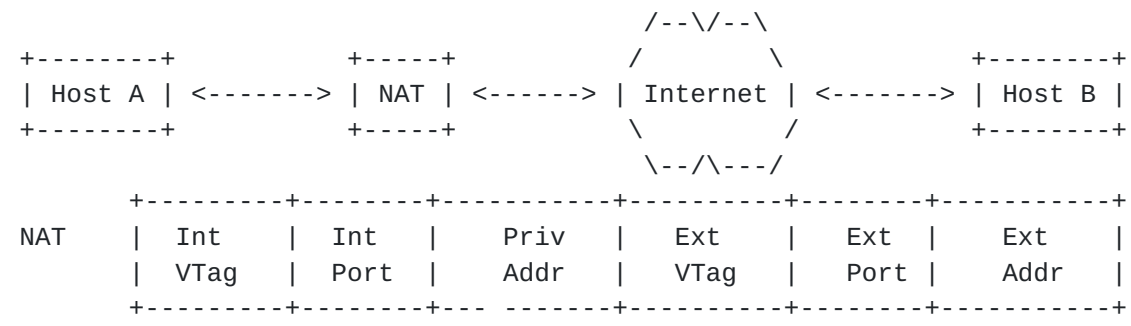
Small NAT boxes, i.e. NAT boxes which only have to support a small number of concurrent SCTP associations, MAY not take the external address into account when processing packets. Therefore the External-Address could also be removed from the NAT table.

This simplification may make implementing a NAT box easier, however, the collision probability is higher than using a mapping which takes the external address into account.

10. Various examples of NAT traversals

10.1. 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.



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

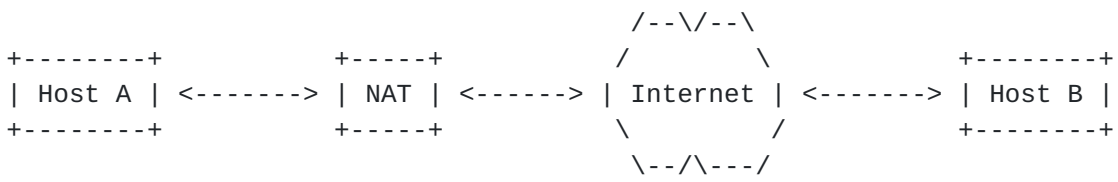
A NAT entry is created, the source address is substituted and the packet is sent on:

NAT creates entry:

NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	1234	1	10.0.0.1	0	2	100.0.0.1

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

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



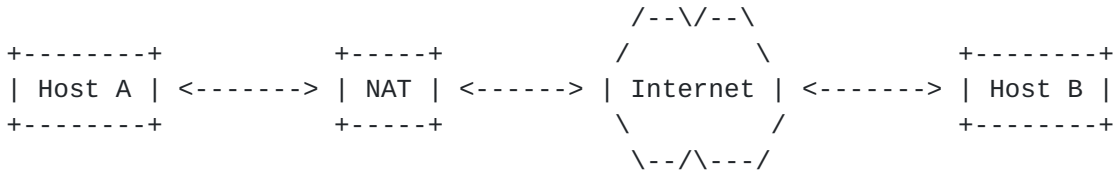
INIT-ACK[Initiate-Tag = 5678]
101.0.0.1:1 <---------- 100.0.0.1:2
Int-VTag = 1234

NAT updates entry:

NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	1234	1	10.0.0.1	5678	2	100.0.0.1

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

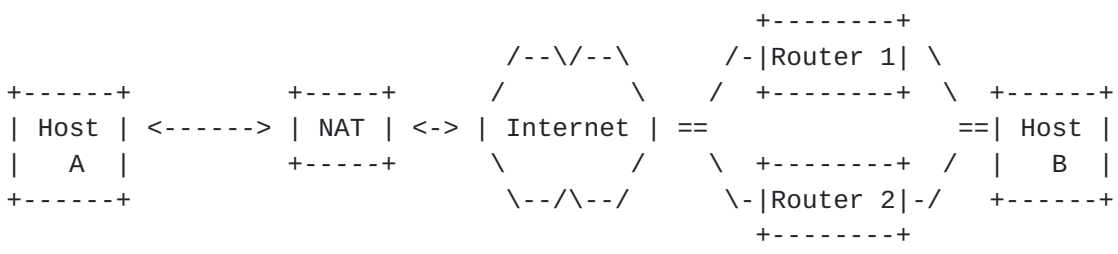
COOKIE-ACK
101.0.0.1:1 <---------- 100.0.0.1:2
Int-VTag = 1234

COOKIE-ACK
10.0.0.1:1 <------ 100.0.0.1:2

Int-VTag = 1234

10.2. 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	Ext Addr

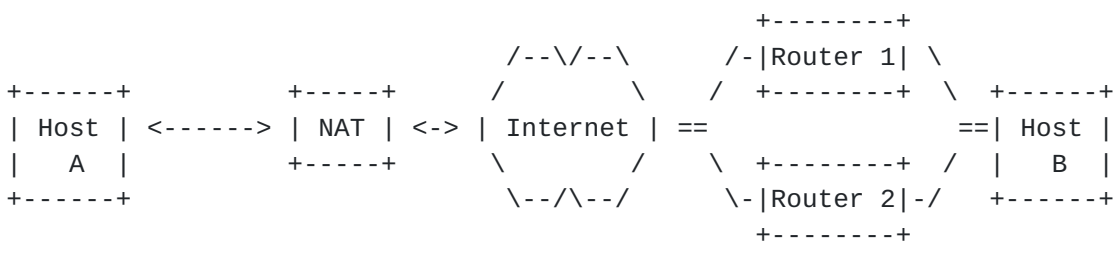
INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 100.0.0.1:2
Ext-VTag = 0

NAT creates entry:

NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	1234	1	10.0.0.1	0	2	100.0.0.1

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.



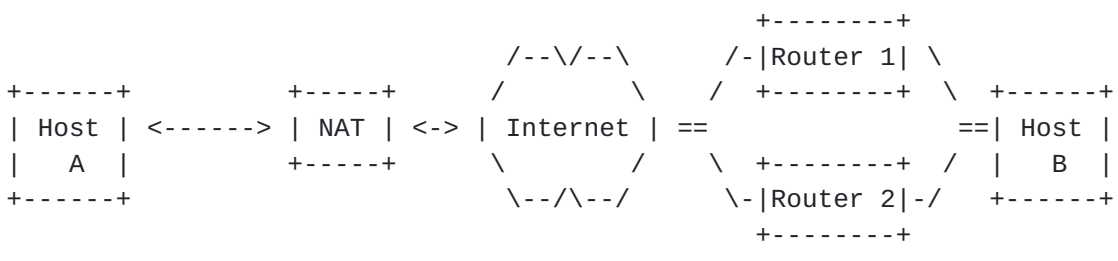
INIT-ACK[Initiate-tag = 5678, IP-Addr = 100.1.0.1]
101.0.0.1:1 <------------------------------ 100.0.0.1:2
Int-VTag = 1234

NAT updates first entry and creates entry for second address:

NAT	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	1234	1	10.0.0.1	5678	2	100.0.0.1
	1234	1	10.0.0.1	5678	2	100.1.0.1

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

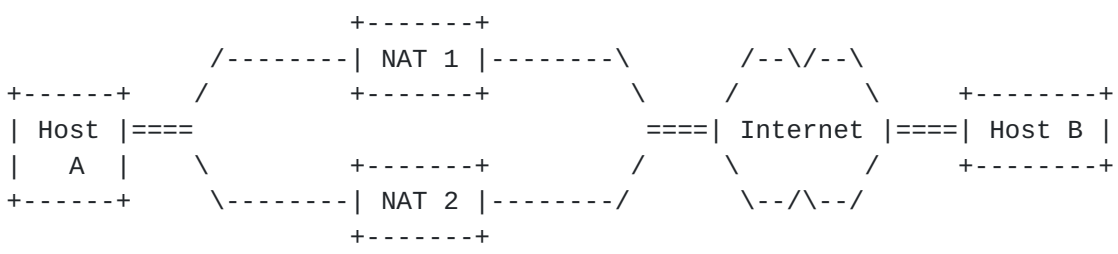
COOKIE-ECHO
101.0.0.1:1 ----------> 100.0.0.1:2
Ext-VTag = 5678

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

10.3. 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	Ext Addr

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

NAT 1 creates entry:

```
       +---------+--------+-----------+----------+--------+-----------+
NAT 1  |  Int    |  Int   |    Priv   |   Ext    |   Ext  |    Ext    |
       |  VTag   |  Port  |    Addr   |   VTag   |   Port |    Addr   |
       +---------+--------+-----------+----------+--------+-----------+
       |  1234   |    1   |  10.0.0.1 |     0    |    2   | 100.0.0.1 |
       +---------+--------+-----------+----------+--------+-----------+
```

```
                              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.

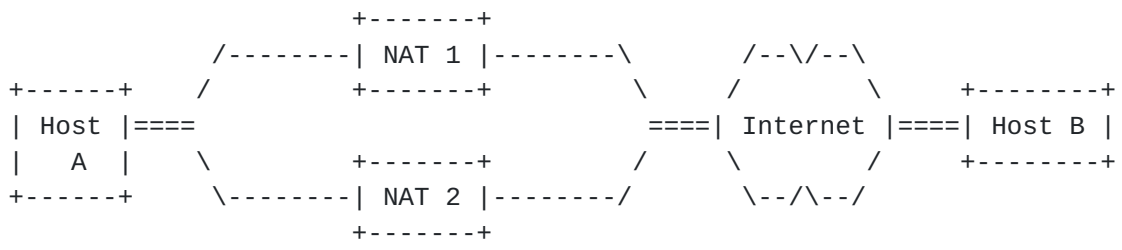
```
                      +-------+
             /--------| NAT 1 |--------\       /--\/--\
+------+    /         +-------+         \     /        \     +--------+
| Host |====                             ====| Internet |====| Host B |
|   A  |    \         +-------+         /     \        /     +--------+
+------+     \--------| NAT 2 |--------/       \--/\--/
                      +-------+

                    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 updates first entry and creates complete entry for second address:

```
       +---------+--------+-----------+----------+--------+-----------+
NAT 1  |  Int    |  Int   |    Priv   |   Ext    |   Ext  |    Ext    |
       |  VTag   |  Port  |    Addr   |   VTag   |   Port |    Addr   |
       +---------+--------+-----------+----------+--------+-----------+
       |  1234   |    1   |  10.0.0.1 |    5678  |    2   | 100.0.0.1 |
       |  1234   |    1   |  10.0.0.1 |    5678  |    2   | 100.1.0.1 |
       +---------+--------+-----------+----------+--------+-----------+
```

```
  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.



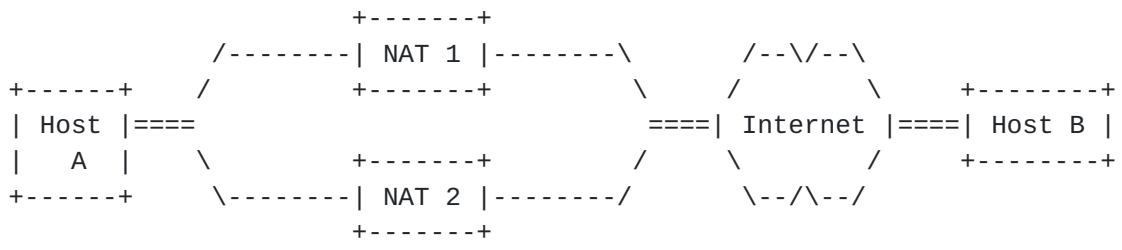
COOKIE-ECHO
10.0.0.1:1 --------> 100.0.0.1:2
Ext-VTag = 5678

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. The address parameter contains an undefined address (0) to indicate that the source address should be added. The lookup address pararmeter 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:

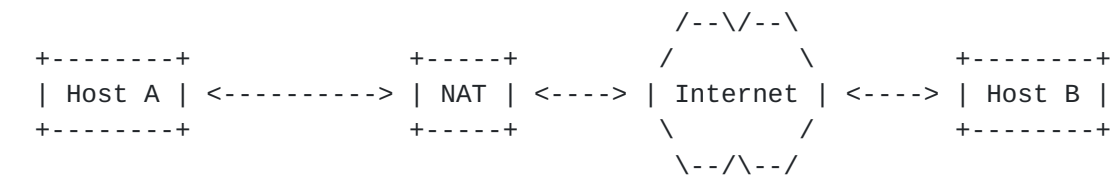
```
       +---------+--------+-----------+----------+--------+-----------+
NAT 2  |  Int    |  Int   |    Priv   |   Ext    |   Ext  |    Ext    |
       |  VTag   |  Port  |    Addr   |   VTag   |   Port |    Addr   |
       +---------+--------+-----------+----------+--------+-----------+
       |  1234   |    1   |  10.1.0.1 |    5678  |    2   | 100.1.0.1 |
       +---------+--------+-----------+----------+--------+-----------+


                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
```

10.4. NAT loses its state

Assocation 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.

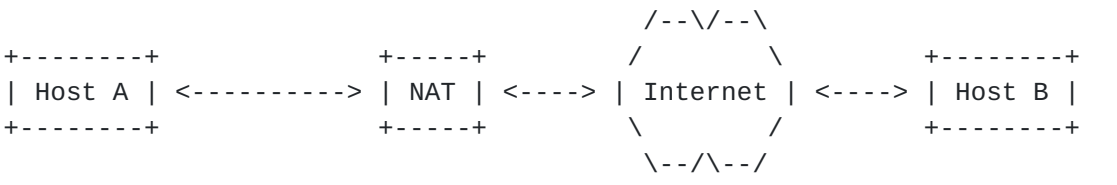
```
                                      /--\/--\
+--------+             +-----+       /        \        +--------+
| Host A | <---------> | NAT | <---> | Internet | <---> | Host B |
+--------+             +-----+       \        /        +--------+
                                      \--/\--/

       +---------+--------+-----------+----------+--------+-----------+
NAT A  |  Int    |  Int   |    Priv   |   Ext    |   Ext  |    Ext    |
       |  VTag   |  Port  |    Addr   |   VTag   |   Port |    Addr   |
       +---------+--------+-----------+----------+--------+-----------+
       |  1234   |    1   |  10.0.0.1 |    5678  |    2   | 100.0.0.1 |
       +---------+--------+-----------+----------+--------+-----------+

               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 indicating that the former information has to be deleted and the source address of the actual packet added.

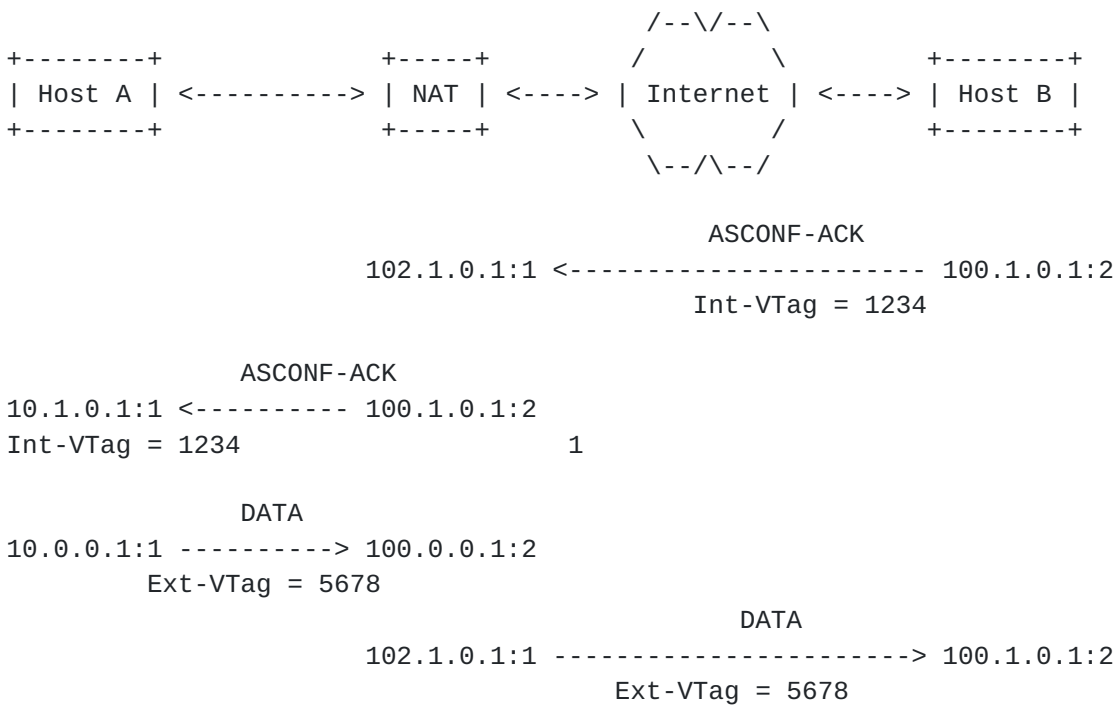



```
ASCONF [ADD-IP,DELETE-IP,Int-VTag=1234, Ext-VTag = 5678]
10.0.0.1:1 ----------> 100.1.0.1:2
          Ext-VTag = 5678
```

NAT A	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	1234	1	10.0.0.1	5678	2	100.0.0.1

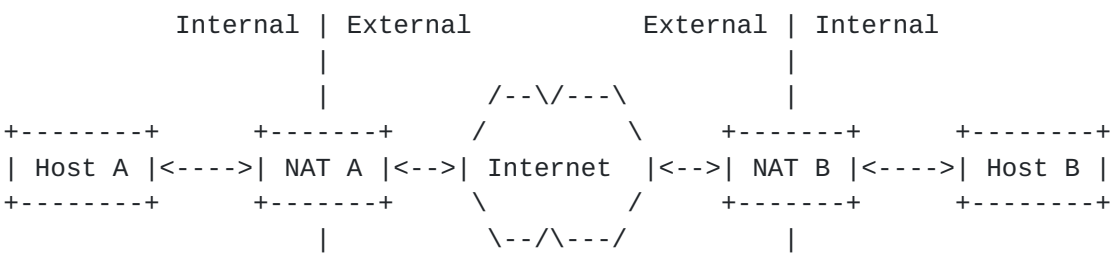
```
        ASCONF [ADD-IP,DELETE-IP,Int-VTag=1234, Ext-VTag = 5678]
                      102.1.0.1:1 ----------------------> 100.1.0.1:2
                                        Ext-VTag = 5678
```

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



10.5. 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

	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
NAT A						

	Int v-tag	Int port	Priv addr	Ext v-tag	Ext port	Ext addr
NAT B						

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

NAT A creates entry:

	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
NAT A	1234	1	10.0.0.1	0	2	100.0.0.1

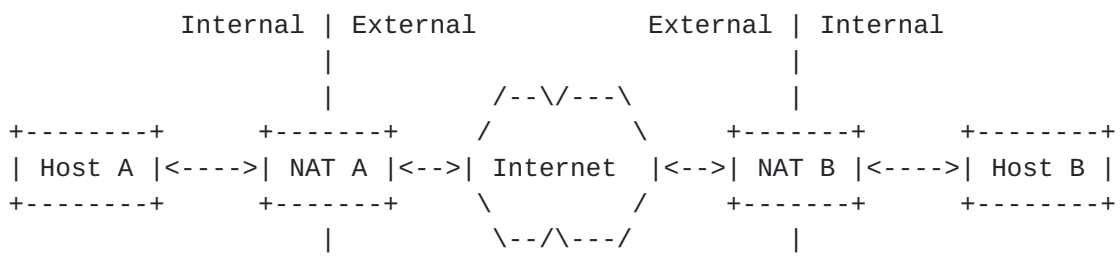
INIT[Initiate-Tag = 1234]
101.0.0.1:1 ----------> 100.0.0.1:2
Ext-VTag = 0

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

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

Now Host B sends INIT, which is processed by NAT B. Its parameters

are used to create an entry.

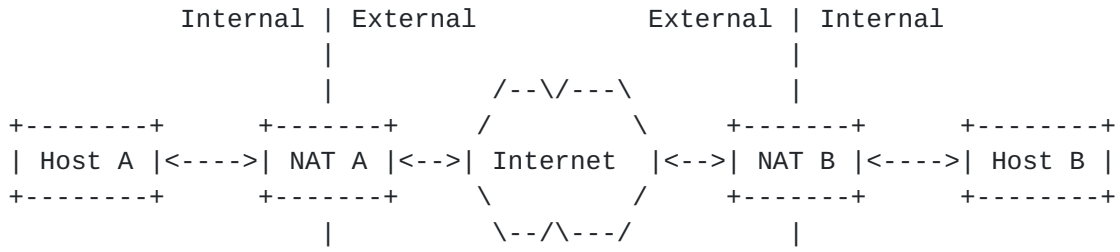


INIT[Initiate-Tag = 5678]
101.0.0.1:1 <-- 10.1.0.1:2
Ext-VTag = 0

NAT B	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	5678	2	10.1.0.1	0	1	101.0.0.1

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:

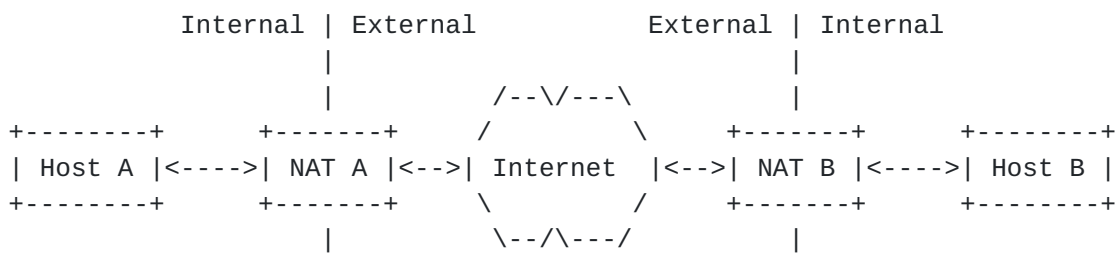


VTag != Int-VTag, but Ext-VTag == 0, find entry.

NAT A	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	1234	1	10.0.0.1	5678	2	100.0.0.1

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

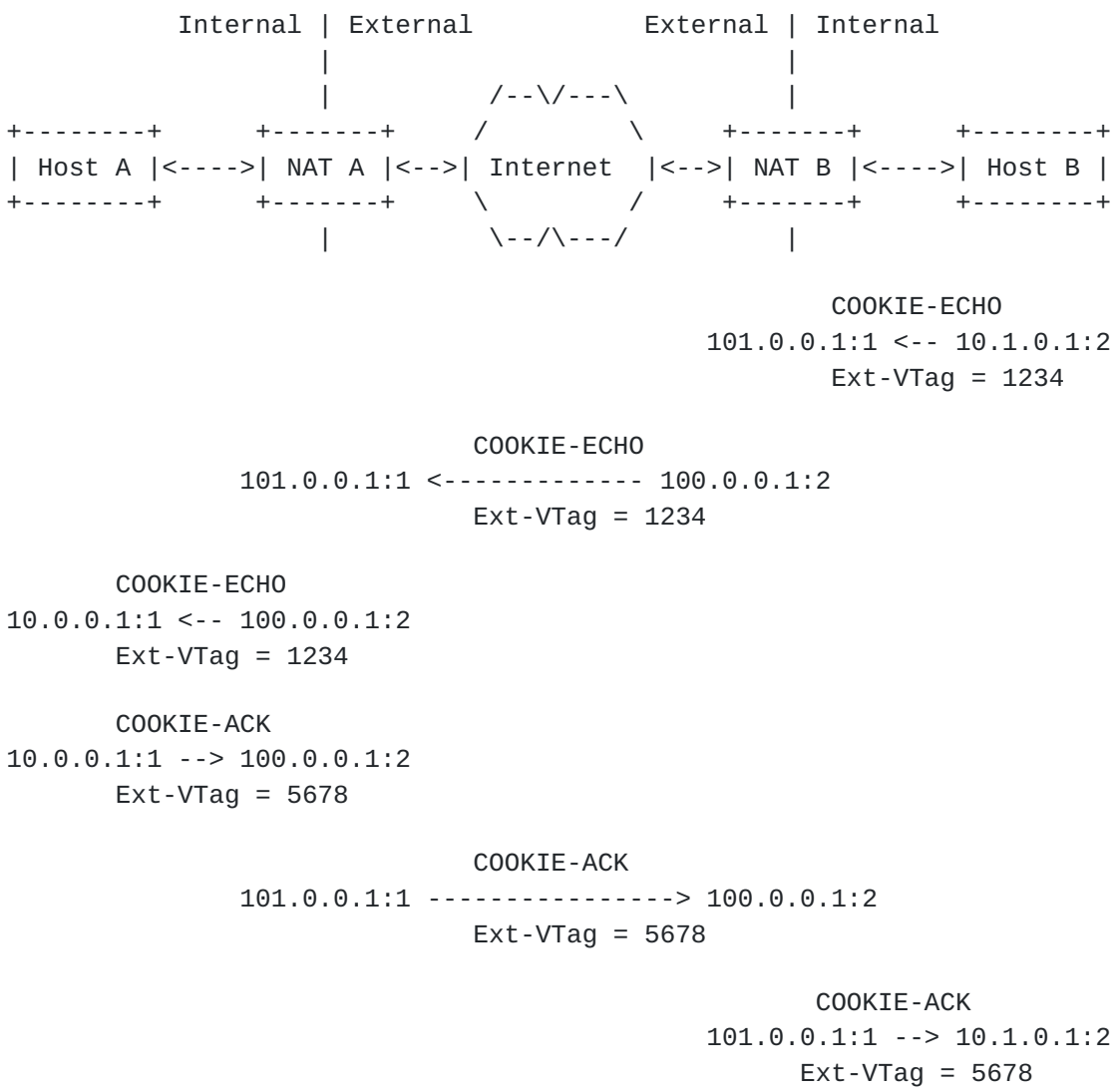
INIT-ACK[Initiate-Tag = 1234]
101.0.0.1:1 ----------> 100.0.0.1:2
Ext-VTag = 5678

NAT B updates entry:

NAT B	Int VTag	Int Port	Priv Addr	Ext VTag	Ext Port	Ext Addr
	5678	2	10.1.0.1	1234	1	101.0.0.1

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.



11. IANA Considerations

TBD

12. 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.

13. Acknowledgments

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

14. References

14.1. 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.

14.2. Informative References

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

Authors' Addresses

Randall R. Stewart
Huawei
Chapin, SC 29036
USA

Phone:
Email: randall@lakerest.net

Michael Tuexen
Muenster Univ. of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Irene Ruengeler
Muenster Univ. of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
Germany

Email: i.ruengeler@fh-muenster.de