Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

R. Stewart
Netflix, Inc.
M. Tuexen
I. Ruengeler
Muenster Univ. of Appl. Sciences
November 4, 2019

Stream Control Transmission Protocol (SCTP) Network Address Translation Support

draft-ietf-tsvwg-natsupp-14

Abstract

The Stream Control Transmission Protocol (SCTP) provides a reliable communications channel between two end-hosts in many ways similar to the Transmission Control Protocol (TCP). 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 (NAPT).

This document describes the protocol extensions required for the SCTP endpoints and the mechanisms for NAT devices necessary to provide 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 https://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 May 7, 2020.

Table of Contents

1. Introduction 3
2. Conventions 5
3. Terminology 5
4. Motivation 6
4.1. SCTP NAT Traversal Scenarios 6
4.1.1. Single Point Traversal 6
4.1.2. Multi Point Traversal 7
4.2. Limitations of Classical NAPT for SCTP 8
4.3. The SCTP-Specific Variant of NAT 8
5. Data Formats 12
5.1. Modified Chunks 12
5.1.1. Extended ABORT Chunk 12
5.1.2. Extended ERROR Chunk 13
5.2. New Error Causes 13
5.2.1. VTag and Port Number Collision Error Cause 13
5.2.2. Missing State Error Cause 14
5.2.3. Port Number Collision Error Cause 15
5.3. New Parameters 15
5.3.1. Disable Restart Parameter 16
5.3.2. VTags Parameter 16
6. Procedures for SCTP Endpoints and NAT Devices 17
6.1. Association Setup Considerations for Endpoints 18
6.2. Handling of Internal Port Number and Verification Tag Collisions 18
6.2.1. NAT Device Considerations 19
6.2.2. Endpoint Considerations 19
6.3. Handling of Internal Port Number Collisions 19
6.3.1. NAT Device Considerations 20
6.3.2. Endpoint Considerations 20
6.4. Handling of Missing State 21
6.4.1. NAT Device Considerations 21

6.4.2. Endpoint Considerations 21
6.5. Handling of Fragmented SCTP Packets by NAT Devices . . . 23
6.6. Multi-Point Traversal Considerations for Endpoints . . . 23
7. Various Examples of NAT Traversals 23
7.1. Single-homed Client to Single-homed Server 23
7.2. Single-homed Client to Multi-homed Server 25
7.3. Multihomed Client and Server 28
7.4. NAT Loses Its State 32
7.5. Peer-to-Peer Communication 34
8. Socket API Considerations 39
8.1. Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY) 40
9. IANA Considerations 40
9.1. New Chunk Flags for Two Existing Chunk Types 40
9.2. Three New Error Causes 41
9.3. Two New Chunk Parameter Types 42
10. Security Considerations 42
11. Acknowledgments 42
12. References 43
12.1. Normative References 43
12.2. Informative References 43
Authors' Addresses 44

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 using private addresses (see [RFC6890]) 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 (NAPT). Please note that this document focuses on the case where the NAT maps multiple private addresses to a single public address. To date, specialized code for SCTP has not yet been added to most NAT devices so that only a translation of IP addresses is supported. The end result of this is that only one SCTP-capable host can successfully operate behind such a NAT and this host can only be single-homed. The only alternative for supporting legacy NAT devices is to use UDP encapsulation as specified in [RFC6951].

This document specifies procedures allowing a NAT to support SCTP by providing similar features to those provided by a NAPT for TCP and other supported protocols. The document also specifies a set of data formats for SCTP packets and a set of SCTP endpoint procedures to support NAT traversal. An SCTP implementation supporting these

procedures can assure that in both single-homed and multi-homed cases a NAT will maintain the appropriate state without the NAT needing to change port numbers.

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 in the same way that a TCP session can use a NAT.

- 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 NAT devices easier.

An SCTP-aware NAT will need to follow these procedures for generating appropriate SCTP packet formats.

When considering this feature it is possible to have multiple levels of support. At each level, the Internal Host, External Host and NAT may or may not support the features described in this document. The following table illustrates the results of the various combinations of support and if communications can occur between two endpoints.

Internal Host	NAT Device	External Host	Communication
Support	Support	Support	Yes
Support	Support	No Support	Limited
Support	No Support	Support	None
Support	No Support	No Support	None
No Support	Support	Support	Limited
No Support	Support	No Support	Limited
No Support	No Support	Support	None
No Support	No Support	No Support	None

Table 1: Communication possibilities

From the table it can be seen that when a NAT device does not support the extension no communication can occur. This assumes that the NAT device does not handle SCTP packets at all and all SCTP packets sent externally from behind a NAT device are discarded by the NAT. In some cases, where the NAT device supports the feature but one of the two hosts does not support the feature, communication may occur but

in a limited way. For example only one host may be able to have a connection when a collision case occurs.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the following terms, which are depicted in Figure 1. Familiarity with the terminology used in [RFC4960] and [RFC5061] is assumed.

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 SCTP Verification Tag (VTag) (see Section 3.1 of [RFC4960]) 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 device that it uses as a source address when sending packets towards the External-Address.

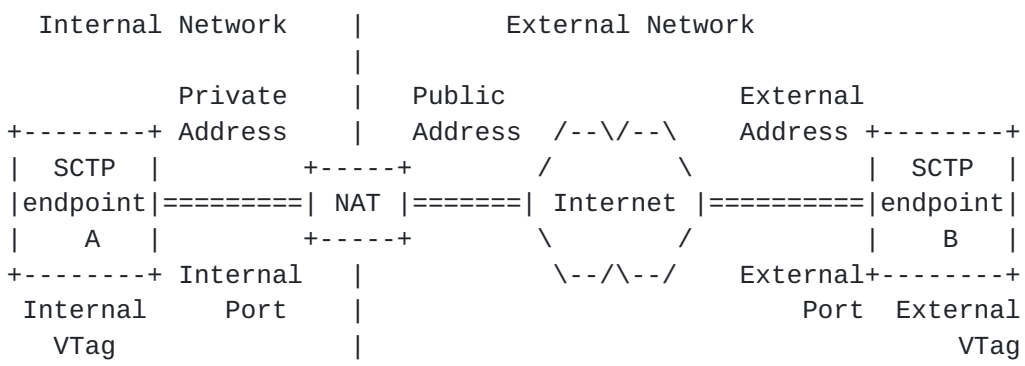


Figure 1: Basic network setup

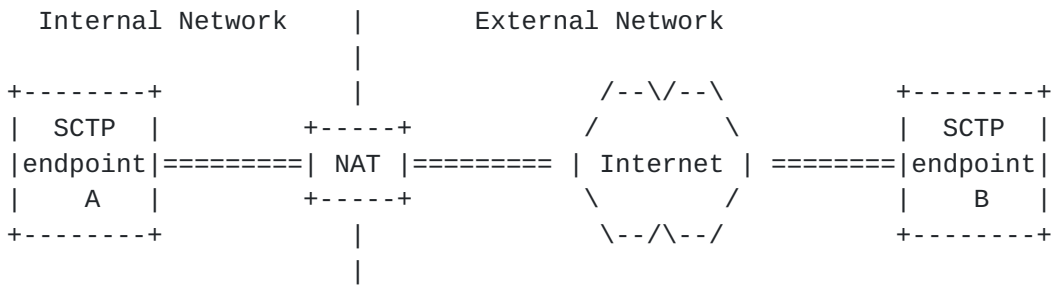
4. Motivation

4.1. SCTP NAT Traversal Scenarios

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

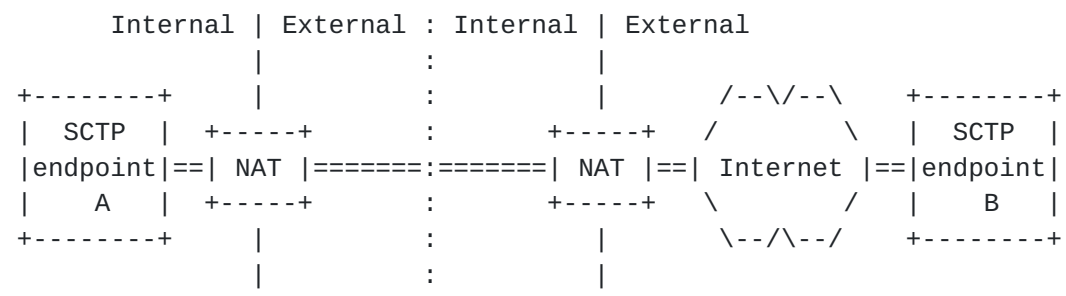
4.1.1. Single Point Traversal

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 NAT devices in a single path:



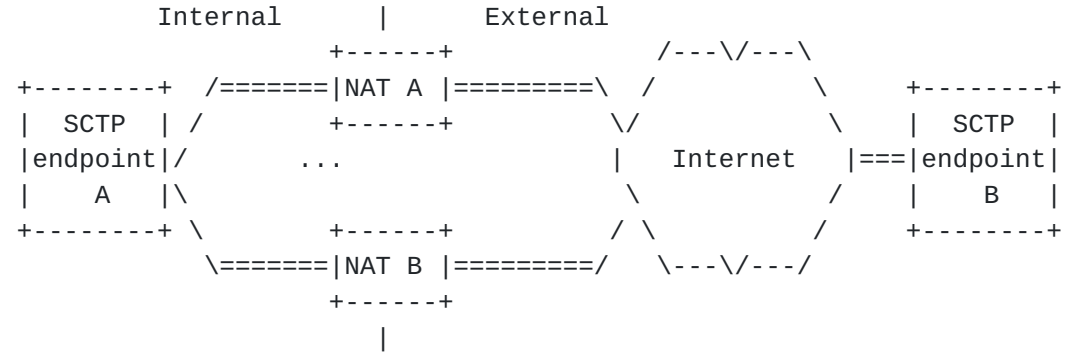
Serial NAT Devices scenario

Although one of the main benefits of SCTP multi-homing is redundant paths, In this single point traversal scenario 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 device (or NAT devices) in this case sees all the packets of the SCTP association.

4.1.2. Multi Point Traversal

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



Parallel NAT devices 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.

4.2. 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 by the NAPT device. Whereas for UDP and TCP this can be done very efficiently, for SCTP the checksum (CRC32c) over the entire packet needs to be recomputed. See Appendix B of [RFC4960] for details of the CRC32c computation. This would considerably add 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 NAT devices 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 a pre-defined table of ports and addresses configured within each NAT. Other mechanisms could make use of NAT to NAT communication. Such mechanisms have not been deployabled on a wide scale base and thus are not a recommended solution. Therefore the SCTP variant of NAT has been developed.

4.3. The SCTP-Specific Variant of NAT

In this section it is allowed that there are multiple SCTP capable hosts behind a NAT that has one Public-Address. Furthermore this section focuses on the single point traversal scenario.

The modification of SCTP packets sent to the public Internet is simple: 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 device to later handle incoming packets.

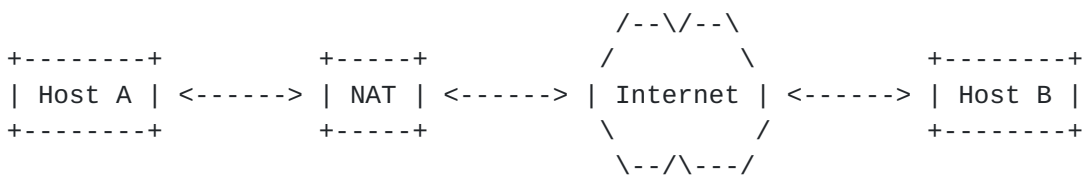
For the SCTP NAT processing the NAT device has to maintain a NAT binding table of Internal-VTag, Internal-Port, External-VTag, External-Port, Private-Address, and whether the restart procedure is disabled or not. An entry in that NAT binding table is called a NAT state control block. The function Create()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

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, Internal-VTag and the Internal-Port.

The entries in the NAT binding table need to fulfill some uniqueness conditions. There must not be more than one entry NAT binding table with the same pair of Internal-Port and External-Port. This rule can

be relaxed, if all NAT binding table 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 NAT binding table 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, 0, Ext-Port, Priv-Addr,
                    RestartSupported)
             Returns(NAT-State control block)

           Translate To:

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

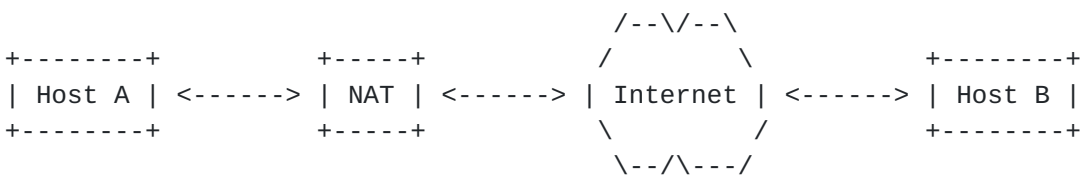
Normally a NAT binding table entry will be created.

However, it is possible that there is already a NAT binding table entry with the same External-Address, External-Port, Internal-Port, and Internal-VTag but different Private-Address. In this case the INIT MUST be dropped by the NAT and an ABORT MUST be sent back to the SCTP host with the M-Bit set and an appropriate error cause (see Section 5.1.1 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 binding table entry 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 Section 5.1.1 for the format).

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

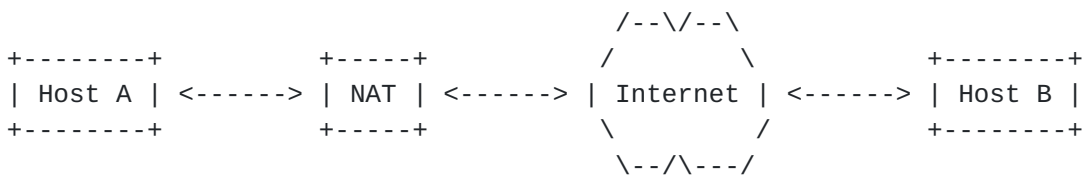


Priv-Addr:Int-Port ------> Ext-Addr:Ext-Port
Ext-VTag

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, and External-Port, returns the corresponding entry of the NAT binding 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

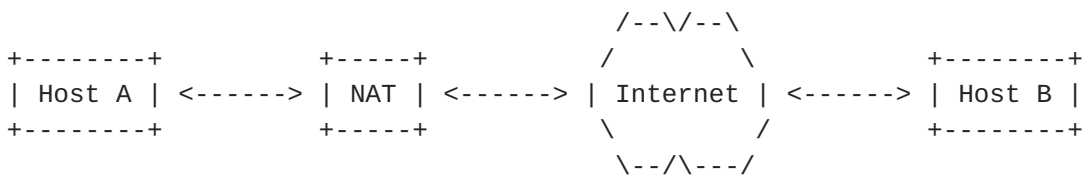
Lookup(Int-VTag, Int-Port, *, Ext-Port)
Update(*, *, Initiate-Tag, *)

Returns(NAT-State control block containing Priv-Addr)

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.



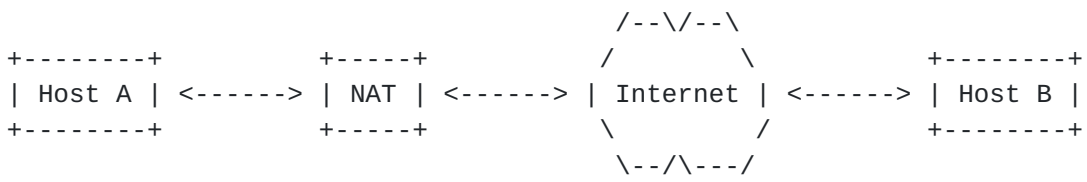
Pub-Addr:Int-Port <------ Ext-Addr:Ext-Port
Ext-VTag

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

Returns(NAT-State control block containing Priv-Addr)

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.

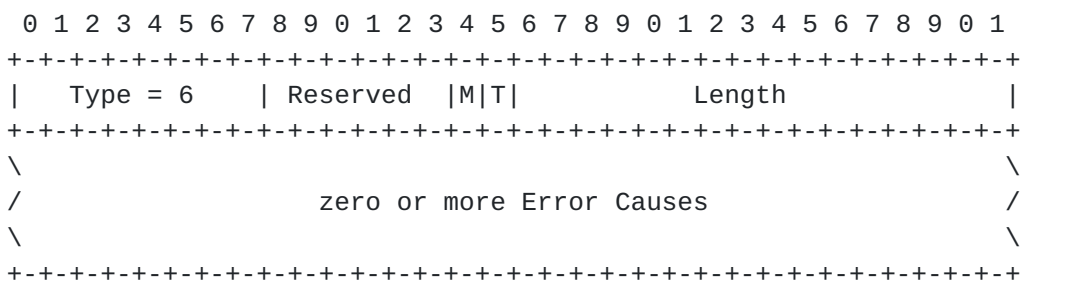
5. Data Formats

This section defines the formats used to support NAT traversal. Section 5.1 and Section 5.2 describe chunks and error causes sent by NAT devices and received by SCTP endpoints. Section 5.3 describes parameters sent by SCTP endpoints and used by NAT devices and SCTP endpoints.

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] that are modified by this document.

5.1.1. Extended ABORT Chunk



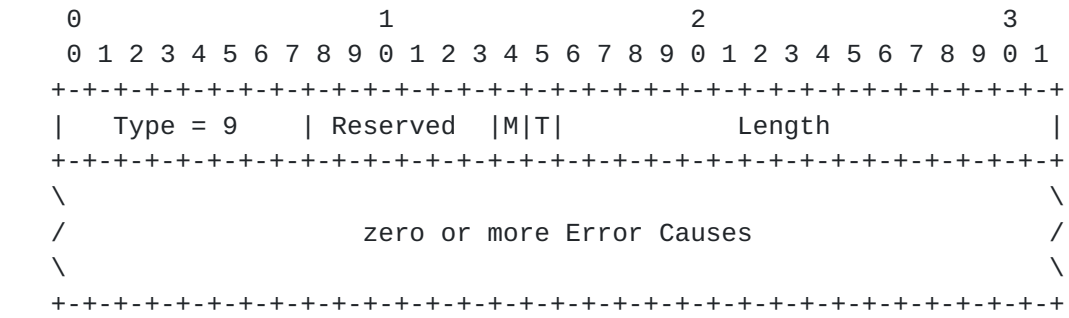
The ABORT chunk is extended to add the new 'M-bit'. The M-bit indicates to the receiver of the ABORT chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[NOTE to RFC-Editor:

Assignment of M-bit to be confirmed by IANA.

]

5.1.2. Extended ERROR Chunk



The ERROR chunk defined in [RFC4960] is extended to add the new 'M-bit'. The M-bit indicates to the receiver of the ERROR chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[NOTE to RFC-Editor:

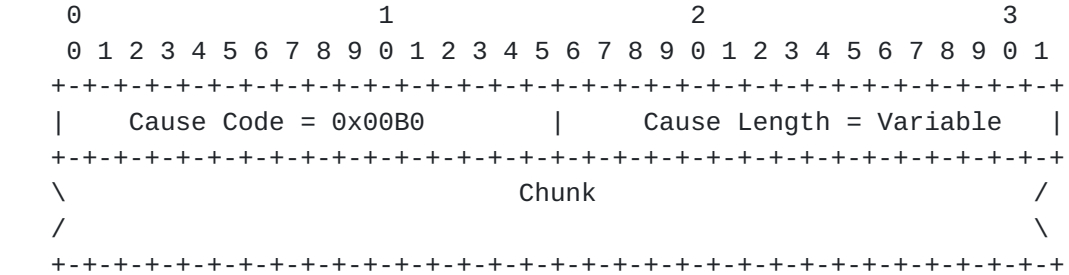
Assignment of M-bit to be confirmed by IANA.

]

5.2. New Error Causes

This section defines the new error causes added by this document.

5.2.1. VTag and Port Number Collision Error Cause



Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'VTag and Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B0 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

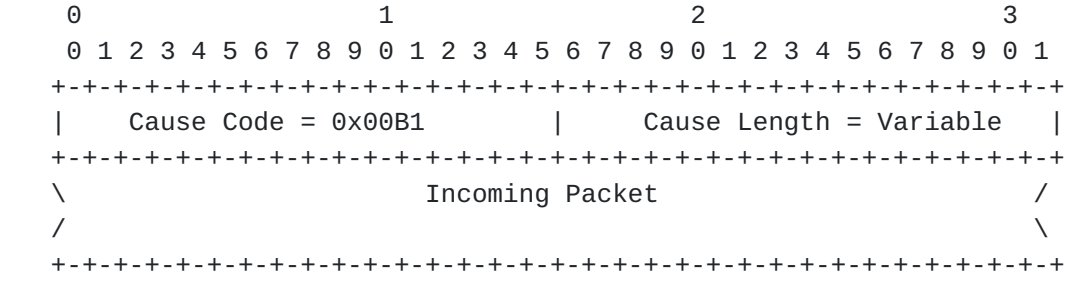
Chunk: variable length
The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT-ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor:

Assignment of cause code to be confirmed by IANA.

]

5.2.2. Missing State Error Cause



Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'Missing State' Error Cause. IANA is requested to assign the value 0x00B1 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

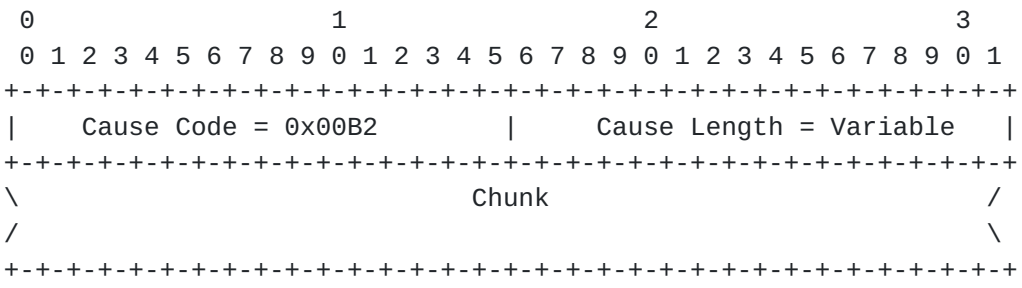
Incoming Packet: variable length
The Cause-Specific Information is filled with the IPv4 or IPv6 packet that caused this error. The IPv4 or IPv6 header MUST be included. Note that if the packet will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor:

Assignment of cause code to be confirmed by IANA.

]

5.2.3. Port Number Collision Error Cause



Cause Code: 2 bytes (unsigned integer)
This field holds the IANA defined cause code for the 'Port Number Collision' Error Cause. IANA is requested to assign the value 0x00B2 for this cause code.

Cause Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the error cause. The value MUST be the length of the Cause-Specific Information plus 4.

Chunk: variable length
The Cause-Specific Information is filled with the chunk that caused this error. This can be an INIT, INIT-ACK, or ASCONF chunk. Note that if the entire chunk will not fit in the ERROR chunk or ABORT chunk being sent then the bytes that do not fit are truncated.

[NOTE to RFC-Editor:

Assignment of cause code to be confirmed by IANA.

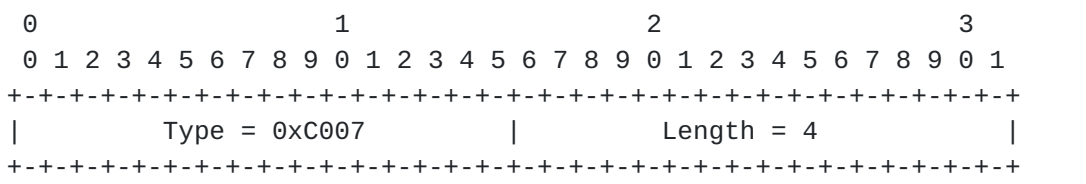
]

5.3. New Parameters

This section defines new parameters and their valid appearance defined by this document.

5.3.1. Disable Restart Parameter

This parameter is used to indicate that the RESTART procedure is requested to be disabled. Both endpoints of an association MUST include this parameter in the INIT chunk and INIT-ACK chunk when establishing an association and MUST include it in the ASCONF chunk when adding an address to successfully disable the restart procedure.



Parameter Type: 2 bytes (unsigned integer)
This field holds the IANA defined parameter type for the Disable Restart Parameter. IANA is requested to assign the value 0xC007 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 4.

[NOTE to RFC-Editor:

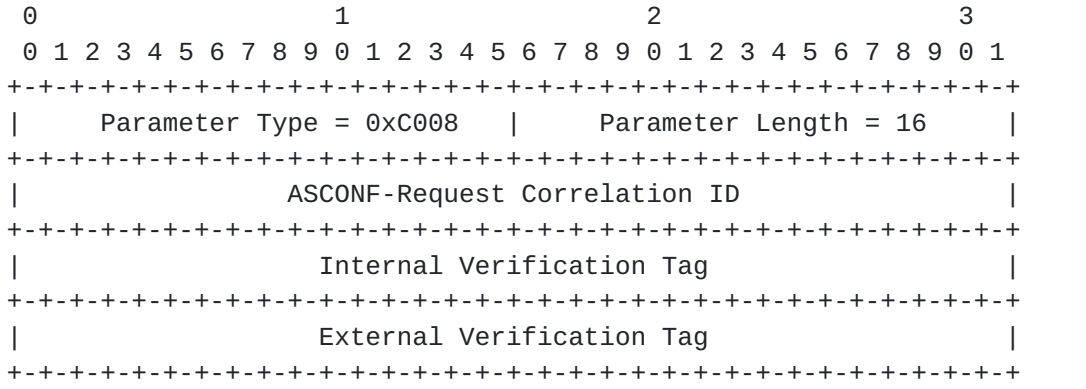
Assignment of parameter type to be confirmed by IANA.

]

This parameter MAY appear in INIT, INIT-ACK and ASCONF chunks and MUST NOT appear in any other chunk.

5.3.2. VTags Parameter

This parameter is used to help a NAT recover from state loss.



Parameter Type: 2 bytes (unsigned integer)
This field holds the IANA defined parameter type for the VTags Parameter. IANA is requested to assign the value 0xC008 for this parameter type.

Parameter Length: 2 bytes (unsigned integer)
This field holds the length in bytes of the parameter. The value MUST be 16.

ASCONF-Request Correlation ID: 4 bytes (unsigned integer)
This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32-bit value into the ASCONF Response Correlation ID field of the ASCONF-ACK response parameter. The sender of the ASCONF can use this same value in the ASCONF-ACK to find which request the response is for. Note that the receiver MUST NOT change this 32-bit value.

Internal Verification Tag: 4 bytes (unsigned integer)
The Verification Tag that the internal host has chosen for its communication. The Verification Tag is a unique 32-bit tag that must accompany any incoming SCTP packet for this association to the Private-Address.

External Verification Tag: 4 bytes (unsigned integer) 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.

[NOTE to RFC-Editor:

Assignment of parameter type to be confirmed by IANA.

]

This parameter MAY appear in ASCONF chunks and MUST NOT appear in any other chunk.

6. Procedures for SCTP Endpoints and NAT Devices

When an SCTP endpoint is behind an SCTP-aware NAT a number of problems may arise as it tries to communicate with its peer:

- IP addresses can not not be included in the SCTP packet. This is discussed in Section 6.1.

- More than one host behind a NAT device could select the same VTag and source port when talking to the same peer server. This creates a situation where the NAT will not be able to tell the two associations apart. This situation is discussed in Section 6.2.

- When an SCTP endpoint is a server communicating with multiple peers and the peers are behind the same NAT, then the two endpoints cannot be distinguished by the server. This case is discussed in Section 6.3.

- A restart of a NAT during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.4.

- NAT devices need to deal with SCTP packets being fragmented at the IP layer. This is discussed in Section 6.5.

- An SCTP endpoint may be behind two NAT devices providing redundancy. The method to set up this scenario is discussed in Section 6.6.

Each of these mechanisms requires additional chunks and parameters, defined in this document, and possibly modified handling procedures from those specified in [RFC4960].

6.1. Association Setup Considerations for Endpoints

The association setup procedure defined in [RFC4960] allows multi-homed SCTP endpoints to exchange its IP-addresses by using IPv4 or IPv6 address parameters in the INIT and INIT-ACK chunks. However, this doesn't work when NAT devices are present.

Every association MUST initially be set up single-homed. There MUST NOT be any IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter in the INIT-chunk. The INIT-ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter.

If the association should finally be multi-homed, the procedure in Section 6.6 MUST be used.

The INIT and INIT-ACK chunk SHOULD contain the Disable Restart parameter defined in Section 5.3.1.

6.2. Handling of Internal Port Number and Verification Tag Collisions

Consider the case where two hosts in the Private-Address space want to set up an SCTP association with the same service provided by some hosts in the Internet. This means that the External-Port is the

same. If they both choose the same Internal-Port and Internal-VTag, the NAT device cannot distinguish between incoming packets anymore. But this is very unlikely. The Internal-VTags are chosen at random and if the Internal-Ports are also chosen from the ephemeral port range at random this gives a 46-bit random number that has to match. A NAPT device can control the 16-bit Natted Port and therefore avoid collisions deterministically.

The same can happen with the External-VTag when an INIT-ACK chunk or an ASCONF chunk is processed by the NAT.

6.2.1. NAT Device Considerations

If the NAT device detects a collision of internal port numbers and verification tags, it MUST send an ABORT chunk with the M-bit set if the collision is triggered by an INIT or INIT-ACK chunk. If such a collision is triggered by an ASCONF chunk, it MUST send an ERROR chunk with the M-bit. The M-bit is a new bit defined by this document to express to SCTP that the source of this packet is a "middle" box, not the peer SCTP endpoint (see Section 5.1.1). If a packet containing an INIT-ACK chunk triggers the collision, the corresponding packet containing the ABORT chunk MUST contain the same source and destination address and port numbers as the packet containing the INIT-ACK chunk. In the other two cases, the source and destination address and port numbers MUST be swapped.

The sender of the ERROR or ABORT chunk MUST include the error cause with cause code 'VTag and Port Number Collision' (see Section 5.2.1).

6.2.2. Endpoint Considerations

The sender of the packet containing the INIT chunk or the receiver of the INIT-ACK chunk, upon reception of an ABORT chunk with M-bit set and the appropriate error cause code for colliding NAT binding table state is included, MUST reinitiate the association setup procedure after choosing a new initiate tag, if the association is in COOKIE-WAIT state. In any other state, the SCTP endpoint MUST NOT respond.

The sender of the ASCONF chunk, upon reception of an ERROR chunk with M-bit set, MUST stop adding the path to the association.

6.3. Handling of Internal Port Number Collisions

When two SCTP hosts are behind an SCTP-aware NAT it is possible that two SCTP hosts in the Private-Address space will want to set up an SCTP association with the same server running on the same host in the Internet. For the NAT, appropriate tracking may be performed by assuring that the VTags are unique between the two hosts.

6.3.1. NAT Device Considerations

The NAT, when processing the INIT-ACK, should note in its NAT binding table that the association supports the Disable Restart extension. This note is used when establishing future associations (i.e. when processing an INIT from an internal host) to decide if the connection should be allowed. The NAT device does the following when processing an INIT:

- If the INIT is destined to an external address and port for which the NAT device has no outbound connection, it MUST allow the INIT creating an NAT binding table entry.

- If the INIT matches the external address and port of an already existing connection, it MUST validate that the external server supports the Disable Restart feature and, if it does, allow the INIT to be forwarded.

- If the external server does not support the Disable Restart extension the NAT device MUST send an ABORT with the M-bit set.

The 'Port Number Collision' error cause (see Section 5.2.3) MUST be included in the ABORT chunk sent in response to the INIT chunk.

If the collision is triggered by an ASCONF chunk, a packet containing an ERROR chunk with the 'Port Number Collision' error cause MUST be sent in response to the ASCONF chunk.

6.3.2. Endpoint Considerations

For the external SCTP server on the Internet this means that the External-Port and the External-Address are the same. If they both have chosen the same Internal-Port the server cannot distinguish between both associations based on the address and port numbers. For the server it looks like the association is being restarted. To overcome this limitation the client sends a Disable Restart parameter in the INIT-chunk.

When the server receives this parameter it does the following:

- It MUST include a Disable Restart parameter in the INIT-ACK to inform the client that it will support the feature.

- It MUST Disable the restart procedures defined in [RFC4960] for this association.

Servers that support this feature will need to be capable of maintaining multiple connections to what appears to be the same peer (behind the NAT) differentiated only by the VTags.

6.4. Handling of Missing State

6.4.1. NAT Device Considerations

If the NAT device receives a packet from the internal network for which the lookup procedure does not find an entry in the NAT binding table, a packet containing an ERROR chunk is sent back with the M-bit set. The source address of the packet containing the ERROR chunk MUST be the destination address of the incoming SCTP packet. The verification tag is reflected and the T-bit is set. Such a packet containing an ERROR chunk SHOULD NOT be sent if the received packet contains an ABORT, SHUTDOWN-COMPLETE or INIT-ACK chunk. An ERROR chunk MUST NOT be sent if the received packet contains an ERROR chunk with the M-bit set.

When sending the ERROR chunk, the error cause 'Missing State' (see Section 5.2.2) MUST be included and the M-bit of the ERROR chunk MUST be set (see Section 5.1.2).

If the NAT device receives a packet for which it has no NAT binding table entry and the packet contains an ASCONF chunk with the VTags parameter, the NAT device MUST update its NAT binding table according to the verification tags in the VTags parameter and the optional Disable Restart parameter.

6.4.2. Endpoint Considerations

Upon reception of this ERROR chunk by an SCTP endpoint the receiver takes the following actions:

- It SHOULD validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet. If the validation fails, discard the incoming ERROR chunk.

- It SHOULD validate that the peer of the SCTP association supports the dynamic address extension. If the validation fails, discard the incoming ERROR chunk.

- It SHOULD generate a new ASCONF chunk containing the VTags parameter (see Section 5.3.2) and the Disable Restart parameter if the association is using the disabled restart feature. By processing this packet the NAT device can recover the appropriate

state. The procedures for generating an ASCONF chunk can be found in [RFC5061].

The peer SCTP endpoint receiving such an ASCONF chunk SHOULD either add the address and respond with an acknowledgment, if the address is new to the association (following all procedures defined in [RFC5061]). Or, if the address is already part of the association, the SCTP endpoint MUST NOT respond with an error, but instead SHOULD respond with an ASCONF-ACK chunk acknowledging the address and take no action (since the address is already in the association).

Note that it is possible that upon receiving an ASCONF chunk containing the VTags parameter the NAT will realize that it has an 'Internal Port Number and Verification Tag collision'. In such a case the NAT MUST send an ERROR chunk with the error cause code set to 'VTag and Port Number Collision' (see Section 5.2.1).

If an SCTP endpoint receives an ERROR with 'Internal Port Number and Verification Tag collision' as the error cause and the packet in the Error Chunk contains an ASCONF with the VTags parameter, careful examination of the association is required. The endpoint does the following:

- It MUST validate that the verification tag is reflected by looking at the VTag that would have been included in the outgoing packet. If the validation fails, it MUST discard the packet.

- It MUST validate that the peer of the SCTP association supports the dynamic address extension. If the peer does not support it, the NAT Device MUST discard the incoming ERROR chunk.

- If the association is attempting to add an address (i.e. following the procedures in Section 6.6) then the endpoint MUST NOT consider the address part of the association and SHOULD make no further attempt to add the address (i.e. cancel any ASCONF timers and remove any record of the path), since the NAT devie has a VTag collision and the association cannot easily create a new VTag (as it would if the error occurred when sending an INIT).

- If the endpoint has no other path, i.e. the procedure was executed due to missing a state in the NAT device, then the endpoint MUST abort the association. This would occur only if the local NAT device restarted and accepted a new association before attempting to repair the missing state (Note that this is no different than what happens to all TCP connections when a NAT device looses its state).

6.5. Handling of Fragmented SCTP Packets by NAT Devices

A NAT device 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 device and the IP header forbids fragmentation a corresponding ICMP packet SHOULD be sent.

6.6. Multi-Point Traversal Considerations for Endpoints

If a multi-homed SCTP endpoint behind a NAT connects to a peer, it SHOULD first set up the association single-homed with only one address causing the first NAT to populate its state. Then it SHOULD add each IP address using ASCONF chunks sent via their respective NAT devices. The address to add is the wildcard address and the lookup address SHOULD also contain the VTags parameter and optionally the Disable Restart parameter as illustrated above.

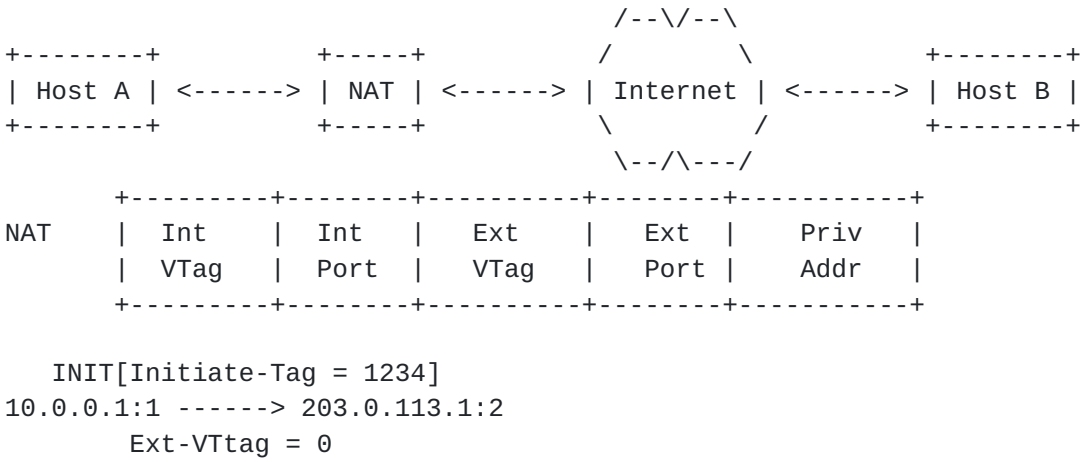
7. Various Examples of NAT Traversals

Please note that this section is informational only.

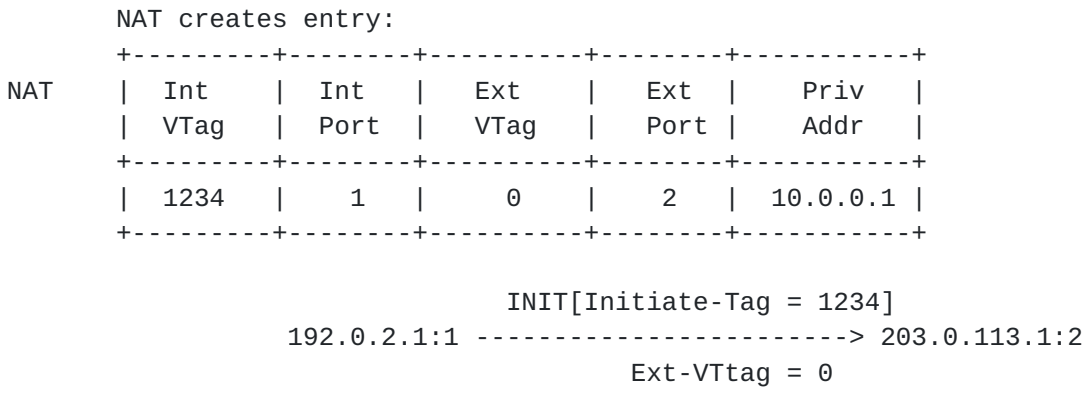
The addresses being used in the following examples are IPv4 addresses for private-use networks and for documentation as specified in [RFC6890]. However, the method described here is not limited to this NAT44 case.

7.1. Single-homed Client to Single-homed Server

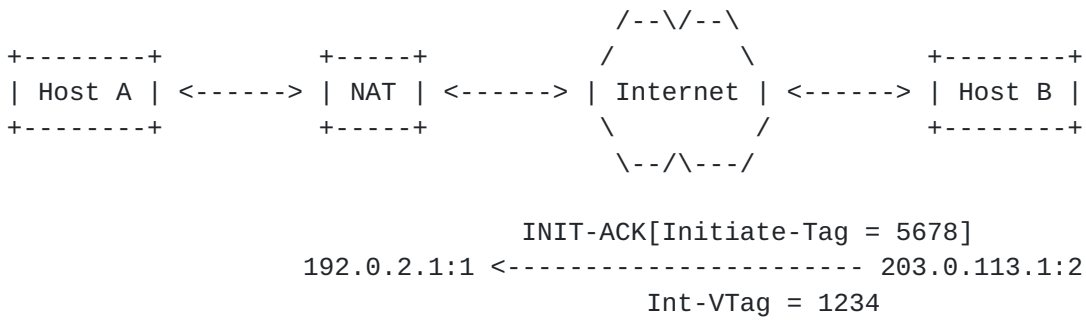
The internal client starts the association with the external server via a four-way-handshake. Host A starts by sending an INIT chunk.



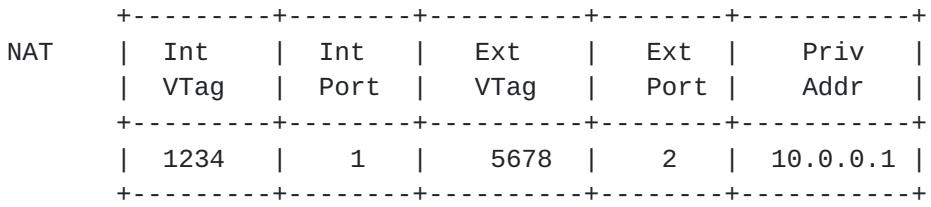
A NAT entry is created, the source address is substituted and the packet is sent on:

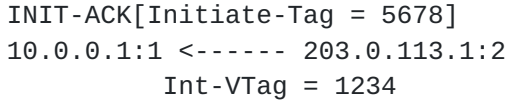


Host B receives the INIT and sends an INIT-ACK with the NAT's external address as destination address.

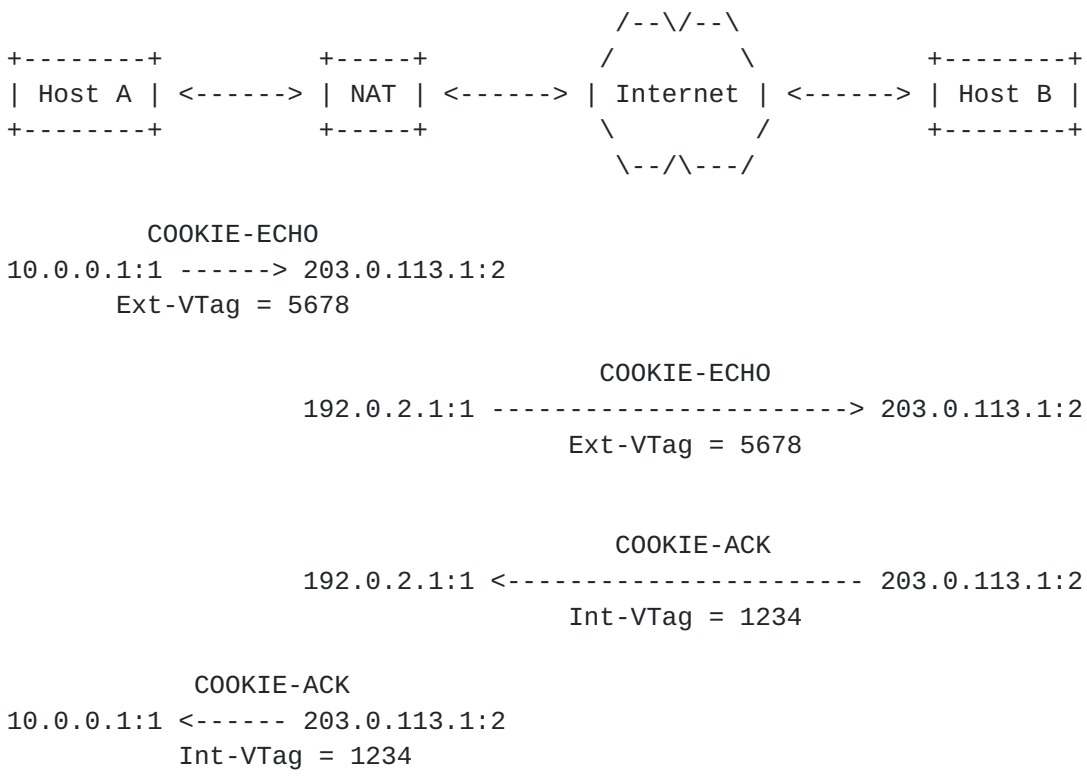


NAT updates entry:



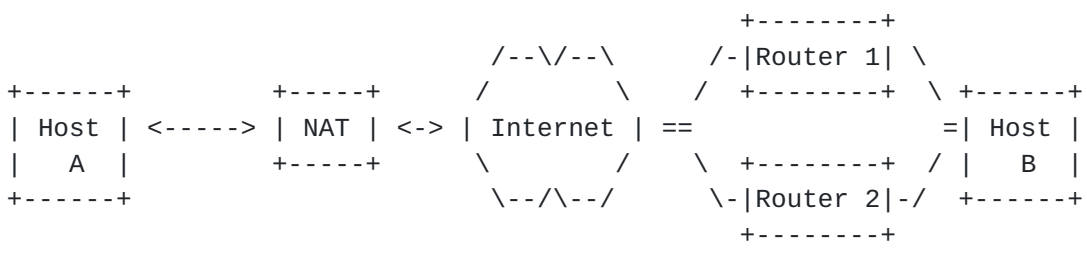


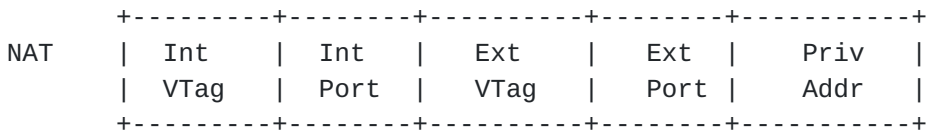
The handshake finishes with a COOKIE-ECHO acknowledged by a COOKIE-ACK.



7.2. Single-homed Client to Multi-homed Server

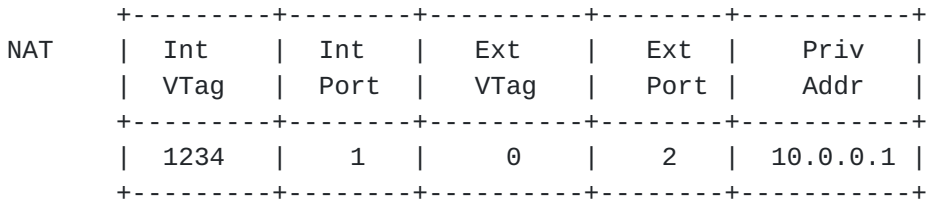
The internal client is single-homed whereas the external server is multi-homed. The client (Host A) sends an INIT like in the single-homed case.





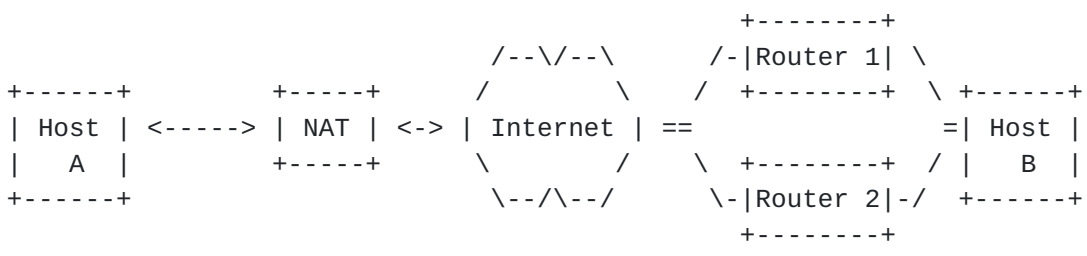
INIT[Initiate-Tag = 1234]
10.0.0.1:1 ---> 203.0.113.1:2
Ext-VTag = 0

NAT creates entry:



INIT[Initiate-Tag = 1234]
192.0.2.1:1 ----------> 203.0.113.1:2
Ext-VTag = 0

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



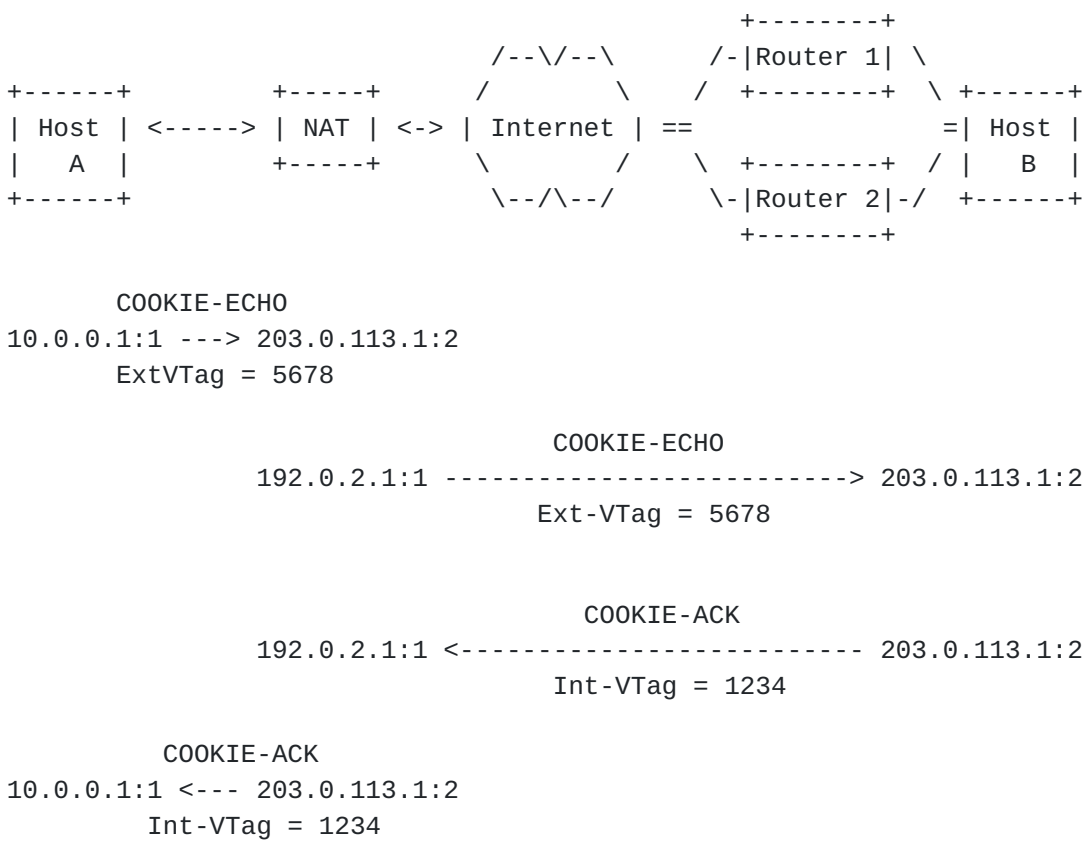
INIT-ACK[Initiate-tag = 5678, IP-Addr = 203.0.113.129]
192.0.2.1:1 <---------- 203.0.113.1:2
Int-VTag = 1234

NAT does need to change the NAT binding table for the second address:

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

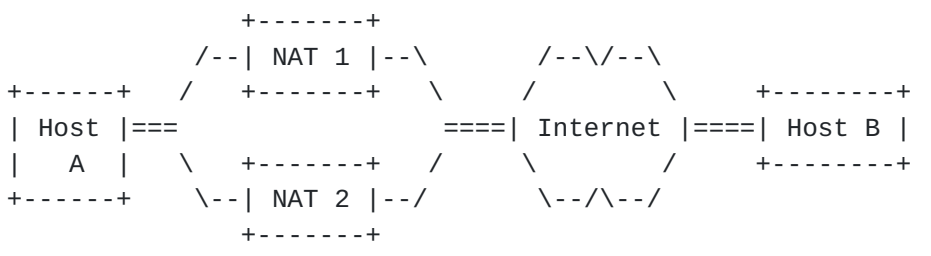
INIT-ACK[Initiate-Tag = 5678]
10.0.0.1:1 <--- 203.0.113.1:2
Int-VTag = 1234

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



7.3. 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	Ext VTag	Ext Port	Priv Addr

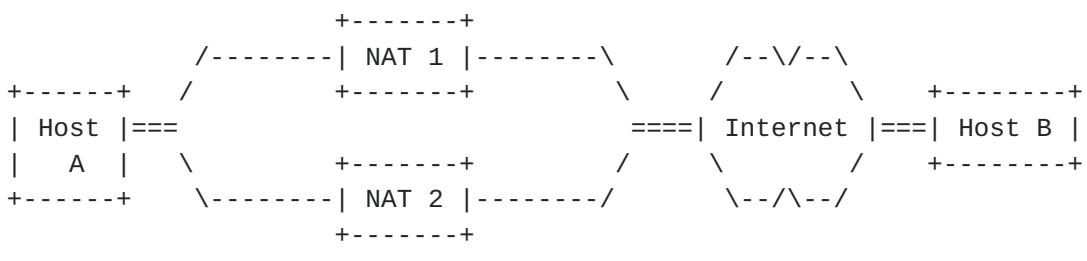
INIT[Initiate-Tag = 1234]
10.0.0.1:1 --------> 203.0.113.1:2
Ext-VTag = 0

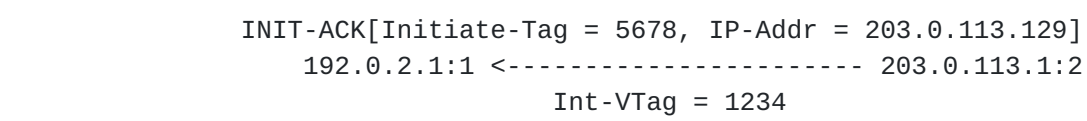
NAT 1 creates entry:

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

INIT[Initiate-Tag = 1234]
192.0.2.1:1 ----------> 203.0.113.1:2
ExtVTag = 0

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

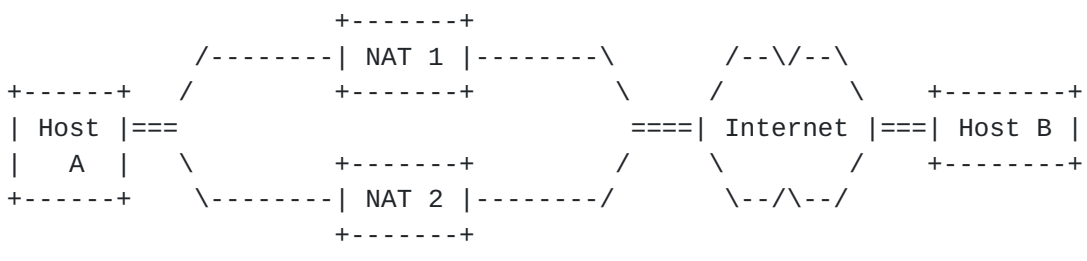




NAT 1 does not need to update the NAT binding table for the second address:

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

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



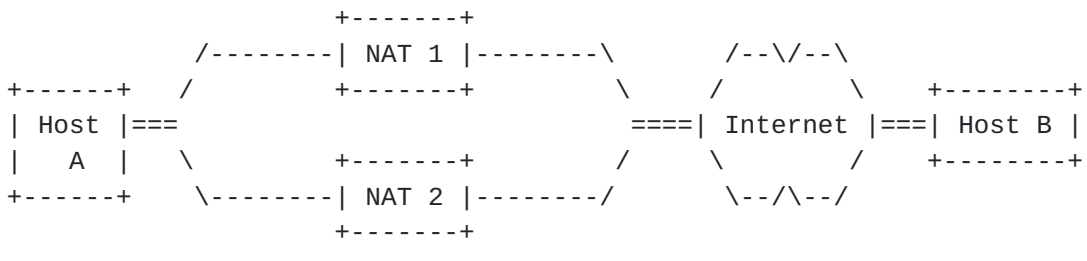
COOKIE-ECHO
10.0.0.1:1 --------> 203.0.113.1:2
Ext-VTag = 5678

COOKIE-ECHO
192.0.2.1:1 ------------------> 203.0.113.1:2
Ext-VTag = 5678

COOKIE-ACK
192.0.2.1:1 <------------------ 203.0.113.1:2
Int-VTag = 1234

COOKIE-ACK
10.0.0.1:1 <------- 203.0.113.1:2
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 NAT binding table entry completely with this single packet.



ASCONF [ADD-IP=0.0.0.0, INT-VTag=1234, Ext-VTag = 5678]
10.1.0.1:1 --------> 203.0.113.129:2
Ext-VTag = 5678

NAT 2 creates a complete entry:

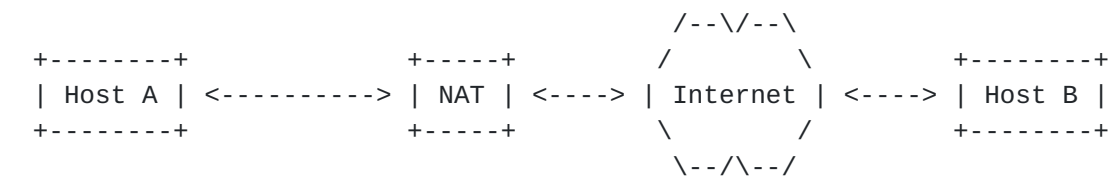
```
       +---------+--------+----------+--------+-----------+
NAT 2  |  Int    |  Int   |   Ext    |   Ext  |    Priv   |
       |  VTag   |  Port  |   VTag   |   Port |    Addr   |
       +---------+--------+----------+--------+-----------+
       |  1234   |    1   |    5678  |    2   |  10.1.0.1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


                ASCONF [ADD-IP, Int-VTag=1234, Ext-VTag = 5678]
                 192.0.2.129:1 ---------------------> 203.0.113.129:2
                                      Ext-VTag = 5678

                                       ASCONF-ACK
                 192.0.2.129:1 <--------------------- 203.0.113.129:2
                                     Int-VTag = 1234

          ASCONF-ACK
10.1.0.1:1 <----- 203.0.113.129:2
         Int-VTag = 1234
```

7.4. 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.

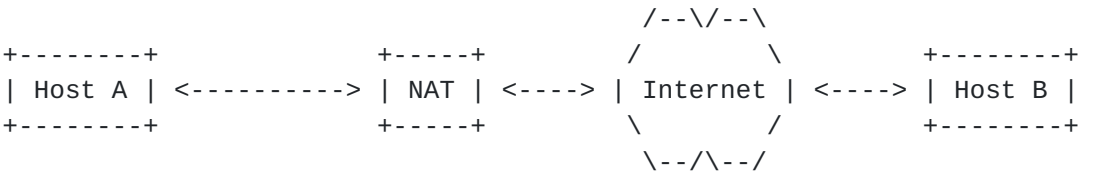
```
                                          /--\/--\
+--------+              +-----+         /        \         +--------+
| Host A | <----------> | NAT | <----> | Internet | <----> | Host B |
+--------+              +-----+         \        /         +--------+
                                   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

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

               DATA
10.0.0.1:1 ----------> 203.0.113.1:2
            Ext-VTag = 5678
```

The NAT device cannot find an entry in the NAT binding table for the association. It sends ERROR an message with the M-Bit set and the cause "NAT state missing".



ERROR [M-Bit, NAT state missing]
10.0.0.1:1 <---------- 203.0.113.1:2
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.



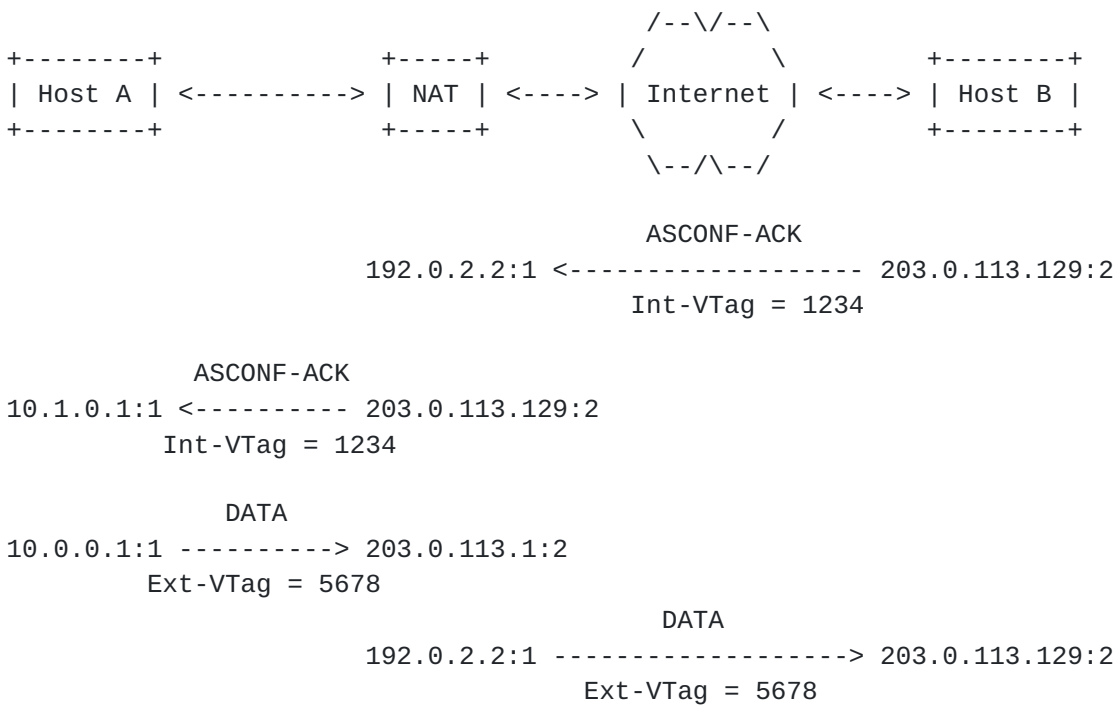
ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Ext-VTag = 5678]
10.0.0.1:1 ----------> 203.0.113.129:2
Ext-VTag = 5678

NAT

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

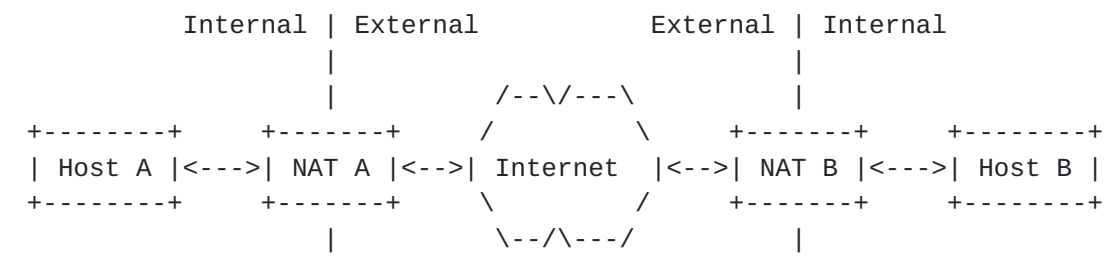
ASCONF [ADD-IP, DELETE-IP, Int-VTag=1234, Ext-VTag = 5678]
192.0.2.2:1 ------------------> 203.0.113.129:2
Ext-VTag = 5678

Host B adds the new source address to this association and deletes all other addresses from this association.



7.5. Peer-to-Peer Communication

If two hosts are behind NAT devices and want to communicate with each other, 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 devices 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 (internal/external) of the verification tag in the packet flow is done from the sending host's point of view.



NAT Binding Tables

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

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

INIT[Initiate-Tag = 1234]
10.0.0.1:1 --> 203.0.113.1:2
Ext-VTag = 0

NAT A creates entry:

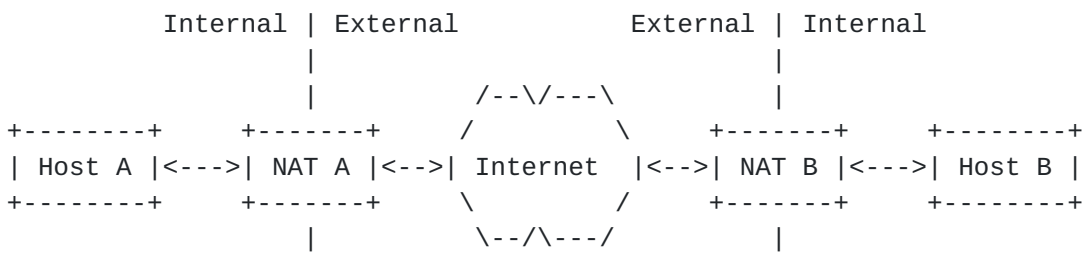
NAT A	Int VTag	Int Port	Ext VTag	Ext Port	Priv Addr
	1234	1	0	2	10.0.0.1

INIT[Initiate-Tag = 1234]
192.0.2.1:1 ----------> 203.0.113.1:2
Ext-VTag = 0

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

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

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



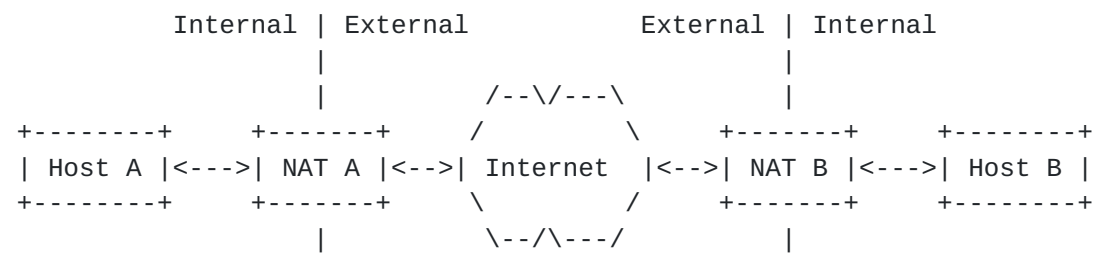
INIT[Initiate-Tag = 5678]
192.0.2.1:1 <-- 10.1.0.1:2
Ext-VTag = 0

NAT B

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

INIT[Initiate-Tag = 5678]
192.0.2.1:1 <--------------- 203.0.113.1:2
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:

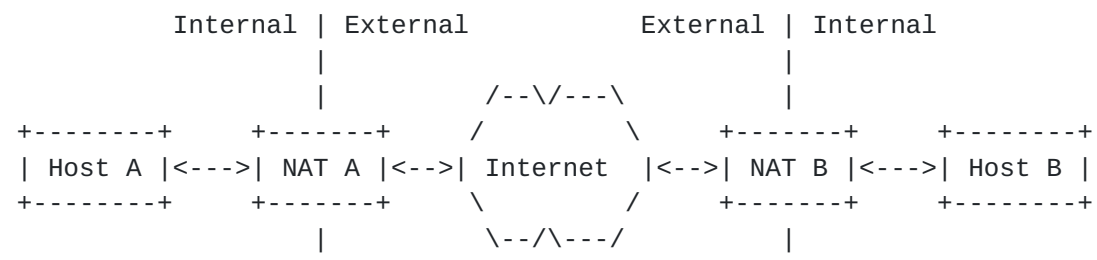


VTag != Int-VTag, but Ext-VTag == 0, find entry.

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

INIT[Initiate-tag = 5678]
10.0.0.1:1 <-- 203.0.113.1:2
Ext-VTag = 0

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



INIT-ACK[Initiate-Tag = 1234]
10.0.0.1:1 --> 203.0.113.1:2
Ext-VTag = 5678

INIT-ACK[Initiate-Tag = 1234]
192.0.2.1:1 ----------> 203.0.113.1:2
Ext-VTag = 5678

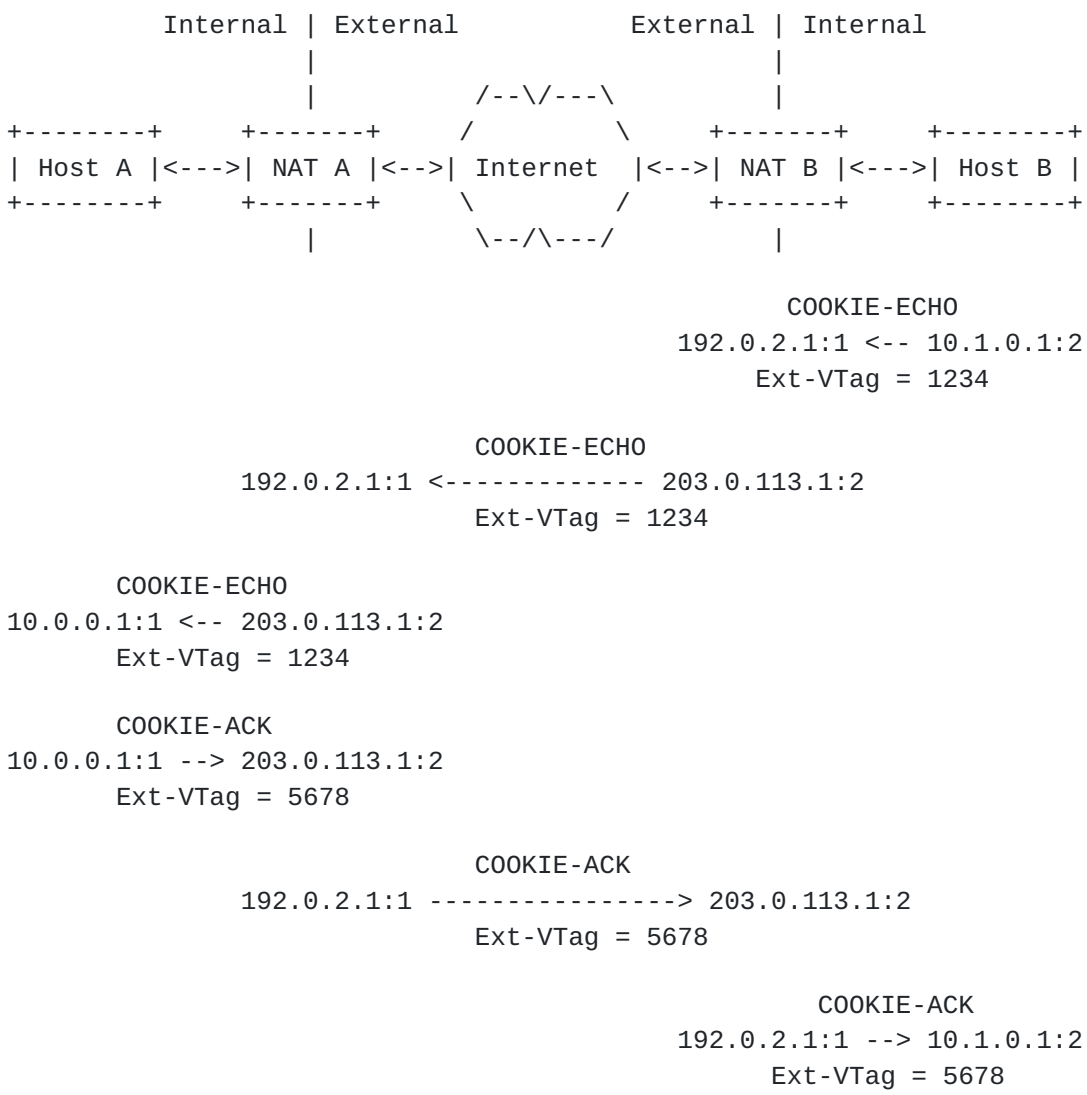
NAT B updates entry:

NAT B

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

INIT-ACK[Initiate-Tag = 1234]
192.0.2.1:1 --> 10.1.0.1:2
Ext-VTag = 5678

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



8. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control NAT friendliness.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by supporting one new read/write socket option.

8.1. Get or Set the NAT Friendliness (SCTP_NAT_FRIENDLY)

This socket option uses the option_level IPPROTO_SCTP and the option_name SCTP_NAT_FRIENDLY. It can be used to enable/disable the NAT friendliness for future associations and retrieve the value for future and specific ones.

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or SCTP_FUTURE_ASSOC for this query. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value: A non-zero value indicates a NAT-friendly mode.

9. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

[NOTE to RFC-Editor:

The requested values for the chunk type and the chunk parameter types are tentative and to be confirmed by IANA.

]

This document (RFCXXXX) is the reference for all registrations described in this section. The requested changes are described below.

9.1. New Chunk Flags for Two Existing Chunk Types

As defined in [RFC6096] two chunk flags have to be assigned by IANA for the ERROR chunk. The requested value for the T bit is 0x01 and for the M bit is 0x02.

This requires an update of the "ERROR Chunk Flags" registry for SCTP:

ERROR Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFCXXXX]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

As defined in [RFC6096] one chunk flag has to be assigned by IANA for the ABORT chunk. The requested value of the M bit is 0x02.

This requires an update of the "ABORT Chunk Flags" registry for SCTP:

ABORT Chunk Flags

Chunk Flag Value	Chunk Flag Name	Reference
0x01	T bit	[RFC4960]
0x02	M bit	[RFCXXXX]
0x04	Unassigned	
0x08	Unassigned	
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

9.2. Three New Error Causes

Three error causes have to be assigned by IANA. It is requested to use the values given below.

This requires three additional lines in the "Error Cause Codes" registry for SCTP:

Error Cause Codes

Value	Cause Code	Reference
176	VTag and Port Number Collision	[RFCXXXX]
177	Missing State	[RFCXXXX]
178	Port Number Collision	[RFCXXXX]

9.3. Two New Chunk Parameter Types

Two chunk parameter types have to be assigned by IANA. It is requested to use the values given below. IANA should assign these values from the pool of parameters with the upper two bits set to '11'.

This requires two additional lines in the "Chunk Parameter Types" registry for SCTP:

Chunk Parameter Types

ID Value	Chunk Parameter Type	Reference
49159	Disable Restart (0xC007)	[RFCXXXX]
49160	VTags (0xC008)	[RFCXXXX]

10. 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.

For SCTP endpoints, this document does not add any additional security considerations to the ones given in [RFC4960], [RFC4895], and [RFC5061]. In particular, SCTP is protected by the verification tags and the usage of [RFC4895] against off-path attackers.

11. Acknowledgments

The authors wish to thank Gorry Fairhurst, Bryan Ford, David Hayes, Alfred Hines, Karen E. E. Nielsen, Henning Peters, Timo Voelker, Dan Wing, and Qiaobing Xie for their invaluable comments.

In addition, the authors wish to thank David Hayes, Jason But, and Grenville Armitage, the authors of [DOI_10.1145_1496091.1496095], for their suggestions.

12. References

12.1. Normative References

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

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <https://www.rfc-editor.org/info/rfc4895>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <https://www.rfc-editor.org/info/rfc6096>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[DOI_10.1145_1496091.1496095] Hayes, D., But, J., and G. Armitage, "Issues with network address translation for SCTP", ACM SIGCOMM Computer Communication Review Vol. 39, pp. 23, DOI 10.1145/1496091.1496095, December 2008.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
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