Workgroup: Network Working Group
Internet-Draft:
draft-porfiri-tsvwg-sctp-natsupp-03
Published: 29 April 2022
Intended Status: Standards Track
Expires: 31 October 2022
Authors: C. Porfiri
Ericsson AB

Stream Control Transmission Protocol (SCTP) Network Address Translation Support

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 functions for TCP that allows multiple hosts to reside behind a NAT function and yet share a single IPv4 address, even when two hosts (behind a NAT function) 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 needed for the SCTP endpoints and the mechanisms for NAT functions necessary to provide similar features of NAPT in the single point and multipoint 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 31 October 2022.

Copyright Notice

Table of Contents

1. Introduction

Stream Control Transmission Protocol (SCTP) [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 functions for TCP that allows multiple hosts to reside behind a NAT function using private-use addresses (see [RFC6890]) and yet share a single IPv4 address, even when two hosts (behind a NAT function) 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 function maps a single or multiple internal addresses to a single external address and vice versa.

To date, specialized code for SCTP has not yet been added to most NAT functions 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 function and this host can only be single-homed. The only alternative for supporting legacy NAT functions is to use UDP encapsulation as specified in [RFC6951] .

The NAT function in the document refers to NAPT functions described in Section 2.2 of [RFC3022] , NAT64 [RFC6146] , or DS-Lite AFTR [RFC6333] .

This document specifies procedures allowing a NAT function to support SCTP by providing similar features to those provided by a NAPT for TCP (see [RFC5382] and [RFC7857]), UDP (see [RFC4787] and [RFC7857]), and ICMP (see [RFC5508] and [RFC7857]). This 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 function will maintain the appropriate state without the NAT function 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 function's external IP address in the same way that a TCP session can use a NAT function.

- If a NAT function 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 used by SCTP.

- Not having to touch the IP payload makes the processing of ICMP messages by NAT functions easier.

An SCTP-aware NAT function will need to follow these procedures for generating appropriate SCTP packet formats, this is needed under circumstances detailed in this document and only triggered by the detection of an SCTP packet containing an INIT chunk.

When considering SCTP-aware NAT it is possible to have multiple levels of support. At each level, the Internal Host, Remote Host, and NAT function does or does not support the procedures described in this document.

The reference configuration for NAT support is depicted in the following figure:

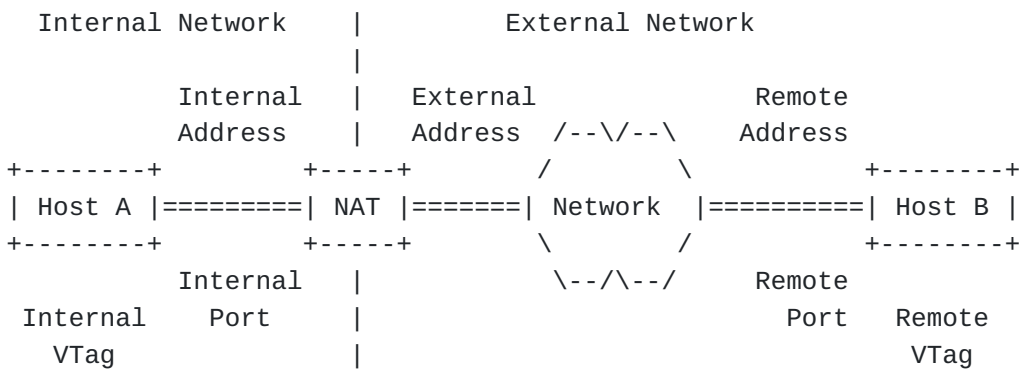


Figure 1: Basic Network Setup

In the above Figure 1 the NAT hides Host A whereas Host B is directly connected to the public internet. Host A has a private IP address, NAT and Host B have public IP addresses.

The following table illustrates the results of the various combinations of support and if communications can occur between two

endpoints with reference to Figure 1, the NAT adaptation is the one described in the current document.

Internal Host	NAT Function	Remote Host	Communication
Support	Support	Support	Yes
Support	Support	No Support	Yes
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 no communication can occur when a NAT function does not support SCTP-aware NAT. This assumes that the NAT function does not handle SCTP packets at all and all SCTP packets sent from behind a NAT function are discarded by the NAT function.

In some cases, where the NAT function supports SCTP-aware NAT but the local host does not support the feature, communication can possibly occur in a limited way. For example, only one host can have a connection when a collision case occurs.

When a SCTP host is deployed behind a NAT and both support SCTP-aware NAT, the communication will suceed independently from the remote peer.

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.

Internal-Address (Int-Addr)
An internal address that is known to the internal host.

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

Internal-VTag (Int-VTag)
The SCTP Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the internal host has chosen for an association. The VTag is a unique 32-bit tag that accompanies any incoming SCTP packet for this association to the Internal-Address.

Remote-Address (Rem-Addr)
The address that an internal host is attempting to contact.

Remote-Port (Rem-Port)
The port number used by the host holding the Remote-Address.

Remote-VTag (Rem-VTag)
The Verification Tag (VTag) (see Section 3.1 of [RFC4960]) that the host holding the Remote-Address has chosen for an association. The VTag is a unique 32-bit tag that accompanies any outgoing SCTP packet for this association to the Remote-Address.

External-Address (Ext-Addr)
An external address assigned to the NAT function, that it uses as a source address when sending packets towards a Remote-Address.

4. Motivation and Overview

4.1. SCTP NAT Traversal Scenarios

This section defines the notion of single and multipoint NAT traversal.

4.1.1. Single Point Traversal

In this case, all packets in the SCTP association go through a single NAT function, as shown in Figure 2 .

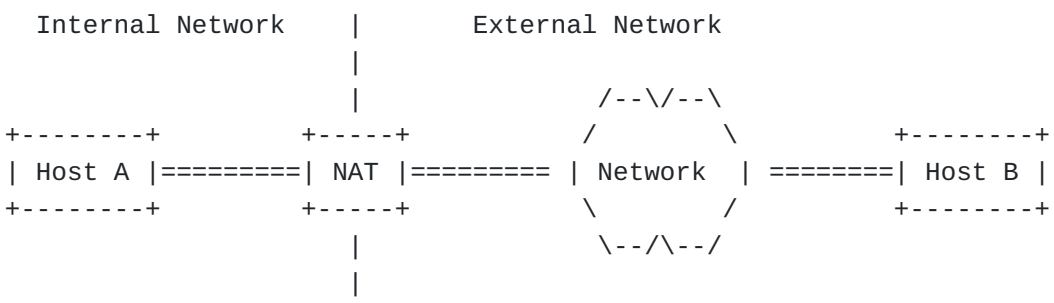


Figure 2: Single NAT Function Scenario

A variation of this case is shown in Figure 3 , i.e., multiple NAT functions in the forwarding path between two endpoints.

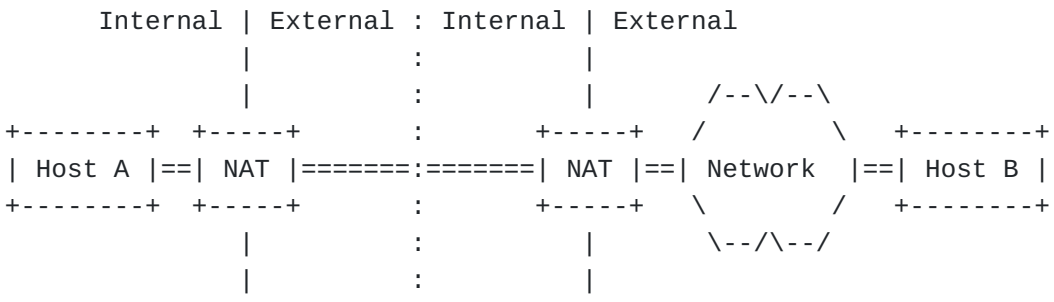


Figure 3: Serial NAT Functions Scenario

Another case where the Endpoint is ditributed among SCTP Hosts is shown in Figure 4 where multiple Hosts behave as Server and share the same Internal Port. A Load Balancer node supports NAT when a new Association request comes. The description of the Load Balancer function and its interwork with NAT function is out of the scope of this document.

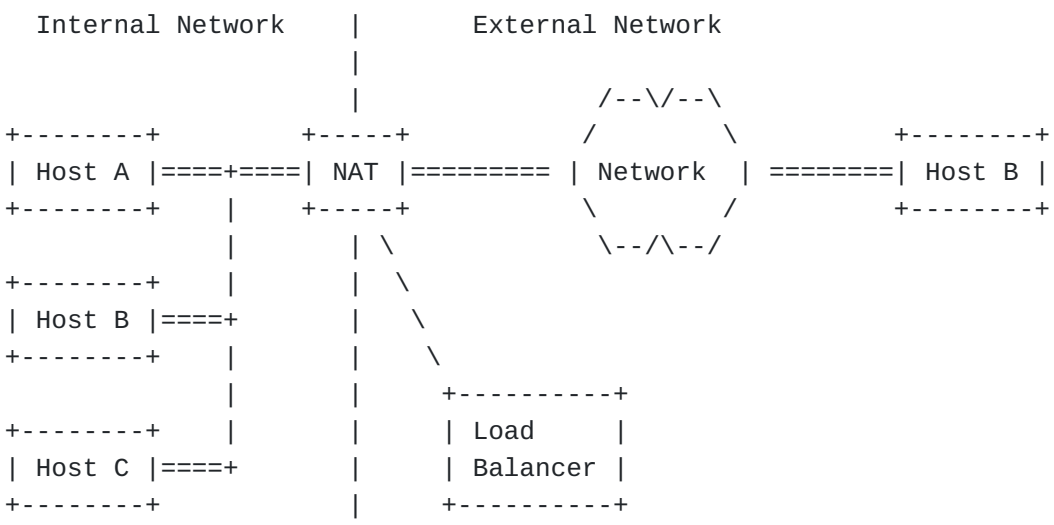


Figure 4: Distributed Endpoint Scenario

Although one of the main benefits of SCTP multi-homing is redundant paths, in the single point traversal scenario the NAT function represents a single point of failure in the path of the SCTP multi-homed association. However, the rest of the path can 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 function in this case sees all the packets of the SCTP association.

4.1.2. Multipoint Traversal

This case involves multiple NAT functions and each NAT function only sees some of the packets in the SCTP association. An example is shown in Figure 5 .

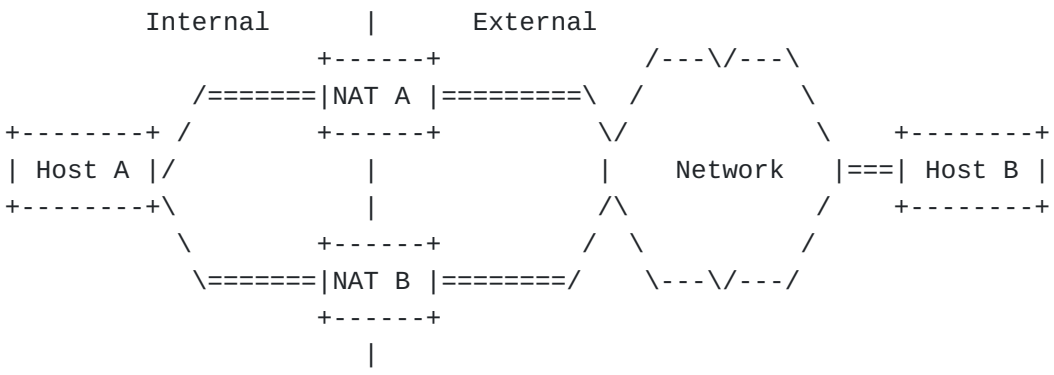


Figure 5: Parallel NAT Functions Scenario

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 possibly results in changing one of the SCTP port numbers during the processing, which requires the recomputation of the transport layer checksum by the NAPT function. 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, even though hardware support can mitigate this in some implementations.

An SCTP endpoint can have multiple addresses but only has a single port number to use. To make multipoint traversal work, all the NAT functions involved need to 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 port numbers and addresses configured within each NAT function. Other mechanisms could make use of NAT to NAT communication. Such mechanisms have not been deployed on a wide scale base and thus are not a preferred solution. Therefore an SCTP variant of NAT function has been developed and is described in draft-ietf-tsvwg-natsupp-23 that is the version at the current time. This document describes an alternative to that function exploiting most of the same principles. Rather than being radically different, it can be seen as a subset with some limitations but less complex and requiring minor computational effort at the SCTP Endpoints and at the NAT functions (see Section 4.3).

4.3. The SCTP-Specific Variant of NAT

In this section it is allowed that there are multiple SCTP capable hosts behind a NAT function that share one External-Address. This section focuses on the single point traversal scenario (see Section 4.1.1) as well as on the multipoint trasversal NAT (see Section 4.1.2).

The modification of outgoing SCTP packets sent from an internal host is simple: the source address of the packets has to be replaced with the External-Address. It might also be necessary to establish some state in the NAT function to later handle incoming packets.

Typically, the NAT function has to maintain a NAT binding table of Internal-Port, Remote-Port, Internal-Address, Remote-Address. 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. Create() instantiates a supervision timer on the NAT-State control block that has duration greather than 2 * HB.interval and lower than 4 * HB.interval (see section 15 of [RFC4960]). A NAT function MAY allow creating NAT-State control blocks via a management interface.

For SCTP packets coming from the external realm of the NAT function the destination address of the packets has to be replaced with the Internal-Address of the host to which the packet has to be delivered, if a NAT state entry is found. The lookup of the Internal-Address is based on the Remote-Address, Remote-Port and the Internal-Port. The lookup function retarts the Nat-State control block supervision timer.

The entries in the NAT binding table need to fulfill some uniqueness conditions. There can not be more than one entry NAT binding table with the same 4-tuple of Internal-Address, Remote-Address, Internal-Port and Remote-Port.

NAT is able understanding that the SCTP packet transports an INIT chunk because the SCTP common header will have VTAG=0 (see section 3.1 of [RFC4960])

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

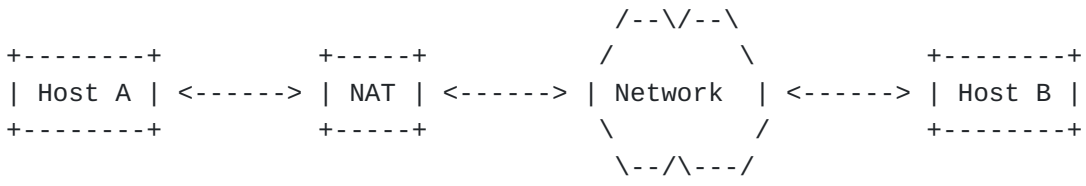

```
            INIT[Initiate-Tag]
 Int-Addr:Int-Port ------> Rem-Addr:Rem-Port
                 Rem-VTag=0

        if lookup(Int-Port, Rem-Port, Rem-Addr) == true
           if lookup(Int-Addr, Int-Port, Rem-Port, Rem-Addr) == false
               sendAbort(Rem-Addr, Rem-Port, Int-Addr, Int-Port, M-bit)
           else
               Returns(control block)
               forwardPkt(Ext-Addr, Int-Port, Rem-Addr, Rem-Port)
        else
               Create(Int-Port, Rem-Port, Int-Addr, Rem-Addr)
               Returns(control block)
               forwardPkt(Ext-Addr, Int-Port, Rem-Addr, Rem-Port)


          Translates To:

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

In the normal case a NAT binding table entry will be created.

However, it is possible that there is already a NAT binding table entry with the same Remote-Address, Internal-Port and Remote-Port but different Internal-Address. In this case the packet containing the INIT chunk MUST be dropped by the NAT and a packet containing an ABORT chunk SHOULD be sent to the SCTP host that originated the packet with the M bit set and 'Port Number Collision' 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.

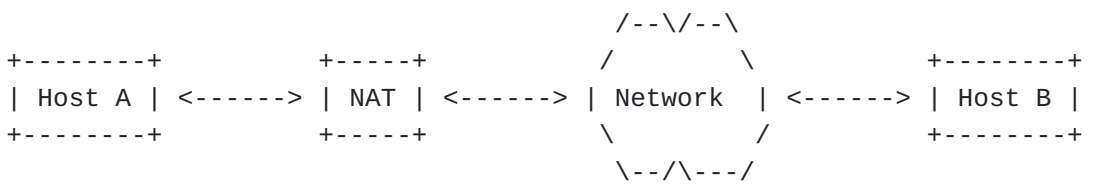
In case that there's already a a NAT binding table entry with the same Remote-Address, Internal-Port, Remote-Port and the same Internal-Address, meaning that the INIT chunk is a new attempt for the same Association, the NAT entry is reused.

The processing of outgoing SCTP packets containing chunks other than INIT is described in the following figure.




```
Int-Addr:Int-Port ------> Rem-Addr:Rem-Port
                  Rem-VTag

              if lookup(Int-Port, Rem-Port, Rem-Addr) == false
                     Create(Int-Port, Rem-Port, Int-Addr, Rem-Addr)
                     Returns(control block)
              forwardPkt(Ext-Addr, Int-Port, Rem-Addr, Rem-Port)

              Translates To:

                        Ext-Addr:Int-Port ------> Rem-Addr:Rem-Port
                                          Rem-VTag
```

The processing of incoming SCTP packets containing an INIT chunk is illustrated in the following figure. This scenario is valid for all message flows in this section.




```
                                  INIT [Initiate-Tag]
                        Ext-Addr:Int-Port <---- Rem-Addr:Rem-Port
                                        Int-VTag=0
```

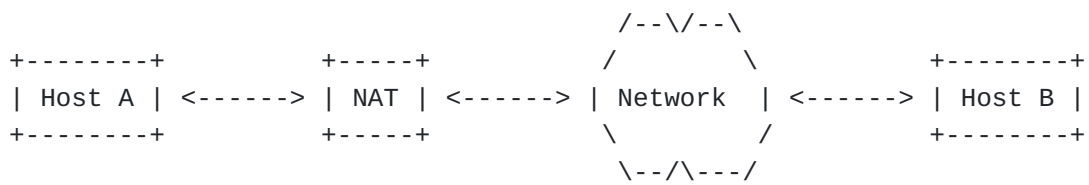
```
if lookup(Int-Port, Rem-Port, Rem-Addr) == true
        Returns(control block)
        forwardPkt(Rem-Addr, Rem-Port, Int-Addr, Int-Port)
else
    if INIT contains RJ option
        send INIT-ACK to the INIT source
    else
        Create(Int-Port, Rem-Port, Int-Addr, Rem-Addr)
        Returns(control block)
        forwardPkt(Rem-Addr, Rem-Port, Int-Addr, Int-Port)
```

Translates To:

```
            INIT[Initiate-Tag]
Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
                Int-VTag=0
```

When INIT chunk contains the RJ option set, it's a duplicate of the INIT used for establishing the association. In such case the reason for RJ option is to be recognized by the NAT function that will reply to the sender instead of the SCTP Host. This allows the SCTP Endpoint to be distributed among hosts, and since the NAT function cannot arbitraly choose among hosts, it takes the role of the unknown host in answering to the INIT issuer so that it can proceed with the ASCONF handshake and extend the association. The final step of setting the path between the NAT function and the unknown host will be completed by the host receiving ASCONF and sending an INIT with RJ option towards the remote peer.

The processing of incoming SCTP packets containing chunk different than INIT is illustrated in the following figure. The Lookup() function has as input the Remote-Address, Remote-Port and the Internal-Port. It returns the corresponding entry of the NAT binding table.



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

```
if lookup(Int-Port, Rem-Port, Rem-Addr) == true
   Returns(NAT-State control block containing Int-Addr)
   forwardPkt(Ext-Addr, Int-Port, Rem-Addr, Rem-Port)
```

Translates To:

Int-Addr:Int-Port <------ Rem-Addr:Rem-Port
Int-VTag

In the case where the Lookup function fails because it does not find an entry, the SCTP packet is dropped.

4.4. Compatibility and increamental deployment

The current proposal for adding SCTP-capable NAT function is meant to provide backwards compatibility in both involved functionality and being compatible with legacy SCTP remote terminations that doesn't implement it.

The compatibility at NAT tracking mechanism allows the NAT functionto be able hiding also SCTP stack that doesn't implement the current specfication, at the same time an SCTP stack implementing the current specification canbe deployed in a NAT scenario where the NAT doesn't implement it. In either cases the SCTP termination will be accomplished with limitations as described earlier.

The compatibility at network level is proposed in a way that makes it possible deploying a cluster of SCTP termination behind a NAT function still with full compatibility towards legacy networking. As an example, the scenario described in Figure 2 shows Host A being hidden by NAT and Host B being directly connected to the internet. In such case only Host A and NAT need to implement the current specification whilst Host B can neglect it. The same applies to more complex scenarios such as the ones shown in Figure 4 or in Figure 5.

4.5. Differences with Current NAT Support Draft

This section describes the differences with the existing draft-ietf-tsvwg-natsupp.

From a functional perspective, the major difference between is in the compatibility towards legacy SCTP hosts. The NAT-adaptation specified in this document allows interoperability between SCTP hosts even when the remote peer hasn't implemented it. Not even is mandatory that all the NAT devices in the path do implement it as long as they allow SCTP packets to pass through transparently. On the existing draft-ietf-tsvwg-natsupp, the specification needs to be implemented on all SCTP Hosts and all NAT devices in the network in order to work.

The main technical difference is that the NAT function is simpler and doesn't require explicit handling of NAT missing states. Actually in this proposal NAT doesn't need to parse all the SCTP payloads. NAT only parses INIT chunks, filtering of SCTP packets containing INIT chunks is based on checking the SCTP Common Header and discriminate the behavior based on Verification Tag = 0, that indicates the SCTP packet contains an INIT chunk. The NAT supervises the association by means of a timer, if no SCTP packets are seen within a certain time, NAT assumes that the association is closed and will remove the related NAT-entry.

The other difference is in the role of the SCTP User. In the current proposal it's up to the SCTP User to change the originating Endpoint (i.e. choose a different port number) if collision is detected. The current proposal guarantees that at each node being in a path belonging to an association, there will be only one 4-uple describing that association, that means the NAT doesn't need to take care of VTAG.

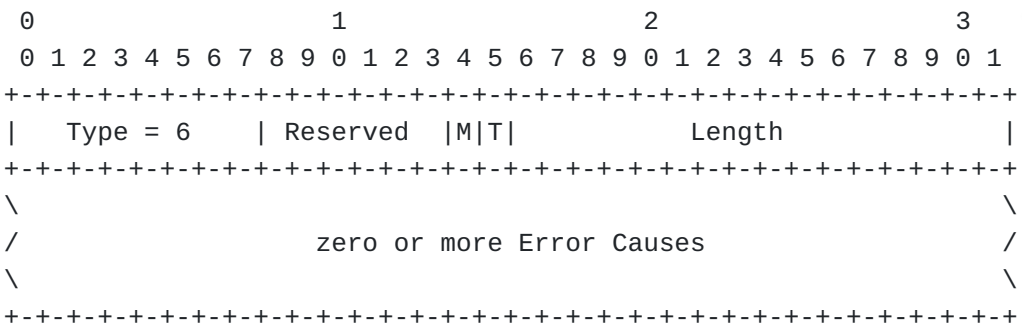
5. Data Formats

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 functions and received by SCTP endpoints. Section 5.3 describes parameters sent by SCTP endpoints and used by NAT functions and SCTP endpoints.

5.1. Modified Chunks

This section presents existing chunks defined in [RFC4960] for which additional flags are specified by this document.

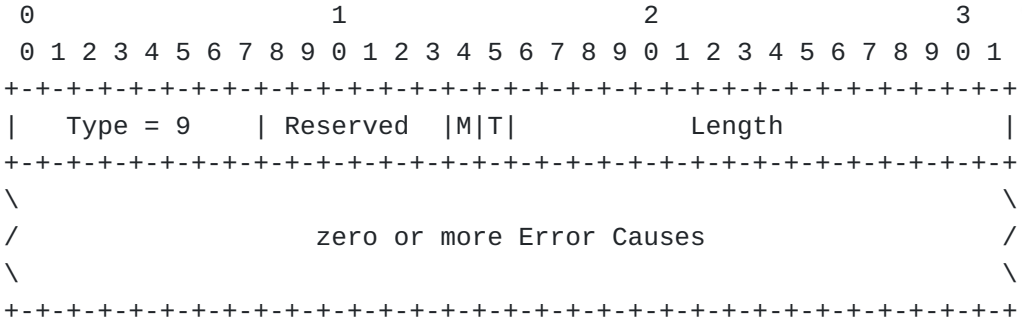
5.1.1. Extended ABORT Chunk



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 (e.g., NAT).

[NOTE to RFC-Editor: Assignment of M bit to be confirmed by IANA.]

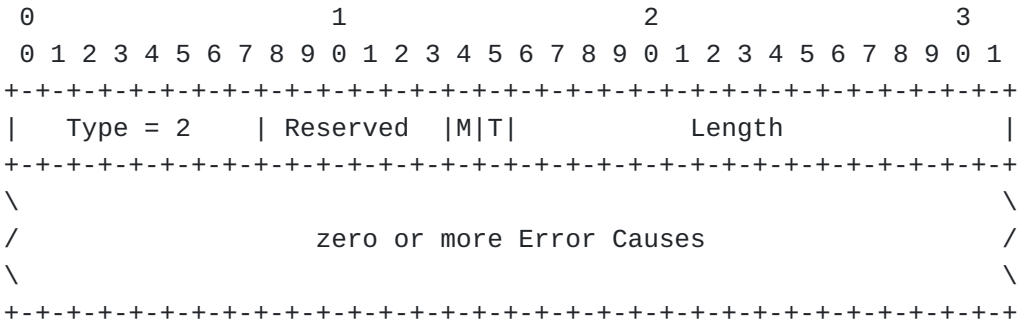
5.1.2. Extended ERROR Chunk



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.]

5.1.3. Extended INIT-ACK Chunk



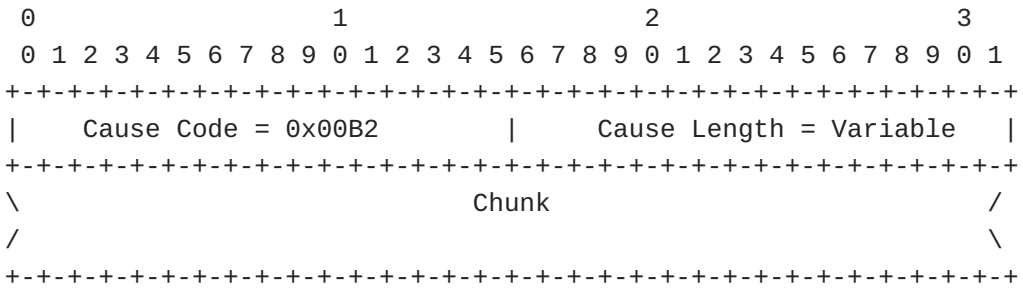
The INIT ACK chunk defined in [RFC4960] is extended to add the new 'M bit'. The M bit indicates to the receiver of the INIT-ACK chunk that the chunk was not generated by the peer SCTP endpoint, but instead by a middle box.

[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. 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.]

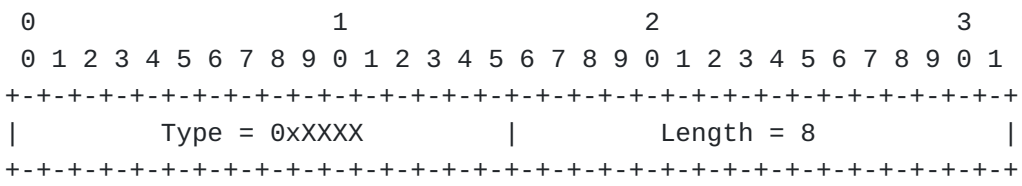
5.3. New Parameters

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

5.3.1. Repetita Juvant Parameter

Repetita Juvant is a latin phase standing for "repeating does good". It's sually said as a jocular remark to defend the speaker's (or writer's) choice to repeat some important piece of information to ensure reception by the audience.

The RJ Parameter is used as Optional Parameter in the INIT chunk. The RJ parameter is used to indicate that INIT chunk is the repetition of an already sent one even if it comes from a different source address. It's used from either peers before sending ASCONF in order to setup the NATs in the path.



6. Procedures for SCTP Endpoints and NAT Functions

If an SCTP endpoint is behind an SCTP-aware NAT, a number of problems can arise as it tries to communicate with its peers:

- IP addresses can not be included in the SCTP packet. This is discussed in Section 6.1 .
- More than one host behind a NAT function could select the same source port number when initiating an association with the same peer server. This creates a situation where the NAT function will not be able to forward the INIT chunk. This situation is discussed in Section 6.3 .
- A restart of a NAT function during a conversation could cause a loss of its state. This problem and its solution is discussed in Section 6.4 .
- NAT functions need to deal with SCTP packets being fragmented at the IP layer. This is discussed in Section 6.5 .
- An SCTP endpoint can be behind two NAT functions in parallel providing redundancy. The method to set up this scenario is discussed in Section 6.6 .

The mechanisms to solve these problems require additional chunks and parameters, defined in this document, and modified handling procedures from those specified in [RFC4960] as described below.

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 does not work when NAT functions are present.

Every association setup from a host behind a NAT function MUST NOT use multiple internal addresses. The INIT chunk MUST NOT contain an IPv4 Address parameter, IPv6 Address parameter, or Supported Address Types parameter. The INIT ACK chunk MUST NOT contain any IPv4 Address parameter or IPv6 Address parameter using non-global addresses. The INIT chunk and the INIT ACK chunk MUST NOT contain any Host Name parameters.

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

6.2. Association Setup Considerations for NAT

When Endpoint is Distributed, NAT needs the cooperation of a Load Balancer function for handling incoming and outgoing Association Requests. It's up to the Load Balancer internal design the strategy for permitting a Distributed Endpoint to handle the traffic. Functionally, it's important that Load Balancer provides NAT a way for assigning Associations to multiple SCTP Hosts.

6.3. Handling of Internal Port Number Collisions

Consider the case where two hosts in the Internal-Address space want to set up an SCTP association with the same service provided by some remote host. This means that the Remote-Port is the same. If they both choose the same Internal-Port the NAT function will experience collision when receiving the INIT and trying to create an Entry in the NAT Tables. In such case NAT will send an ABORT chunk with M-bit set to the SCTP Client. Since it's up to the SCTP User Application to choose the Internal Port, it may be that an Association chooses the Internal Port from the ephemeral port range at random (see [RFC6056]), this would make the probability for Port Number Collision low.

At the Association initialization, the Client will experience one out of three alternative answers from the network:

* INIT-ACK from the peer, this means a viable path exists between peers, all the involved NATs have NAT tables properly configured and the Association can be established.

* ABORT with M-bit set from one of the NATs within the path, this means that the Association cannot be established. The SCTP User

application SHOULD decide whether to retry with a different Internal Port or to give up. The way SCTP and the SCTP User interact in this case is implementation dependent.

*ABORT from the remote peer.

The way SCTP and SCTP User Application interact can be either:

*An application can request a specific local port number (in the socket API, using bind() with a non-zero port number) and in case of a local port number collision, the connection setup has to fail. It is up to the application to close() the socket and restart from the beginning.

*An application leaves the local port number selection up to the SCTP stack (in the socket socket API by either calling bind() with a zero port number or not calling bind() at all before calling connect() or sendto()). However, once the port number is chosen, it can not be changed. So in case of a local port number collision, the association setup has to fail. It is up to the application to close() the socket and restart from the beginning.

*An application leaves the local port number selection up to the SCTP stack (in the socket socket API by either calling bind() with a zero port number or not calling bind() at all before calling connect() or sendto()). In addition, it indicates that the SCTP can change the local port number over time (in the socket API this would be calling an IPPROTO_SCTP level new socket option). In this case, the SCTP stack can automatically retry a connection setup in case of an local port number collision.

6.3.1. NAT Function Considerations

NAT function checks for collision only on packets containing INIT chunk. If the NAT function detects a collision of internal port numbers, it SHOULD send a packet containing an ABORT chunk with the M bit set. 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). the source and destination address and port numbers MUST be swapped.

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

If the INIT chunk contains the RJ option the NAT function MUST NOT forward the INIT chunk to the SCTP Host but it MUST reply to the remote peer with INIT-ACK chunk with the M bit set. 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 (see Section 5.1.3). The

information contained in INIT-ACK chunk SHOULD be copied from the INIT chunk. The value for Initiate Tag and Initial TSN MAY be chosen random.

6.3.2. Endpoint Considerations

The sender of the packet containing the INIT chunk upon reception of a packet containing an ABORT chunk with M bit set and the appropriate error cause code for colliding NAT binding table state is included, SHOULD evaluate the reason for ABORT. If the reason is "Port Number Collision" it SHOULD reinitiate the association setup procedure after choosing a new Internal Port.

6.4. Handling of Missing State

6.4.1. NAT Function Considerations

When experiencing a restart, the NAT function will start handling SCTP packets with time difference between the ones containing INIT chunks and all the other ones. Handling of SCTP packets containing INIT chunks will start at least 4 * HB.interval after handling other SCTP packets (see section 15 of [RFC4960]). This avoids race condition between the recreation of existing Entries in the NAT Table and the creation of new ones from new Association requests.

If the NAT function receives a packet not containing an INIT chunk from the internal network for which the lookup procedure does not find an entry in the NAT binding table, it must create an Entry for that packet and forward it. If the NAT function receives a packet not containing an INIT chunk from the external network for which the lookup procedure does not find an entry in the NAT binding table, it must silently drop it.

6.4.2. Endpoint Considerations

Upon restart of a NAT function, the endpoint will experience connectivity interruption, depending on the Association state it will keep on retrying sending SCTP packets containint DATA chunks or HB chunks. Since the longest interval between SCTP packets is HB.interval, it will be able restoring the connectivity at most 2 * HB.interval after NAT function is back at work.

If the Endpoint is trying to establish an Association, it will experience a longer connectivity unavalilability of more than 4 * HB.interval as NAT needs to rebuild the NAT Table with the existing Associations first.

6.5. Handling of Fragmented SCTP Packets by NAT Functions

SCTP minimizes the use of IP-level fragmentation. However, it can happen that using IP-level fragmentation is needed to continue an SCTP association. For example, if the path MTU is reduced and there are still some DATA chunk in flight, which require packets larger than the new path MTU. If IP-level fragmentation can not be used, the SCTP association will be terminated in a non-graceful way. See [RFC8900] for more information about IP fragmentation.

Therefore, a NAT function MUST be able to handle IP-level fragmented SCTP packets. The fragments MAY arrive in any order.

When an SCTP packet can not be forwarded by the NAT function due to MTU issues and the IP header forbids fragmentation, the NAT MUST send back a "Fragmentation needed and DF set" ICMPv4 or PTB ICMPv6 message to the internal host. This allows for a faster recovery from this packet drop.

6.6. Multipoint Traversal Considerations for Endpoints

If a multi-homed SCTP endpoint behind a NAT function connects to a peer, it MUST first set up the association single-homed with only one destination address causing the first NAT function to populate its state.

Once an Association has been created, it's possible to add further external IP addresses for the peer to use, but before adding each IP address it must be created the needed set of Entries in all NAT functions towards all the peer's IP addresses. An INIT chunk containing a RJ option (see Section 5.3.1) SHOULD be sent towards all peers IP addresses using a path selector that is expected to result in another external addres than association creation. The reason why an INIT chunk with RJ option set is to be used is for permitting the remote to be able discriminating between a request for a new Association in case of Distributed Endpoint. The result from that INIT is according to the given rules for Association setup (see Section 6.1) and can cause collision. The reception of INIT ACK confirms that the path from the new IP address and the remote one is available and that all the NATs involved are properly configured. In case INIT ACK has M-bit set, the remote Endpoint is distributed.

After succefull confirmation, the Endpoint SHOULD add each IP address using packets containing ASCONF chunks sent via their respective NAT functions. The address used in the Add IP address parameter is the wildcard address (0.0.0.0 or ::0) and the address parameter in the ASCONF chunk SHOULD also contain the VTags parameter.

When an Endpoint gets a new Remote IP Address added to an Association, it SHOULD send INIT chunks with RJ option towards from all its own IP Addresses towards that address in order to properly set all the NATs in the path.

6.6.1. NAT Function Considerations

NAT function differentiates the behavior towards INIT chunk depending on the RJ option. If the RJ option exists and the packet contains an incoming INIT chunk, the NAT function SHOULD NOT forward the INIT chunk towards the SCTP Host, it shall reply instead with an INIT ACK chunk with the M-bit set. Section 5.1.3). NAT function SHOULD create INIT ACK data by using the parameters from the received INIT chunk.

6.6.2. Endpoint Considerations

When the Endpoint receives an INIT chunk with RJ option set, it will ignore the RJ option and handle INIT as in the legacy case.

The Endpoint originating INIT chunk with RJ option set can receive different answers:

*When receiving INIT ACK, it will assume the NATs on the path are properly set and the Endpoint can continue with the ASCONF procedure.

*When receiving as ABORT with M-bit set, it shall assume that a path is not possible to be established. The Endpoint SHOULD retry after a time greather than 4 * HB.interval.

*When receiving an ABORT without M-bit set, it shall assume that some temporary NAT configuration has led the INIT towards the wrong SCTP Host. The Endpoint SHOULD retry after a time greather than 4 * HB.interval.

6.7. Path Probing considerations

The SCTP protocol relies on continous path probing by means of data sending or using the Heartbeat mechanism as specified in section 5.4 of [RFC4960] The adoption of the NAT mechanisms as described in this document introduces a criticality in the Path Probing mechanism of SCTP.

The problem happens when, due to network problem, one or more secondary paths belonging to an Association will experience timeout in Path probing so than in some of the NAT functions used in the path there's no SCTP traffic for the given Association, causing the NAT entry to be canceled because of supervision timeout.

It is recommended that before sending HEARTBEAT to an UNCONFIRMED address, an INIT chunk with RJ paramter set is sent so that NAT functions in the path can setup entries in the NAT tables properly.

7. Examples of Operation

This section describes examples of Association Establishements using the reference scenario depicted in Figure 6 . Hosts A1 and A2 implement a distributed client towards the same remote Host. Hosts B1 and B2 implement a distributed Endpoint 'B' acting as Server. The Load Balancer functionality is not shown as it doesn't affect SCTP protocol.

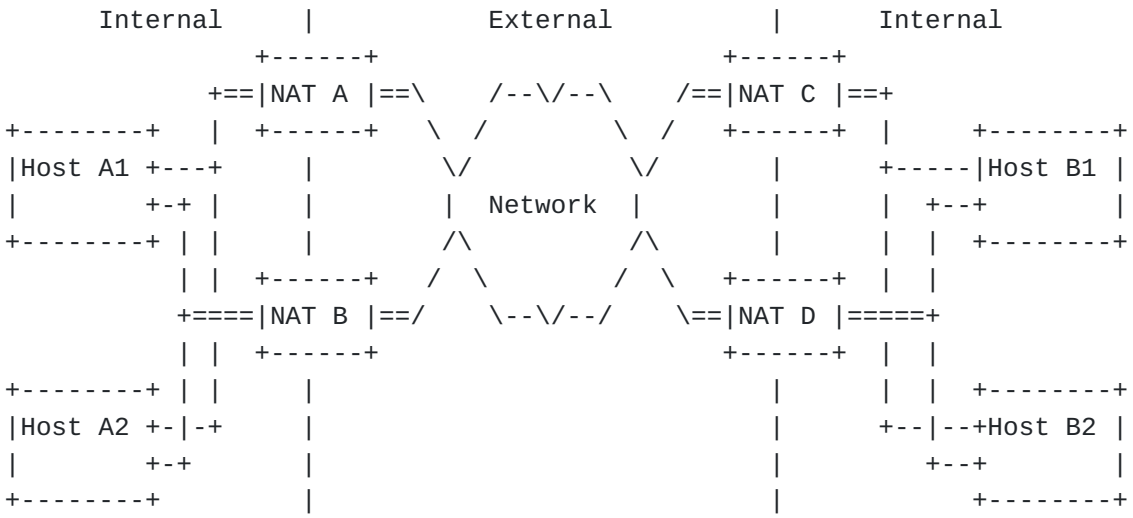


Figure 6: Parallel NAT with distributed endpoints Scenario

7.1. Single Homed Association Setup

This section describes a successfull Association Establishment from A1 towards the distributed endpoint B. The sequence chart is shown in Figure 7 .

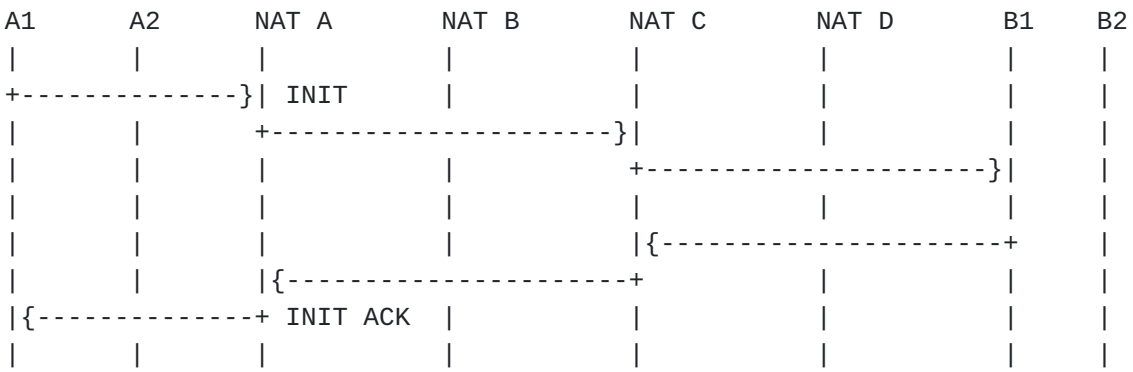


Figure 7: Single Homed successfull Association Setup

7.2. Single Homed Association Setup with Collision

This section describes a successfull Association Establishment from A2 towards the distributed endpoint B. The collision happens at NAT A. The sequence chart is shown in Figure 8 .

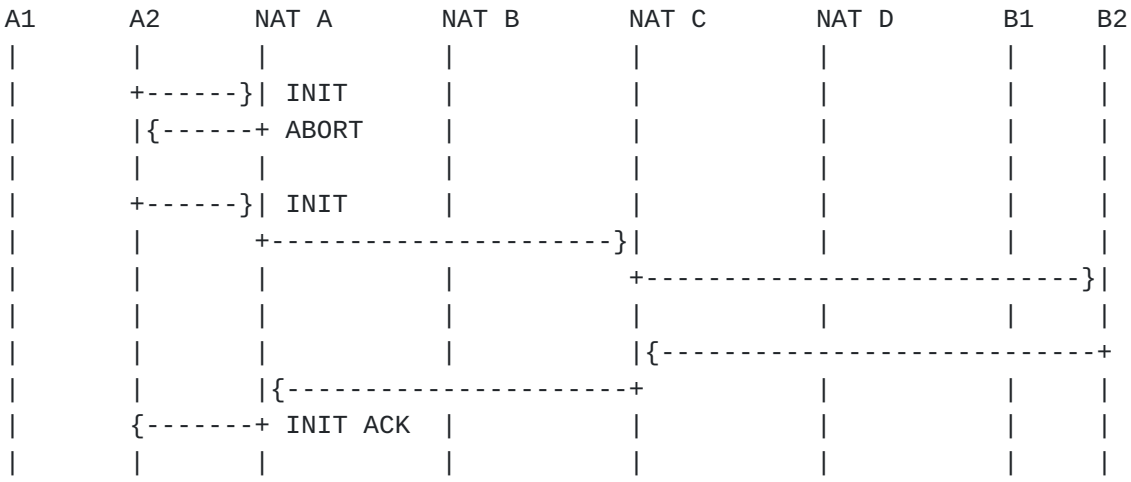


Figure 8: Single Homed successfull Association Setup after congestion

7.3. Multi Homed Association Setup

This section describes how the single homed established at Section 7.1 becomes multihomed. Note that the decision for what peer has to handle the INIT message requires support of Load Balancer. It's assumed that a Load Balancer exists and provides NAT with the right information. Success happens at all steps. Figure 9 .

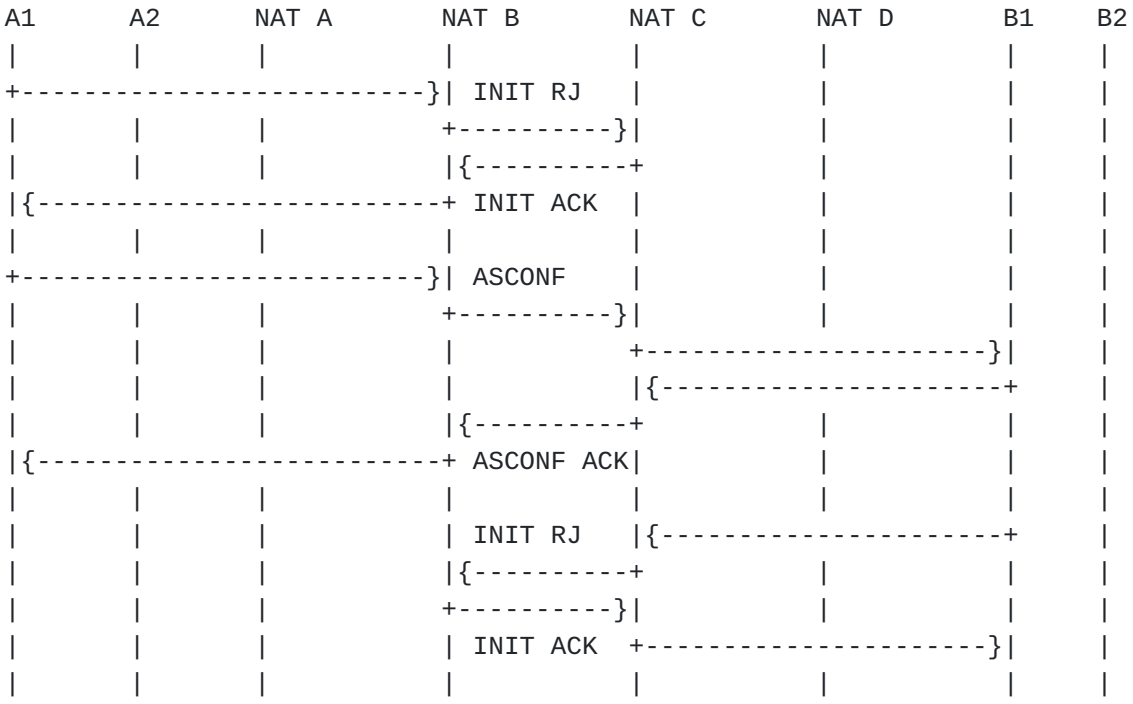


Figure 9: Multi Homed successfull Association Setup

7.4. Multi Homed Association Setup

This section describes how the multihome homed established at Section 7.3 becomes multihomed from the other peer. Success happens at all steps. Figure 10 .

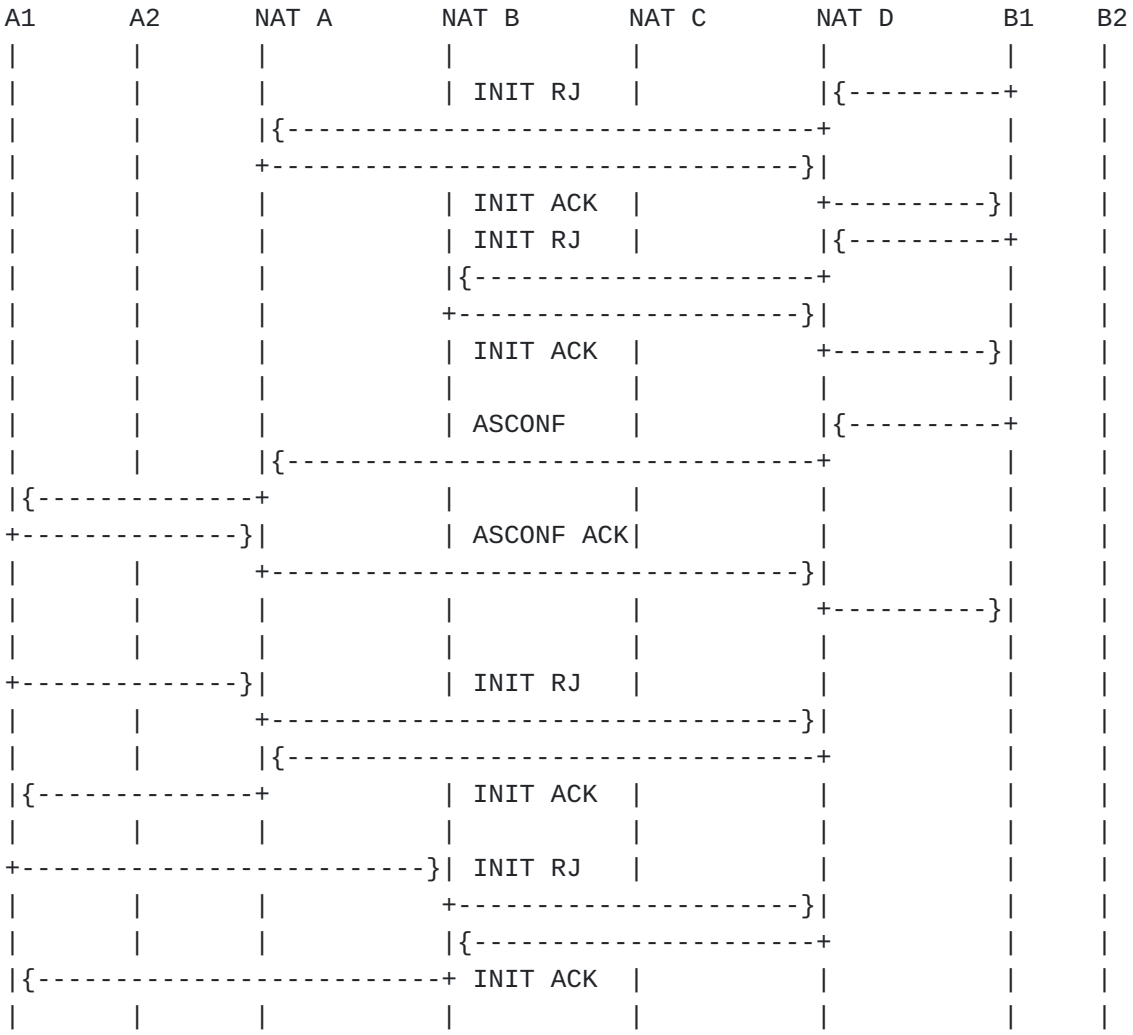


Figure 10: Multi Homed successfull Association Setup

8. 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.

8.1. 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	

Table 2

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	

Table 3

8.2. Four New Error Causes

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

This requires Four 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]

Value	Cause Code	Reference
178	Port Number Collision	[RFCXXXX]
179	VTag Not Found	[RFCXXXX]

Table 4

8.3. Two New Chunk Parameter Types

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

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]

Table 5

9. Security Considerations

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

Generic issues related to address sharing are discussed in [RFC6269] and apply to SCTP as well.

For SCTP endpoints not disabling the restart procedure, this document does not add any additional security considerations to the ones given in [RFC4960] , [RFC4895] , and [RFC5061] .

SCTP endpoints disabling the restart procedure, need to monitor the status of all associations to mitigate resource exhaustion attacks by establishing a lot of associations sharing the same IP addresses and port numbers.

In any case, SCTP is protected by the verification tags and the usage of [RFC4895] against off-path attackers.

For IP-level fragmentation and reassembly related issues see [RFC4963] .

*Setting a low timeout for SCTP mapping entries to cause failures to deliver incoming SCTP packets.

*Instantiating mapping entries to cause NAT collision.

10. 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>.

11. Informative References

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

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI

10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <https://www.rfc-editor.org/info/rfc5382>.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <https://www.rfc-editor.org/info/rfc5508>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <https://www.rfc-editor.org/info/rfc6056>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <https://www.rfc-editor.org/info/rfc6269>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <https://www.rfc-editor.org/info/rfc6333>.

[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>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <https://www.rfc-editor.org/info/rfc7857>.

[RFC8900]
Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

Acknowledgments

The author wishes to thank Michael Tuxen , and Magnus Westerlund for their invaluable comments.

In addition, the author wishes to thank Sriram Yagnaraman , for their suggestions.

The author also wishes to thank the authors of draft-ietf-tsvwg-natsupp-22 which this document is based.

Author's Address

Claudio Porfiri
Ericsson AB
Torshamnsgatan 21
16440 Stockholm
Sweden

Email: claudio.porfiri@ericsson.com