BEHAVE
Internet-Draft
Expires: August 1, 2005

S. Sivakumar
K. Biswas
Cisco Systems
B. Ford
M.I.T

January 28, 2005

NAT Behavioral Requirements for TCP
draft-sivakumar-behave-nat-tcp-req-00

Status of this Memo

Copyright Notice

Abstract

Inconsistent behavior by NATs makes it difficult for the application developers and network administrators to predict the operation of NATs. This document describes the behavior required by NATs when handling TCP traffic. It also specifies the address and port binding

behavioral requirement, timeout aspects and adjusting the sequence numbers and the acknowledgement numbers when changing the payload length.

Table of Contents

1. Introduction 3
2. Scope 3
3. TCP requirements 3
3.1 State Machine 3
3.2 NAT address and port binding 4
3.3 Timeouts 4
3.4 Port reservation. 5
3.5 Processing IP fragments and TCP segments 6
3.5.1 IP Fragments handling 6
3.5.2 TCP Segments handling 7
3.6 Adjusting Sequence Acknowledgement Numbers 8
3.7 Handling of ICMP Error messages 10
4. Security considerations 11
5. IANA Considerations 11
6. IAB Considerations 11
7. Acknowledgements 11
8. References 11
8.1 Normative References 11
8.2 Informative References 11
Authors' Addresses 13
Intellectual Property and Copyright Statements 14

1. Introduction

A lot of issues caused by inconsistent NAT behavior are documented in [UDP-REQ]. Different NAT implementations behave differently when handling the TCP traffic streams. This document defines the required behavior of NATs when handling TCP traffic.

NATs maintain various pieces of session information to translate the TCP streams correctly. NATs would have to run a state machine for TCP to keep track of the state changes. The state machines are predominantly used for controlling the different timers that are maintained by NAT. NATs should also keep track of the payload changes by upper layer ALGs in order to adjust the sequence numbers and acknowledgement numbers properly.

2. Scope

This document will focus specifically on issues that relates to TCP. This document will refer to [UDP-REQ] for all the common NAT behavioral issues and requirements. Application Layer Gateways (ALGs) are out of scope for this document. This document will not propose any solution but will define only the requirements of NAT when handling TCP.

3. TCP requirements

The behavioral requirements of NAT when processing TCP packets are described in this section.

3.1 State Machine

NATs maintain a database of active TCP sessions flowing across the NAT. Each session in the NAT's database has an associated state machine that dynamically tracks the state of the TCP session from the perspective of the NAT. The NAT creates a database entry for a new session, and starts the TCP state machine for that session, when it

forwards the first SYN packet for that session across the NAT. The NAT's TCP state machine transitions from the "active" to the "closed" state when the NAT observes a FIN/FIN ACK sequence, representing graceful shutdown reached cooperatively by both endpoints, or when the NAT observes a RST from either endpoint, representing a non-graceful connection reset forced by one endpoint. Finally, the NAT deletes its database entry for the session some time after the state machine enters the "closed" state, depending on the timeouts described below.

In addition to this basic state information, many NATs also record information about the TCP sequence numbers and the acknowledgment

numbers they observe in the TCP packets flowing across the NAT. If the NAT contains built-in ALGs that can change the payload length of TCP packets, effectively inserting or removing bytes from the TCP stream in one or both directions, then the NAT MUST adjust the sequence numbers in all subsequent packets exchanged in either direction to reflect these inserted or removed bytes.

NATs also use the state machine information associated with a TCP session in order to filter packets arriving from the external realm toward the internal realm. Unless specifically configured to do otherwise, any SYN packets originating from the external realm will be filtered out by the NAT if the NAT's database contains no entry for that session. This behavior reflects the standard firewall policy of rejecting all "unsolicited incoming connection attempts" by default. NATs that implement a state machine for keeping track of the TCP streams are referred to as State Aware NAT (abbreviated SM = YES).

A NAT SHOULD be SM=YES.

3.2 NAT address and port binding

The address and port binding requirements remain the same as the requirements described in [UDP-REQ].

3.3 Timeouts

NATs maintain different types of timeouts for the TCP sessions. These timeouts apply to the different states of the TCP state

machine.

* The NAT's SYN timer has a relatively short timeout, and helps to protect the NAT (and, potentially, the hosts behind the NAT) from SYN flood attacks. The SYN timeout starts when the NAT observes the first SYN on a new session, and is cancelled when the NAT receives an ACK for that SYN from the opposite endpoint, indicating that legitimate two-way communication is taking place.

* The NAT's session timer is a relatively long timeout, and ensures that the NAT is eventually able to delete database entries for formerly-active TCP sessions on which both endpoints have silently ceased communication without either closing or resetting the connection. The NAT's session timer starts when the TCP session enters the active, "fully-open" state (typically at the same time its SYN timer is cancelled), and the session timer MUST be reset whenever the NAT observes an outbound packet from the internal realm to the external realm. Inbound packets SHOULD NOT cause the NAT to reset its session timer.

When the NAT's session timer expires, it SHOULD NOT immediately delete its database entry for the session, since TCP sessions are legitimately allowed to go idle for arbitrary lengths of time with no exchanged traffic. Instead, the NAT SHOULD enter a special "probe" state, in which it sends TCP keep-alive packets to the internal endpoint, using the technique described in section 4.2.3.6 of [HOST], in order to detect whether the internal endpoint still considers the connection to be open. If the NAT receives an ACK or other traffic from the internal endpoint, it resets the session timer and re-enters the "active" state. If the NAT receives a RST from the internal endpoint, it enters the "closed" state and starts the close timer as described below. If the NAT receives no response from the internal endpoint after sending several keep-alive packets, the NAT assumes that the internal endpoint is dead and again enters the "closed" state.

* The NAT's close timer is a relatively short timeout that ensures that the ACKs for the final FINs on a gracefully-closed TCP session have a chance to propagate in both directions, and also to allow time for either endpoint to re-open a recently closed or reset TCP session if desired. The NAT starts the close timer after it observes a FIN packet in each direction, or after it observes an RST from either

endpoint. If a new SYN packet arrives from either endpoint before the close timer expires, the NAT re-enters the active, "half-open" state & re-starts the SYN timer as described above. Otherwise, once the close timer expires the NAT is free to delete its database entry and release all resources allocated to the session.

The following requirements apply to the NAT's timeouts:

A NAT MUST have a SYN timer so that the box is not prone to SYN attacks. The SYN timeout value MUST be configurable, and SHOULD default to at least 30 seconds and no more than 60 seconds.

A NAT MUST have a session timeout. The NAT's session timeout MUST be configurable. The session timeout MUST by default be at least 60 minutes if the NAT uses TCP keep-alives to probe the session after the session timeout expires, and the session timeout MUST by default be atleast 120 minutes if the NAT just silently deletes the database entry for the session after the session timeout expires.

A NAT MUST have a close timeout. NAT MAY have separate timeouts for session close due to FIN versus RST. NAT's close timeouts MUST be at least 2xMSL, or 60 seconds.

3.4 Port reservation.

The TCP port space associated with a NAT's own IP addresses,

particularly its IP addresses on the external network side, are commonly shared between two separate functions:

1. TCP endpoints for address and port translated sessions

2. TCP endpoints for applications residing on the NAT device

The first function results in port reservation for address-translated client sessions. The second function results in port reservation due to applications running on the NAT device itself. Examples of such applications include: a web-based externally accessible management interface, or a port forwarding application that has a predefined binding. Many NATs are actually general-purpose network hosts that also run ordinary TCP-based applications as well. Even dedicated NAT middleboxes often provide remote management functionality, requiring

communication between external hosts and applications on the NAT itself. The resulting sharing of the NAT's TCP port space between address-translation sessions and local applications creates a potential for resource contention.

NATs MUST NOT use a single TCP port for both address-translated sessions and local application sessions at the same time. NATs MAY dynamically re-assign a TCP port from one function to the other on demand, but only after any previous TCP sessions involving that port have become inactive.

3.5 Processing IP fragments and TCP segments

This section describes how a NAT is recommended to behave with regards to handling IP fragments and TCP segments. There are 2 scenarios which can cause fragmentation/segmentation:

1. The MTU of the link can be as such as to cause the IP packets to be fragmented

2. The TCP stack at the either end-point can have the TCP MSS set to value lower than the application payload size which will cause the packets to be TCP segmented.

3.5.1 IP Fragments handling

A NAT may receive a fragmented TCP packet. The following section is provided from [UDP-REQ] for completeness.

The IP packet containing the TCP header could arrive first or last depending of various conditions. NAT that is capable only of receiving TCP fragments in order (that is, with the TCP header in the first packet) and forwarding each of the fragments to the internal

host is described as Received Fragments Ordered (abbreviated RF=O).

NAT that is capable of receiving TCP fragments in or out of order and forwarding the individual packets (or a reassembled packet) to the internal host is referred to as Receive Fragments Out of Order (abbreviated RF=OO).

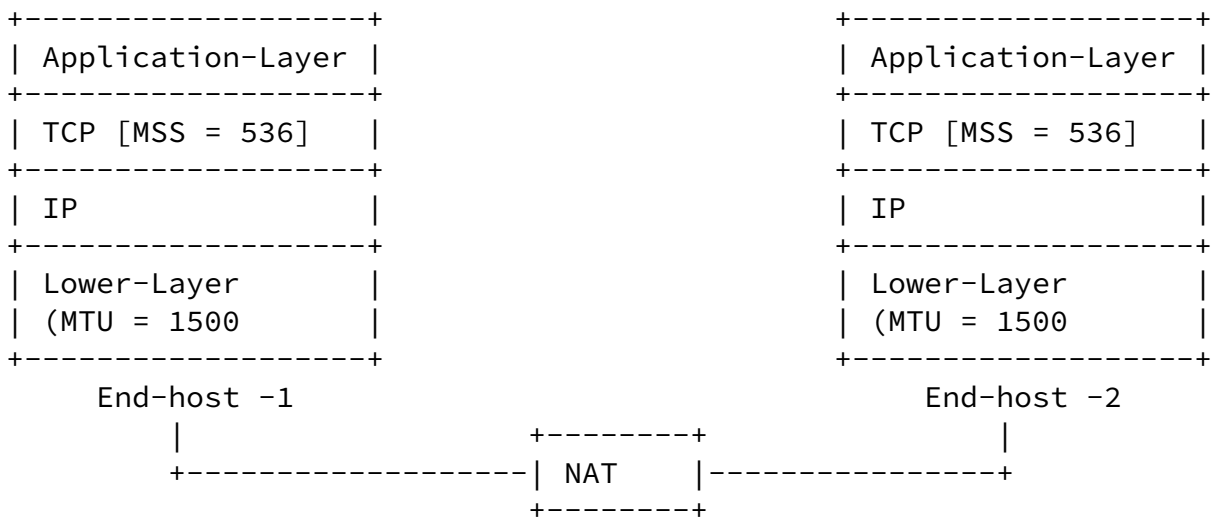
A NAT that is neither of these is referred to as Receive Fragments

None (abbreviated RF=N).

NAT MUST support RF=O or RF=OO. NAT MAY support RF=OO.

3.5.2 TCP Segments handling

A NAT may receive a TCP segmented packet. The following diagram explains this:



Say the application layer is sending data with size = 600. Since the MTU is 1500 this should not be an issue with respect to IP fragmentations, but this packet will still be TCP segmented. The following 2 TCP segments might appear on the wire:

TCP Segment 1:

IP[Frag-offset=0,More-flags =0	TCP hdr[Payload-Len=536]	Appl-data1

TCP Segment 2:

IP[Frag-offset=0,More-flags =0	TCP hdr[Payload-Len=64]	Appl-data2

There are 2 scenarios :

1. The TCP segments are received by NAT, in-order

2. The TCP segments are received by NAT, out-of-order

For out-of-order segments, if the NAT has the intelligence to process the application-payload, it should be able to figure out the out-of-order TCP segments and enforce some queueing mechanism such that when the prior segment is received it SHOULD reassemble the TCP segments and proceed with the application-payload processing. In addition it SHOULD send a TCP ACK for getting subsequent TCP segments from the endpoint.

For in-order segments, the application has to provide the length of data. In such cases the NAT has to enforce the same queuening mechanism as mentioned above and reassemble the TCP segments and proceed for its processing. This applies only for application which has the support for providing the length in the application-data.

NATs that behave as described in this section are refered to as "Support Segmentation Yes" (abbreviated SS=Y). NATs that do not support any of the above are called "Support Segmentation none" (abbreviated SS=N).

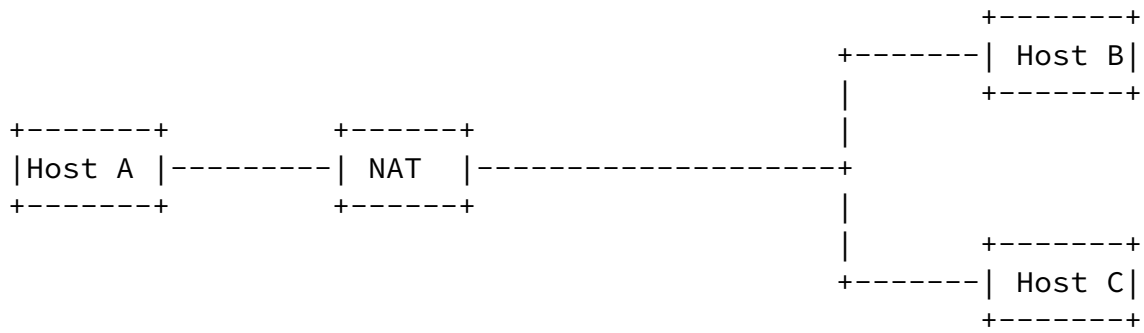
If a NAT is doing application-payload processing it SHOULD support SS=Y.

3.6 Adjusting Sequence Acknowledgement Numbers

A NAT might modify the TCP data carrying the application-payload resulting in the TCP data to increase or decrease in size. As a result of this the NAT is expected change/remember the sequence/ acknowledgement number in the TCP header and save this information along with the delta of change, in the NAT entry associated with this particular packet flow so that subsequent TCP packets of this flow are adjusted accordingly.

If a NAT receives a particular type of TCP payload for which it is capable/enabled to do a deep packet inspection it is expected to create a NAT entry with the full flow information irrespective of the type of NAT configuration, and save the [seq, ack, delta] associated with the TCP flow. It may chose to store any other additional information.

For eg:



Say the NAT is configured to do NAT and not PAT, meaning it is configured to do 1:1 translation say (A->A'). In such cases most NATs will create a simple NAT entry in the database to optimize on the memory or otherwise, and not create one with the full flow information.

So when A is sending a TCP flow to B & to C and the NAT does not have the intelligence to do a deep packet inspection for this TCP payload, then the NAT can chose to create just the following entry in the NAT database.

Prt	L-Addr	L-Port	G-Addr	G-Port	Dest Addr	Dest Port
-	A	-	A	-	-	-

Where L-Addr/Port are the internal address/port of the Host, G-Addr/port are the NAT Translated address/port.

But if A sends TCP traffic for which the NAT can do deep packet inspection and thereby possibly change the TCP payload-data-length, then the NAT SHOULD create entries with full flow information with the information about the [seq, ack,delta].

Prt	L-Addr	L-Port	G-Addr	G-Port	Dest Addr	Dest Port	
TCP	A	p	A	p'	B	q	[seq, ack, delta]

Say for example the NAT is not doing so and the TCP sessions from A->B and A->C are not differentiable then the subsequent TCP flows'

sequence/acknowledgement number-fixups will be incorrect.

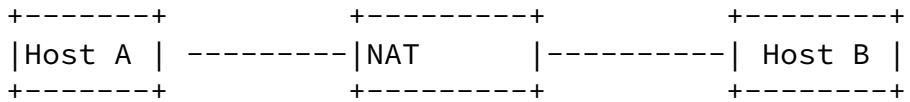
NATs that operate as described in this section are described as "Supports Seq/Ack Delta Adjustment" (abbreviated SDA=S). NATs that do not operate in this mode are described as "Supports Seq/Ack Delta Adjustment none" (abbreviated SDA=N). NATs MAY chose to be SDA=N for TCP flows for which it does not do deep packet inspection but SHOULD do SDA=S for the ones it does.

A NAT SHOULD support SDA=S.

3.7 Handling of ICMP Error messages

In addition to what has been described in [UDP-REQ] with respect to handling ICMP Error-messages, a NAT is expected to do a fixup of the embedded TCP payload in the ICMP Error message. As a result the ICMP Error message will correctly communicate the proper address/port information to the TCP stack on endpoint which can associate it with its internal data-structures and act accordingly.

For eg:



Say A sends a TCP packet via NAT to Host B (A,p) -> (B,p). NAT changes the source address/port of the TCP header to say (A',p'). For some reason the B sends back an ICMP Error message back towards A'. This ICMP Error message will contain part of the TCP header. The NAT is expected to not only modify the ICMP header but look into the TCP payload in the ICMP Error message and modify the TCP header accordingly so that when the ICMP Error is processed by the stack on A, it is able to correctly co-relate with its internal data-structures and act accordingly.

NATs that operate as described in this section are described as

"Supports ICMP Fixup Yes" (abbreviated SIF=Y).

NATs that do not are described as "Supports ICMP Fixup No" (abbreviated SIF=N).

A NAT SHOULD support SIF=Y.

4. Security considerations

NATs are often deployed to achieve security goals. Most of the recommendations and requirements in this document do not affect the security properties of these devices, but a few of them do have security implications and are discussed in this section.

When a fragmented packet is received from the external side and the packets are out of order so that the initial fragment does not arrive first, many systems simply discard the out of order packets.

Moreover, since some networks deliver small packets ahead of large ones, there can be many out of order fragments. NATs that are capable of delivering these out of order packets are possible but they need to store the out of order fragments, which can open up a DoS opportunity. Fragmentation has been a tool used in many attacks, some involving passing fragmented packets through NATs and others involving DoS attacks based on the state needed to reassemble the fragments. NAT implementers SHOULD be aware of [TINY] and [FRAG]

5. IANA Considerations

There are no IANA considerations.

6. IAB Considerations

There are no IAB considerations.

7. Acknowledgements

The authors would like to thank Cullen Jennings & Nagendra Modadugu for their review comments.

8. References

8.1 Normative References

[KEYWRD] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.

[UNSAF] Daigle, L. and IAB, "IAB Considerations for Unilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

8.2 Informative References

[ASND] Reynolds, J. and J. Postel, "Assigned numbers", RFC 923, October 1984.

[FRAG] Ziemba, G. and D. Reed, "Security Considerations for IP Fragment Filtering", RFC 1858, October 1995.

[H323] "Packet-based Multimedia Communications Systems, ITU-T Recommendation H.323", July 2003.

[HOST] Braden, R., "Requirements for Internet-Hosts - Communication Layers", RFC 1122, October 1998.

[ICE] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP) draft-ietf-mmusic-ice-02 (work in progress)", RFC , July 2004.

[ICMP] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[NAT-1] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[NAT-2] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[NAT-3] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator", RFC 3027, January 2001.

[P2P-1] Ford, B., Srisuresh, P. and D. Kegel, "Peer-to-Peer(P2P) communication across Network Address Translators(NATs) draft-ford-midcom-p2p-03 (work in progress)", June 2004.

[P2P-2] Ford, B. and D. Andersen, "Nat Check Web Site: http://midcom-p2p.sourceforge.net", June 2004.

[RTP] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[SIP] Rosenberg, J., Schulzrinne,, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[STUN-1] Rosenbert, J., Weinberger, J., Huitema, C. and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489,

March 2003.

[STUN-2] Jennings, C., "NAT Classification Results using STUN, draft-jennings-midcom-stun-results-01 (work in progress)", July 2004.

[TINY] Miller, I., "Protection Against a Variant of the Tiny Fragment Attack", RFC 3128, June 2001.

[UDP-REQ] Audet, F. and C. Jennings, "NAT Behavioral Requirements for Unicast UDP, draft-ietf-behave-nat-00.txt (work-in-progress)", January 2005.

[V4-REQ] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

Authors' Addresses

Senthil Sivakumar
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
USA

Phone:
Email: ssenthil@cisco.com

Kaushik Biswas
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
USA

Phone: +1 408 525 5134
Email: kbiswas@cisco.com

Bryan Ford
M.I.T.
Laboratory for Computer Science
77 Massachusetts Ave.
Cambridge, MA 02139
USA

Phone: 1-617-253-5261
Email: baford@mit.edu

Intellectual Property Statement

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.