BEHAVE
Internet-Draft
Intended status: Standards Track
Expires: August 17, 2007

D. Wing
J. Rosenberg
Cisco Systems

February 13, 2007

Controlling NAT Bindings using STUN
draft-wing-behave-nat-control-stun-usage-01

Status of this Memo

Copyright Notice

Abstract

Simple Traversal Underneath NAT (STUN) is a mechanism for traversing NATs. STUN requests are transmitted through a NAT to external STUN servers. While this works very well, its two primary drawbacks are the inability to modify the properties of a NAT binding and the need to query a public STUN server for every NAT binding. These drawbacks require frequent messages which present a load on servers (like SIP servers and STUN servers) and are bad for low speed access networks, such as cellular. This document proposes that the STUN server be

embedded in the NAT itself, and describes how these STUN servers can be readily discovered and utilized to reduce queries to public STUN servers and to reduce NAT keepalive traffic.

Table of Contents

1. Introduction

Two common usages of STUN [I-D.ietf-behave-rfc3489bis] are Binding Discovery and NAT Keepalive. The Binding Discovery usage allows a STUN client to learn its public IP address (from the perspective of the STUN server it contacted) and the NAT keepalive usage allows a STUN client to keep an active NAT binding alive. Unlike some other techniques (e.g., UPnP [UPnP], MIDCOM [RFC3303], Bonjour [Bonjour]), STUN does not interact directly with the NAT. Because STUN doesn't interact directly with the NAT, STUN cannot request additional services from the NAT such as longer lifetimes (which would reduce keepalive messages).

This paper describes a mechanism for the STUN client to interact directly with the NAT and request additional services, by using the STUN protocol itself. Thereafter, the STUN client need only ask that NAT's embedded STUN server for public IP addresses and UDP ports -- as it will return the same information as the public STUN server. Additionally, the STUN client can ask the NAT's embedded STUN server to extend the NAT binding for the flow, and the STUN client can learn the IP address of the next-outermost NAT. By repeating this procedure with the next-outermost NAT, all of the NATs along that path can have their bindings extended. By learning all of the STUN servers on the path between the public Internet and itself, an endpoint can optimize the path of peer-to-peer communications.

2. Motivation

There are a number of problems with existing NAT traversal techniques such as STUN [RFC3489], [UPnP], and [Bonjour]):

nested NATs
: Today, many ISPs provide their subscribers with modems that have embedded NATs, often with only one physical Ethernet port. These subscribers then install NATs behind those devices to provide

additional features, such as wireless access.

Nested NATs are, unfortunately, becoming quite common and often occur without the knowledge of users. For example, some ISPs provide their subscribers with modems that include integrated NAT functionality. When the subscriber installs another NAT, perhaps to provide himself with wireless network access, the endpoints are now behind nested NATs. Another example is a NAT in the basement of an apartment building or a campus dormitory, which combined with a NAT within the home or dormitory room also result in nested NATs. In both of these situations, UPnP and Bonjour no longer function at all, as those protocols can only control the first

NAT. STUN continues to function, but is unable to optimize network traffic behind those nested NATs (e.g., traffic that stays within the same house or within the same apartment building). The technique described in this paper allows optimization of the traffic behind those NATs so that the traffic can traverse the fewest NATs possible.

chattiness

To perform its binding discovery, a STUN client communicates to a server on the Internet. This consumes bandwidth across the user's access network which in some cases is bandwidth constrained (e.g., wireless, satellite).

STUN's chattiness is often cited as a reason to use other NAT traversal techniques such as UPnP or Bonjour. However, those NAT traversal techniques bring restrictions (they both require a UPnP-aware or Bonjour-aware NAT, they do not work with nested NATs, and they only work within one broadcast domain). The technique described in this paper provides a significant reduction in STUN's chattiness, to the point that the only time a STUN client needs to communicate with the STUN server on the public Internet is when the STUN client is initialized.

incremental deployment

Many NAT traversal techniques require the endpoint and the NAT to both support the new feature or else NAT traversal isn't possible at all.

However, the technique described in this paper allows incremental

deployment of local endpoints, local NATs, and remote endpoints and their remote NATs which support the features described in this paper. Only the local endpoints and the NATs on the path to their STUN server need to implement the technique in this paper to optimize their functionality.

3. Conventions Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4. Overview of Operation

When a STUN client sends a STUN Request to a STUN server, it receives a STUN Response which indicates the IP address and UDP port seen by the STUN server. If the IP address and UDP port differs from the IP

address and UDP port of the socket used to send the request, the STUN client knows there is at least one NAT between itself and the STUN server, and knows the 'public' IP address of that NAT. For example, in the following diagram, the STUN client learns the public IP address of its NAT is 192.0.2.1:

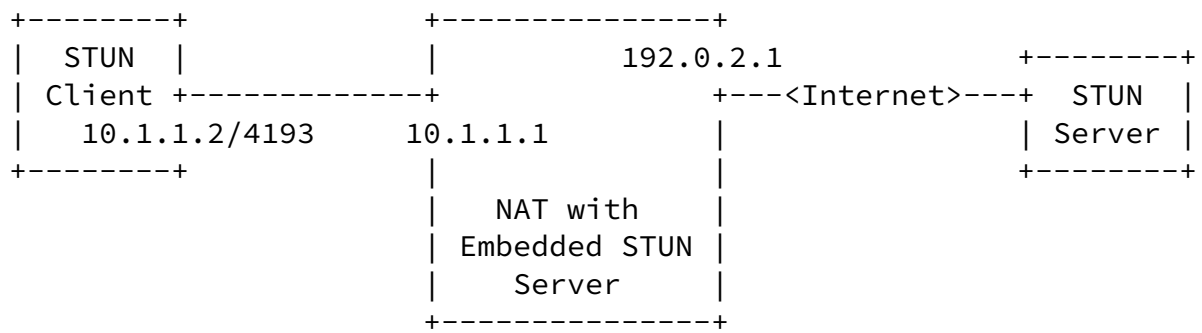


Figure 1: One NAT with embedded STUN server

After learning the public IP address of its outer-most NAT, the STUN client attempts to communicate with the STUN server embedded in that outer-most NAT. The STUN client does this by first obtaining a

shared secret, over a TLS connection, to the NAT's public IP address (192.0.2.1 in the figure above). After obtaining a shared secret, it sends a STUN Binding Request to the NAT's public IP address. The NAT will return a STUN Binding Response message including the XOR-INTERNAL-ADDRESS attribute, which will indicate the IP address and UDP port seen on the *internal* side of the NAT for that translation. In the example above, the IP address and UDP port indicated in XOR-INTERNAL-ADDRESS are the same as that used by the STUN client (10.1.1.2/4193), which indicates there are no other NATs between the STUN client and that outer-most NAT.

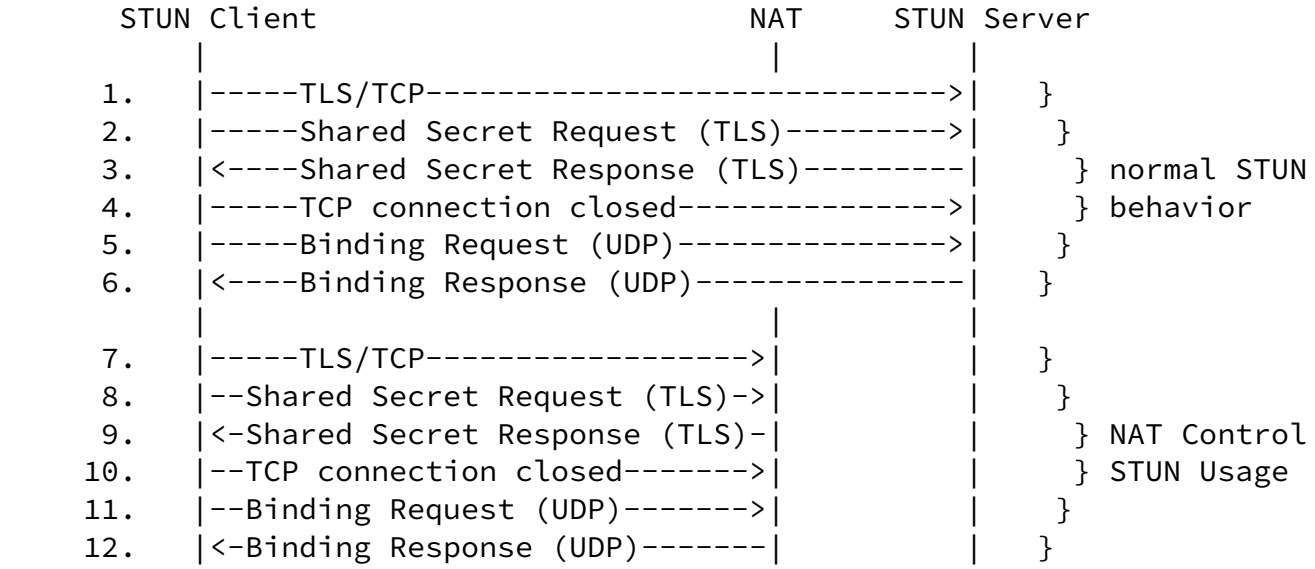


Figure 2: Communication Flow

In the call flow above, steps 1-6 are normal STUN behavior [I-D.ietf-behave-rfc3489bis]:

1: STUN client initiates a TLS-over-TCP connection to its STUN server on the Internet.

2: Using that connection, the STUN client sends Shared Secret Request to that STUN server.

3: Using that connection, the STUN server sends Shared Secret Response. This contains the STUN username the client should use for subsequent queries to this STUN server, and the STUN password that will be used to integrity-protect subsequent STUN messages with this STUN server.

4: TCP connection is closed.

5: STUN client sends UDP Binding Request to its STUN server on the Internet, using the STUN username obtained from that STUN server (from step 3).

6: STUN server responds with UDP Binding Response, integrity protected with the STUN password (from step 3). The STUN client now knows the public IP address of its outer-most NAT. This is used in the next step.

The next steps are the additional steps performed by a STUN client that has implemented the NAT Control STUN Usage:

7: STUN client initiates a TLS-over-TCP connection to the STUN server embedded in its outer-most NAT.

8: Using that connection, the STUN client sends Shared Secret Request to that STUN server.

9: Using that connection, the STUN server sends Shared Secret Response. This contains the STUN username the client should use

for subsequent queries to this STUN server, and the STUN password that will be used to integrity-protect subsequent STUN messages with this STUN server.

10: TCP connection is closed.

11: STUN client sends UDP Binding Request to the STUN server embedded in its outer-most NAT, using the STUN username obtained from from that STUN server (from step 10).

12: STUN server responds with UDP Binding Response, integrity protected with the STUN password (from step 10).

The response obtained in the message 12 contains the XOR-MAPPED-ADDRESS attribute which will have the same value as when the STUN server on the Internet responded (in step 6). The STUN client can perform steps 11-12 for any new UDP communication (e.g., for every new phone call), without needing to repeat steps 1-10. This meets the desire to reduce chattiness.

The response obtained in message 12 will also contain the XOR-INTERNAL-ADDRESS, which allows the STUN client to repeat steps 7-12 in order to communicate with all the on-path NATs between itself and its STUN server on the Internet. This is described in detail in section Section 4.1. This meets the desire to optimize traffic between nested NATs.

The STUN client can request each NAT to increase the binding lifetime, as described in Section 5.5. The STUN client receives positive confirmation that the binding lifetime has been extended, allowing the STUN client to significantly reduces its NAT keepalive traffic. Additionally, as long as the NAT complies with [RFC4787], the STUN client's keepalive traffic need only be sent to the outer-most NAT's IP address. This further meets the desire to reduce chattiness.

4.1. Nested NATs

Nested NATs are controlled individually. The nested NATs are discovered, from outer-most NAT to the inner-most NAT, using the XOR-

INTERNAL-ADDRESS attribute.

The following diagram shows how a STUN client iterates over the NATs to communicate with all of the NATs in the path. First, the STUN client would learn the outer-most NAT's IP address by performing the steps above. In the case below, however, the IP address and UDP port indicated by the XOR-INTERNAL-ADDRESS will not be the STUN client's own IP address and UDP port -- rather, it's the IP address and UDP port on the *outer* side of the NAT-B -- 10.1.1.2.

Because of this, the STUN client repeats the procedure and sends another STUN Binding Request to that newly-learned address (the *outer* side of NAT-B). NAT-B will respond with a STUN Binding Response containing the XOR-INTERNAL-ADDRESS attribute, which will match the STUN client's IP address and UDP port. The STUN client knows there are no other NATs between itself and NAT-B, and finishes.

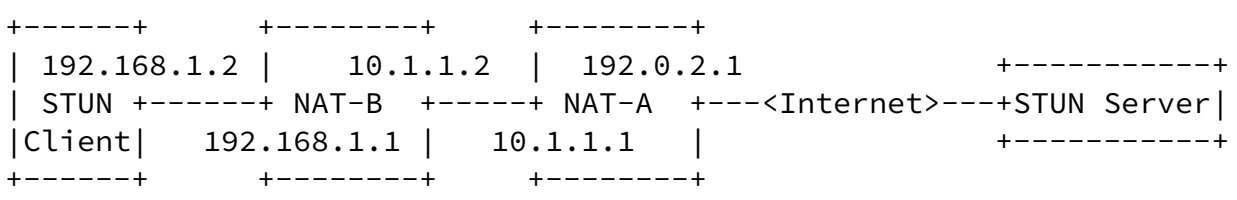


Figure 3: Two NATs with embedded STUN servers

Internally, the NAT can be diagrammed to function like this, where the NAT operation occurs before the STUN server.

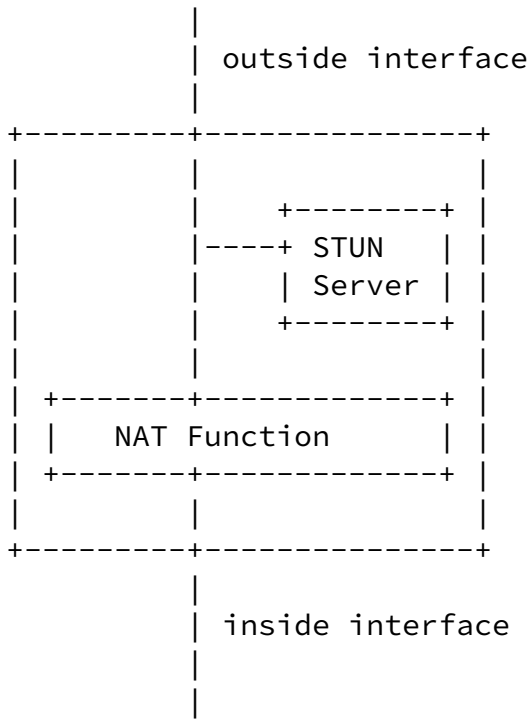


Figure 4: Block Diagram of Internal NAT Operation

4.2. Interacting with Legacy NATs

There will be cases where the STUN client attempts to communicate with an on-path NAT which does not support the usage described in this document. There are two cases:

- the NAT does not run a STUN server on its public interface (this will be the most common)
- the NAT does run a STUN server on its public interface, but doesn't return the XOR-INTERNAL-ADDRESS attribute defined in this document

In both cases the optimizations described in this document won't be available to the STUN client and the STUN client. This is no worse than the condition today. This allows incremental upgrades of applications and NATs that implement the technique described in this document.

5. NAT Control Usage

This section describes a new STUN usage, following the recommendation

for defining a new usage in [I-D.ietf-behave-rfc3489bis].

5.1. Applicability

This STUN usage MAY be used by a STUN client that discovers there is a NAT between itself and its STUN server. Such discovery would most likely occur with a STUN Binding Request / Binding Response exchange to a STUN server on the Internet, and by noticing the IP address and UDP port indicated by the XOR-MAPPED-ADDRESS attribute of the STUN Binding Response differs from the local socket's IP address and UDP port. Such a difference indicates a NAT is present between the STUN client and its STUN server.

5.2. Client Discovery of Server

As this usage involves communicating with on-path NATs directly, the client needs to find those NATs. The outer-most NAT is found by the normal XOR-MAPPED-ADDRESS attribute in a STUN Response. To iterate through the inner NATs, each NAT needs to support the usage described in this document, and the STUN client discovers each of those NATs by iterating through the XOR-INTERNAL-ADDRESS attribute returned by those NATs. This is described in diagrams and examples in Section 4.

5.3. Server Determination of Usage

If a STUN Binding Request is received from a NAT's private interface and sent to the IP address of its public interface, the STUN server can assume the NAT Control Usage.

5.4. New Requests or Indications

This usage does not define any new message types.

5.5. New Attributes

The following figure indicates which attributes are present in which messages for this usage. An M indicates that inclusion of the attribute in the message is mandatory, O means its optional, C means it's conditional based on some other aspect of the message, and -

means that the attribute is not applicable to that message type.

Attribute	Request	Response	Error Response	Indication
XOR-INTERNAL-ADDRESS	-	M	-	-
REFRESH-INTERVAL	O	C	-	-

5.5.1. XOR-INTERNAL-ADDRESS

This attribute MUST be present in a Binding Response and may be used in other responses as well. This attribute is necessary to allow a STUN client to 'walk backwards' and communicate directly with all of the STUN-aware NATs along the path.

The format of the XOR-INTERNAL-ADDRESS attribute is:

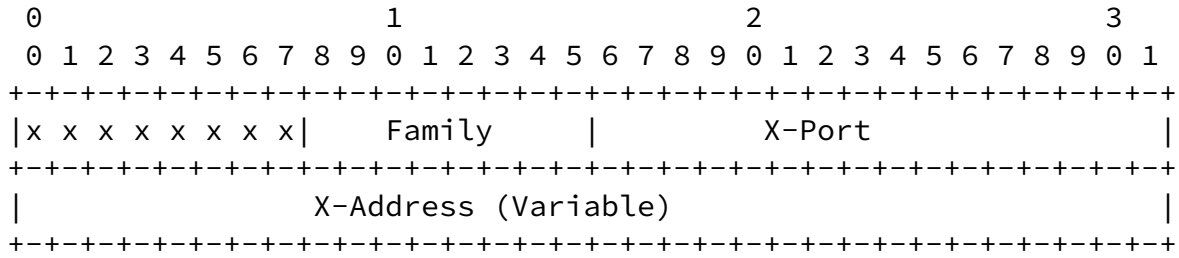


Figure 6: XOR-INTERNAL-ADDRESS Attribute

The meaning of Family, X-Port, and X-Address are exactly as in [I-D.ietf-behave-rfc3489bis]. The length of X-Address depends on the address family (IPv4 or IPv6).

5.5.2. REFRESH-INTERVAL

The REFRESH-INTERVAL attribute is defined in [I-D.ietf-behave-rfc3489bis] where it can only appear in a response. In the NAT Control usage defined in this document, the REFRESH-INTERVAL may also appear in a request.

In a Binding Request, the REFRESH-INTERVAL indicates the desired mapping timeout. In a Binding Response, the REFRESH-INTERVAL

indicates the NAT's mapping timeout.

5.6. Client Procedures

The STUN client sends a STUN Binding Request to its STUN server on the Internet and receives a STUN Binding Response, as normal. The STUN Binding Response contains the XOR-MAPPED-ADDRESS attribute which indicates the IP address and UDP port of the STUN Binding Request, as seen by the STUN server. If this IP address differs from the STUN client's IP address, the STUN client knows there is at least one NAT between itself and the STUN server, and it continues with the procedure; otherwise, it stops.

The STUN client now knows the public IP address of its outer-most NAT -- it was returned in the XOR-MAPPED-ADDRESS attribute. The STUN client performs the Shared Secret Usage (as described in

[I-D.ietf-behave-rfc3489bis]) with the public IP address of its outer-most NAT. After performing that usage, the STUN client now has a STUN USERNAME and PASSWORD which authenticate subsequent messages between the STUN client and this NAT's STUN server.

If subsequent messages from the STUN server fail authentication, the STUN client MUST re-obtain its IP address from a public STUN server, not from its outer-most NAT (see section Section 8.3).

To modify an existing NAT mapping's attributes, or to request a new NAT mapping for a new UDP port, the STUN client can now send a STUN Binding Request to the IP address of address in its outer-most NAT's STUN UDP port (3478). The NAT's STUN server will respond with a STUN Binding Response containing an XOR-MAPPED-ADDRESS attribute (which points at the NAT's public IP address and port -- just as if the STUN Binding Request had been sent to a STUN server on the public Internet) and an XOR-INTERNAL-ADDRESS attribute (which points to the source IP address and UDP port the packet STUN Binding Request packet had prior to being NATted).

If the XOR-INTERNAL-ADDRESS attribute indicates an IP address and UDP port different from the STUN client's own IP address and port, the STUN client knows there is at least one NAT between itself and the STUN server it last contacted. If the STUN client wants to use multiple STUN servers, or wants to control the properties of the NAT

bindings in each of those NATs, the STUN client can iteratively perform the Short-Term Password Usage followed by the Binding Discovery Usage with each NAT learned via the XOR-INTERNAL-ADDRESS attribute from the previous NAT.

In each case where the STUN client is sending STUN Binding Requests to the NATs, the STUN client can also include other STUN attributes to request certain properties for the flow. Requesting certain properties may require the STUN client to obtain short-term credentials. Defined in this document is a requested lifetime for the NAT binding in order to reduce keepalive traffic (REFRESH-INTERVAL).

5.7. Server Procedures

The server should listen for STUN Shared Secret Requests and STUN Binding Requests on the STUN UDP and TCP ports (UDP/3478, TCP/3478) on its public IP address, from hosts connected to its private interface(s). The NAT SHOULD only respond to such message which arrive from its 'internal' interface. STUN Binding Requests sent to the NAT's public IP address which arrived from its public interface SHOULD be handled as if the NAT isn't listening on that port (e.g., return an ICMP error).

After receiving a STUN Shared Secret Request, the NAT follows the procedures described in the Short-Term Usage section of [I-D.ietf-behave-rfc3489bis]. The embedded STUN server MUST store that username and password so long as any NAT bindings, created or adjusted with that same STUN username, have active mappings on the NAT.

After receiving a STUN Binding Request containing the REFRESH-INTERVAL attribute, the server SHOULD authenticate the request using the USERNAME attribute and the previously-shared STUN password (this is to defend against resource starvation attacks, see Section 8.1). If authenticated, the new binding's lifetime can be maximized against the NAT's configured sanity check and the lifetime indicated in the REFRESH-INTERVAL attribute of the STUN Binding Response.

In addition to its other attributes, the Binding Response always contains the XOR-MAPPED-ADDRESS and XOR-INTERNAL-ADDRESS attributes. The XOR-MAPPED-ADDRESS contains the public IP address and UDP port

for this binding. The XOR-INTERNAL-ADDRESS contains the IP address and UDP port of the STUN Binding Request prior to the NAT translation. The XOR-INTERNAL-ADDRESS is used by the STUN client to walk backwards through nested NATs.

For example, looking at Figure 1, the XOR-INTERNAL-ADDRESS is the IP address and UDP port _prior to_ the NAPT operation. If there is only one NAT, as shown in Figure 1, XOR-INTERNAL-ADDRESS would contain the STUN client's own IP address and UDP port. If there are multiple NATs, XOR-INTERNAL-ADDRESS would indicate the IP address of the next NAT (that is, the next NAT closer to the STUN client). Iterating over this procedure allows the STUN client to find all of the NATs along the path.

6. Benefits

6.1. Incremental Deployment

NAT Control can be incrementally deployed. If the outer-most NAT does not support it, the STUN client behaves as normal. Where the outer-most STUN NAT does support it, the STUN client can gain some significant optimizations as described in the following sections.

Likewise, there is no change to applications if NATs are deployed which support NAT Control.

6.2. Optimize SIP-Outbound

In sip-outbound [I-D.ietf-sip-outbound], the SIP proxy is also the STUN server. This document enables two optimizations of SIP-Outbound's keepalive mechanism:

1. STUN keepalive messages need only be sent to the outer-most NAT, rather than across the access link to the SIP proxy, which vastly reduces the traffic to the SIP proxy, and;

2. all of the on-path NATs can explicitly indicate their timeouts,

reducing the frequency of keepalive messages.

6.3. Optimize ICE

The NAT Control usage provides several opportunities to optimize ICE [I-D.ietf-mmusic-ice].

6.3.1. Candidate Gathering

During its candidate gathering phase, an ICE endpoint normally contacts a STUN server on the Internet. If an ICE endpoint discovers that its outer-most NAT runs a STUN server, the ICE endpoint can use the outer-most NAT's STUN server rather than using the STUN server on the Internet. This saves access bandwidth and reduces the reliance on the STUN server on the Internet -- the STUN server on the Internet need only be contacted once.

6.3.2. Keepalive

> [Note: In ICE-13, the keepalives were changed to STUN Indications. If this change to ICE becomes working group consensus for ICE keepalives, this section in this document should be deleted.]

ICE uses STUN as its primary media stream keepalive mechanism. This document enables two optimizations of ICE's keepalive techniques:

1. STUN keepalive messages need only be sent to the outer-most NAT, rather than across the access link to the remote peer, and;

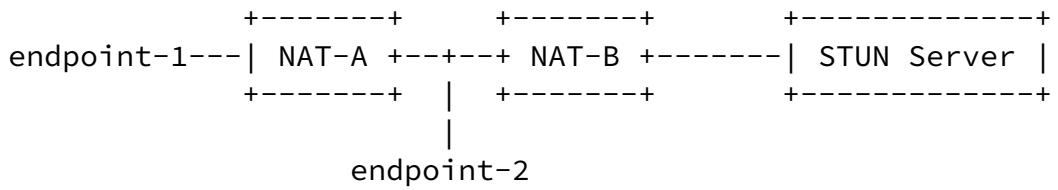
2. all of the on-path NATs can explicitly indicate their timeouts, reducing the frequency of keepalive messages.

6.3.3. Learning STUN Servers without Configuration

ICE allows endpoints to have multiple STUN servers, but it is difficult to configure all of the STUN servers in the ICE endpoint --

it requires some awareness of network topology. By using the 'walk backward' technique described in this document, all the on-path NATs and their embedded STUN servers can be learned without additional configuration. By knowing the STUN servers at each address domain, ICE endpoints can optimize the network path between two peers.

For example, if endpoint-1 is only configured with the IP address of the STUN server on the left, endpoint-1 can learn about NAT-B and NAT-A. Utilizing the STUN server in NAT-A, endpoint-1 and endpoint-2 can optimize their media path so they make the optimal path from endpoint-1 to NAT-A to endpoint-2:



7. Limitations

7.1. Overlapping IP Addresses with Nested NATs

If nested NATs have overlapping IP address space, there will be undetected NATs on the path. When this occurs, the STUN client will be unable to detect the presence of NAT-A if NAT-A assigns the same UDP port. For example, in the following figure, NAT-A and NAT-B are both using 10.1.1.x as their 'private' network.

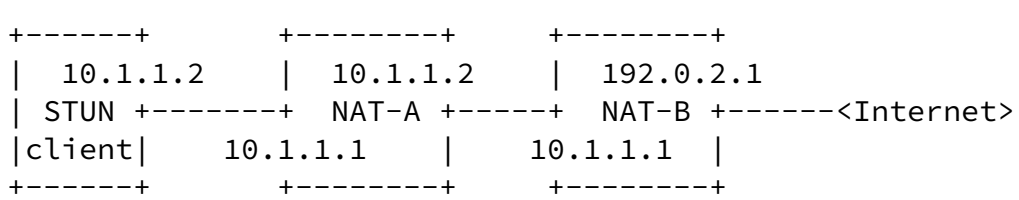


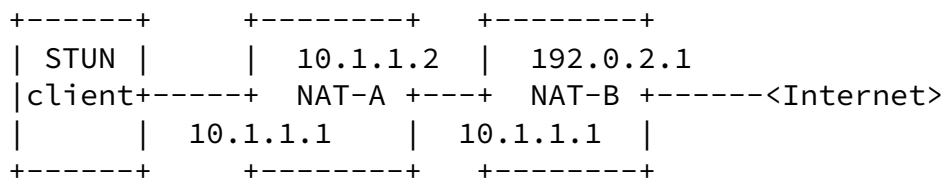
Figure 8: Overlapping Addresses with Nested NATs

When this situation occurs, the STUN client can only learn the outer-most address. This isn't a problem -- the STUN client is still able to communicate with the outer-most NAT and is still able to avoid consuming access network bandwidth and avoid communicating with the public STUN server. All that is lost is the ability to optimize paths within the private network that has overlapped addresses.

7.2. Address Dependent NAT on Path

In order to utilize the mechanisms described in this document, a STUN Request is sent from the same source IP address and source port as the original STUN Binding Discovery message, but is sent to a different destination IP address -- it is sent to the IP address of an on-path NAT. If there is an on-path NAT, between the STUN client and the STUN server, with 'address dependent' or 'address and port-dependent' mapping behavior (as described in section 4.1 of [RFC4787]), that NAT will prevent a STUN client from taking advantage of the technique described in this document. When this occurs, the ports indicated by XOR-MAPPED-ADDRESS from the public STUN server and the NAT's embedded STUN server will differ.

An example of such a topology is shown in the following figure:



In this figure, NAT-A is a NAT that has address dependent mapping. Thus, when the STUN client sends a STUN Binding Request to 192.0.2.1 on UDP/3478, NAT-A will choose a new public UDP port for that communication. NAT-B will function normally, returning a different port in its XOR-MAPPED-ADDRESS, which indicates to the STUN client that a symmetric NAT exists between the STUN client and the STUN server it just queried (NAT-B, in this example).

Figure 9: Address Dependant NAT on Path

Open issue: We could resolve this problem by introducing a new STUN attribute which indicates the UDP port the STUN client wants to control. However, this changes the security properties of NAT Control, so this seems undesirable.

Open issue: When the STUN client detects this situation, should we recommend it abandon the NAT Control usage, and revert to operation as if it doesn't support the NAT Control usage?

7.3. Address Dependent Filtering

If there is an NAT along the path that has address dependent filtering (as described in section 5 of [RFC4787]), and the STUN client sends a STUN packet directly to any of the on-path NATs public

addresses, the address-dependent filtering NAT will filter packets

from the remote peer. Thus, after communicating with all of the on-path NATs the STUN client MUST send a UDP packet to the remote peer, if the remote peer is known.

Discussion: How many filter entries are in address dependent filtering NATs? If only one, this does become a real limitation if NATs are nested; if they're not nested, the outer-most NAT can avoid overwriting its own address in its address dependent filter.

8. Security Considerations

This security considerations section will be expanded in a subsequent version of this document. So far, the authors have identified the following considerations:

8.1. Authorization and Resource Exhaustion

Only hosts that are 'inside' a NAT, which a NAT is already providing services for, can query or adjust the timeout of a NAT mapping.

A malicious STUN client could ask for absurdly long NAT bindings (days) for many UDP sessions, which would exhaust the resources in the NAT. To ensure the STUN client is not spoofing its IP address when launching such an attack, the STUN server can challenge requests to extend the timeout by sending a NONCE to the STUN client. The STUN server can authorize an extension to the refresh timeout if a new request is sent with that same NONCE value.

Without considering this document and without considering STUN or other UNSAF NAT traversal techniques, a malicious TCP client can open many TCP connections, and keep them open, causing resource exhaustion in the NAT. A NAT which provide protection against such a TCP attack can provide a similar level of protection, via the NONCE challange/ response, as they can for TCP sessions.

8.2. Comparison to Other NAT Control Techniques

Like UPnP, Bonjour, and host-initiated MIDCOM, the STUN usage described in this document allows a host to learn its public IP

address and UDP port mapping, and to request a specific lifetime for that mapping.

However, unlike those technologies, the NAT Control usage described in this document only allows each UDP port on the host to create and adjust the mapping timeout of its own NAT mappings. Specifically, an application on a host can only adjust the duration of a NAT bindings for itself, and not for another application on that same host, and

not for other hosts. This provides security advantages over other NAT control mechanisms where malicious software on a host can surreptitiously create NAT mappings to another application or to another host.

8.3. Rogue STUN Server

As described in Section 6, a STUN client can learn its outer-most NAT runs an embedded STUN server. However, without the STUN client's knowledge, the outer-most NAT may acquire a new IP address. This could occur when the NAT moves to a new mobile network or its DHCP lease expires. When the NAT acquires a new IP address, the STUN client will send a STUN Binding Request to the NAT's prior public IP address, which will be routed to the NAT's previous address.

If an attacker runs a rogue STUN server on that address, the attacker has effectively compromised the STUN server (the attacked described in section 12.2.1 of [RFC3489]). The attacker will send STUN Binding Responses indicating his IP address, which will be indistinguishable, to the STUN client, from the behavior of the legitimate STUN server.

To defend against this attack, the STUN client and STUN server obtain a short-term password as described in section Section 5.6.

9. IANA Considerations

This section registers one new STUN attribute per the procedures in [I-D.ietf-behave-rfc3489bis]:

0x0026	XOR-INTERNAL-ADDRESS

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-behave-rfc3489bis] Rosenberg, J., "Simple Traversal Underneath Network Address Translators (NAT) (STUN)", draft-ietf-behave-rfc3489bis-05 (work in progress), October 2006.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC3489] Rosenberg, 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.

10.2. Informational References

[UPnP] UPnP Forum, "Universal Plug and Play", 2000, <http://www.upnp.org>.

[Bonjour] Apple Computer, "Bonjour", 2005, <http://www.apple.com/macosx/features/bonjour/>.

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[I-D.ietf-mmusic-ice] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-13 (work in progress), January 2007.

[I-D.ietf-sip-outbound]

Jennings, C. and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-07 (work in progress), January 2007.

Authors' Addresses

Dan Wing
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: dwing@cisco.com

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
USA

Email: jdrosen@cisco.com

Full Copyright Statement

OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).