NAT Traversal for VoIP

Ai-Chun Pang
Graduate Institute of Networking and Multimedia
Dept. of Comp. Sci. and Info. Engr.
National Taiwan University
References

- “SIP, NAT and Firewalls”, Fredrik Thernelius
- Baruch Sterman and David Schwartz, “NAT Traversal in SIP”, Deltathree
- “STUN – Simple Traversal of UDP Through Network Address Translators”, RFC 3489, IETF
- “An Extension to the SIP for Symmetric Response Routing”, RFC 3581, IETF
- “TURN – Traversal Using Relay NAT”, Internet Draft, IETF
Outline

- Introduction
- Problems of NAT Traversal for VoIP
- Possible Solutions for VoIP over NAT
What is NAT?

- NAT - Network Address Translation
- Converts Network Address (and Port) between private and public realm
- Works on IP layer
- Transparent to Upper-layer Applications
Flavors of NAT [1/3]

Static NAT

- Requires the same number of globally IP addresses as that of hosts in the private environment
- Maps between internal IP addresses and external addresses is set manually
  - This mapping intends to stay for a long period of time
Flavors of NAT [2/3]

**Dynamic NAT**

- Collect the public IP addresses into an IP address pool
- A host connecting to the outside network is allocated an external IP address from the address pool managed by NAT
Flavors of NAT [3/3]

NAPT (Network Address and Port Translation)

- A special case of Dynamic NAT
  - Use port numbers as the basis for the address translation
- Most commonly used
Types of NAT

- Full Cone
- Restricted Cone
- Port Restricted Cone
- Symmetric
Full Cone NAT

- Client sends a packet to public address A.
- NAT allocates a public port (12345) for private port (21) on the client.
- Any incoming packet (from A or B) to public port (12345) will dispatch to private port (21) on the client.
Client sends a packet to public address A.
NAT allocate a public port (12345) for private port (21) on the client.
Only incoming packet from A to public port (12345) will dispatch to private port (21) on the client.
Client sends another packet to public address B.

NAT will reuse allocated public port (12345) for private port (21) on the client.

Incoming packet from B to public port (12345) will now dispatch to private port (21) on the client.
Port Restricted Cone NAT

- Client sends a packet to public address A at port 20202.
- NAT will allocate a public port (12345) for private port (21) on the client.
- Only incoming packet from address A and port 20202 to public port (12345) will dispatch to private port (21) on the client.

**Mapping Table**

<table>
<thead>
<tr>
<th>Client IP: 10.0.0.1</th>
<th>Port: 21</th>
<th>IP: 202.123.211.25</th>
<th>Port: 12345</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.1:21 &lt;-&gt; 12345 (for A: 20202)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0.0.1:21 &lt;-&gt; 12345 (for A: 30303)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Computer A**

- IP: 222.111.99.1
- Port: 20202
- Port: 30303
Symmetric NAT

- NAT allocates a public port each time the client sends a packet to different public address and port.
- Only incoming packet from the original mapped public address and port will dispatch to private port on client.

Mapping Table:
- 10.0.0.1:21 <-> 12345 (for A: 20202)
- 10.0.0.1:21 <-> 45678 (for B: 10101)
VoIP Protocol and NAT

- NAT converts IP addresses on IP layer

  Problem 1:
  - SIP, H.323, Megaco and MGCP are application layer protocol but contain IP address/port info in messages, which is not translated by NAT

  Problem 2:
  - Private client must send an outgoing packet first (to create a mapping on NAT) to receive incoming packets
Solving NAT Traversal Problems

Objectives
- To discover the mapped public IP & port for a private IP & port
- To use the mapped public IP & port in application layer message
- To keep this mapping valid

Issues
- NAT will automatically allocate a public port for a private address & port if needed.
- NAT will release the mapping if the public port is “idle”
  - No TCP connection on the port
  - No UDP traffic on the port for a period
- Keep a TCP connection to destination
- Send UDP packets to destination every specified interval
NAT Solutions

- IPv6 (Internet Protocol Version 6)
- UPnP (Universal Plug-and-Play)
- Proprietary protocol by NAT/ Firewall
  - SIP ALG (Application Level Gateway)
- SIP extensions for NAT traversal
  - RFC 3581
  - Works for SIP only, can not help RTP to pass through NAT
- STUN (Simple Traversal of UDP Through Network Address Translators)
  - RFC 3489
  - Works except for symmetric NAT
- TURN (Traversal Using Relay NAT)
  - draft-rosenberg-midcom-turn-04
  - for symmetric NAT
Two Distinct Cases – NAT Deployment [1/2]

Case I: SIP Provider is the IP Network Provider
Two Distinct Cases – NAT Deployment [2/2]

Case II: SIP Provider is NOT IP Network Provider
Solution for Case I – ALG [1/2]

Separate Application Layer NAT from IP Layer NAT

- Like MEGACO Decomposition
  - MG = Packet Filter
  - MGC = Control Proxy

- Advantages
  - Better scaling
  - Load balancing
  - Low cost
Solution for Case I – ALG [2/2]

- A control Protocol between application-layer NATs and IP-layer NATs
- Main Requirements
  - **Binding Request**: To give a private address and obtain a public address
  - **Binding Release**
    - **Open Hole** (firewall)
    - **Close Hole** (firewall)
Proposed Solution for Case II

**Much harder problem**
- No way to control firewall or NAT
- Cascading NATs
- Variable firewall NAT behaviors

**Proposed Solution**
- Make SIP “NAT-Friendly”
  - Minor extensions
  - Address the issues for SIP only, not RTP
  - Accepted by IETF (RFC 3581)
- Develop a protocol for traversal of UDP through NAT
  - Work for RTP
  - Also support other applications
SIP Extension to NAT Friendly

Client Behavior

- Include an “rport” parameter in the Via header
  - This parameter MUST have no value
  - It serves as a flag
- The client SHOULD retransmit its INVITE every 20 seconds
  - Due to UDP NAT binding period and to keep the binding fresh
SIP Extension to NAT Friendly [2/2]

Server Behavior

- Examines the Via header field value of the request
  - If it contains an “rport” parameter,
    - A “received” parameter
    - An “rport” parameter
- The response MUST be sent to the IP address listed in the “received” parameter, and the port in the “rport” parameter.
Example [1/2]

Client A: 10.1.1.1
Proxy B: 68.44.10.3
NAT C: 68.44.20.1

- A issues request
  INVITE sip:user@domain SIP/2.0
  Via: SIP/2.0/UDP 10.1.1.1:4540; rport

- A→C (mapping port 9988)→B
  INVITE sip:user@domain SIP/2.0
  Via: SIP/2.0/UDP proxy.domain.com
  Via: SIP/2.0/UDP 10.1.1.1:4540;
  received=68.44.20.1;rport=9988;
Example [2/2]

- Server B receives the response
  SIP/2.0 200 OK
  Via: SIP/2.0/UDP proxy.domain.com
  Via: SIP/2.0/UDP
  10.1.1.1:4540; received=68.44.20.1; rport=9988;

- B (68.44.10.3:5060) → C (68.44.20.1:9988) → A
  SIP/2.0 200 OK
  Via: SIP/2.0/UDP
  10.1.1.1:4540; received=68.44.20.1; rport=9988;
UPnP [1/2]

- Universal Plug and Play
- It is being pushed by Microsoft
  - Windows® Messenger
- A UPnP-aware client can ask the UPnP-enabled NAT how it would map a particular IP:port through UPnP
- It will not work in the case of cascading NATs

http://www.upnp.org/
**UPnP [2/2]**

- **A: Private Network**
  - UPnP-aware Internet gateway device
  - The UPnP-enabled NAT allows “A” to be aware of its external IP

- **B: Public Internet**
  - “B” and “A” can communicate with each other
External Query

- A server sits listening for packets (NAT probe)
- When receiving a packet, it returns a message from the same port to the source containing the IP:port that it sees
STUN

- Simple Traversal of UDP Through NAT
- RFC 3489
- In Working Group IETF MIDCOM Group
- Simple Protocol
- Works with existing NATs
- Main features
  - Allow Client to Discover Presence of NAT
  - Works in Multi-NAT Environments
  - Allow Client to Discover the Type of NAT
  - Allows Client to Discover the Binding Lifetimes
  - Stateless Servers
STUN Server

- Allow client to discover if it is behind a NAT, what type of NAT it is, and the public address & port NAT will use.
- A simple protocol, easy to implement, little load

Client wants to receive packet at port 5060
Send a query to STUN server from port 5060
STUN Server receives packet from 202.123.211.25 port 12345
STUN Server send a response packet to client. Tell him his public address is 202.123.211.25 port 12345
Binding Acquisition

- STUN Server can be ANYWHERE on Public Internet
- Call Flow Proceeds Normally
STUN Message [1/3]

- TLV (type-length-value)
- Start with a STUN header, followed by a STUN payload (a series of STUN attributes depending on the message type)
- Format

| STUN Header | STUN Payload (can have none to many blocks) |
STUN Message [2/3]

STUN Payload (can have none to many blocks)

<table>
<thead>
<tr>
<th>Message Type (16 bits)</th>
<th>Message Length (16 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction ID (128 bits)</td>
<td></td>
</tr>
</tbody>
</table>

**Message Types**

- 0x0001: Binding Request
- 0x0101: Binding Response
- 0x0111: Binding Error Response
- 0x0002: Shared Secret Request
- 0x0102: Shared Secret Response
- 0x0112: Shared Secret Error Response
### STUN Message [3/3]

<table>
<thead>
<tr>
<th>Attribute Type (16 bits)</th>
<th>Attribute Length (16bits)</th>
<th>Attribute Value (Variable length)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute Types</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0001: MAPPED-ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0002: RESPONSE-ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0003: CHANGE-REQUEST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0004: SOURCE-ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0005: CHANGED-ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0006: USERNAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0007: PASSWORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0008: MESSAGE-INTEGRITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0009: ERROR-CODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000a: UNKNOWN-ATTRIBUTES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000b: REFLECTED-FROM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Automatic Detection of NAT Environment [1/2]

<table>
<thead>
<tr>
<th>Test</th>
<th>Destination</th>
<th>Change IP</th>
<th>Change Port</th>
<th>Return IP:port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test I</td>
<td>IP1:1</td>
<td>N</td>
<td>N</td>
<td>IP1:1</td>
</tr>
<tr>
<td>Test II</td>
<td>IP1:1</td>
<td>Y</td>
<td>Y</td>
<td>IP2:2</td>
</tr>
<tr>
<td>Test III</td>
<td>IP2:1</td>
<td>N</td>
<td>N</td>
<td>IP2:1</td>
</tr>
<tr>
<td>Test IV</td>
<td>IP1:1</td>
<td>N</td>
<td>Y</td>
<td>IP1:2</td>
</tr>
</tbody>
</table>
Automatic Detection of NAT Environment [2/2]

<table>
<thead>
<tr>
<th>Test</th>
<th>Destination</th>
<th>Change IP</th>
<th>Change Port</th>
<th>Return IP:port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test I</td>
<td>IP1:1</td>
<td>N</td>
<td>N</td>
<td>IP1:1</td>
</tr>
<tr>
<td>Test II</td>
<td>IP1:1</td>
<td>Y</td>
<td>Y</td>
<td>IP2:2</td>
</tr>
<tr>
<td>Test III</td>
<td>IP2:1</td>
<td>N</td>
<td>N</td>
<td>IP2:1</td>
</tr>
<tr>
<td>Test IV</td>
<td>IP1:1</td>
<td>N</td>
<td>Y</td>
<td>IP1:2</td>
</tr>
</tbody>
</table>
Binding Lifetime Determination

If it receives Binding Response on socket X, the binding has not expired.

Another Binding Request, RESPONSE-ADDRESS is set to (Pa, Pp)

Start Timer T

If it receives Binding Response on socket X, the binding has not expired.

Socket X

Bind Req.

Bind (Pa, Pp)

Binding Resp.

MAPPED-ADDRESS (Pa, Pp)

Socket Y

Client

STUN

Another Binding Request, RESPONSE-ADDRESS is set to (Pa, Pp)
Binding Acquisition Procedure

Shared Secret Request and Response

Binding Request and Response (Pa, Pp)

Binding Request and Response (Pa', Pp')

RESPONSE-ADDRESS is set to (Pa, Pp)

SIP Message

RTP
STUN - Pros and Cons

Benefits
- No changes required in NAT
- No changes required in Proxy
- Works through most residential NAT

Drawbacks
- Doesn’t allow VoIP to work through Symmetric NAT
- RTCP may not work
Is STUN suitable for Symmetric NAT

- Absolutely not

Client A
IP: 10.0.0.1
Port: 21

Client B
IP: 222.111.88.2
Port: 10101

STUN Server
IP: 222.111.99.1
Port: 20202

Mapping Table
10.0.0.1:21 <-> 12345 (for **222.111.99.1 : 20202**)

NAT
IP: 202.123.211.25
Port: 12345
Solutions for Symmetric NATs

- Connection Oriented Media
- RTP-Relay
Connection Oriented Media

- The endpoint outside the NAT must wait until it receives a packet from the client before it can know where to reply.

- Add a line to the SDP message (coming from the client behind the NAT)
  
  \[ \texttt{a=direction:active} \]

- The initiating client will “actively” set up the IP:port to which the endpoint should return RTP
  - The IP:port found in the SDP message should be ignored.
Problem?

1) If the endpoint does not support the `a=direction:active` tag.

2) If both endpoints are behind Symmetric NATs.
RTP-Relay

- For either of the cases considered in the previous slide, one solution is to have an RTP Relay in the middle of the RTP flow between endpoints.

- The RTP Relay acts as the second endpoint to each of the actual endpoints that are attempting to communicate with each other.
The following is a typical call flow that might be instantiated between a User Agent behind a symmetric NAT and a voice gateway on the open Internet.
TURN

- Traversal Using Relay NAT
- draft-rosenberg-midcom-turn-06.txt

Private NET

 TURN
 Client

NAT

 TURN
 Server

Public Internet
Obtaining a One Time Password

1. Client generates and sends **Shared Secret Request** (with no attribute)

2. TURN Server reject it with a **Shared Secret Error Response** (code=401, contain **NONCE** and **REALM**)

3. Client generate a new **Shared Secret Request** (contain **NONCE**, **REALM**, **USERNAME**)

4. TURN Server generate a **Shared Secret Response** (contain **USERNAME** and **PASSWORD**)
Allocating a Binding

1. Client generates and sends **Initial Allocate Request** (contain **BANDWIDTH**, **LIFETIME**, **USERNAME**, **MESSAGE_INTEGRITY**)}

2. TURN Server generates and sends **Allocate Response** (contain **MAPPED_ADDRESS**, **LIFETIME**, **BANDWIDTH**, **MESSAGE_INTEGRITY**)
Refreshing a Binding

1. Client generates and sends Subsequent Allocate Request (contain LIFETIME, USERNAME, MESSAGE_INTEGRITY)

2. TURN Server generates and sends Allocate Response (contain MAPPED_ADDRESS, LIFETIME, MESSAGE_INTEGRITY, MAGIC_COOKIE)
Sending Data

1. TURN Client generates and sends **Send Request** (contain `DESTINATION_ADDRESS` and `DATA`).

2. TURN Server sets default destination address to `DESTINATION_ADDRESS`, and add this address to the list of permission. Then TURN Server relays the data to Peer.

3. TURN Server generates and sends **Send Response** to TURN Client.
1. Peer sends packet to the mapped address of TURN Client.

2. TURN Server checks whether the source IP address and port are equal to the default destination address or not.

3. TURN Server checks whether the source IP address and port are listed amongst the set of permission for the binding or not.

4. TURN Server generates **Data Indication** message to relay the packet to TURN Client.
Tearing Down a Binding

1. Client generates and sends Subsequent Allocate Request (contain LIFETIME=0)

2. TURN Server will tearing down the binding.
TURN – Pros and Cons

- Pros
  - No change required in NAT.
  - Work through firewall and all kinds of NAT.

- Cons
  - Long latency
  - Heavy load for TURN server