NAT
Firewall
NAT Traversal

Systems Integration
Credits

Based on the work "How to traverse NAT" by Davide Carboni

License

Attribution-ShareAlike 2.5
You are free:
to copy, distribute, display, and perform the work
to make derivative works
to make commercial use of the work
Under the following conditions:

Attribution. You must give the original author credit.
Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.
For any reuse or distribution, you must make clear to others the licence terms of this work.
Any of these conditions can be waived if you get permission from the copyright holder.
Your fair use and other rights are in no way affected by the above.
This is a human-readable summary of the Legal Code (the full licence).

Disclaimer
Private Address Space

- **RCF 1918** sets aside three ranges of IP addresses for private networks
  - 10.0.0.0/8
  - 192.168.0.0/16
  - 172.16.0.0 through 172.31.255.255 (172.16/12 prefix)

- Do not route addresses in these blocks to the Internet
Port Ranges

- Well-known
  - 1-1023

- Registered
  - 1024-49151

- Dynamic/Private
  - 49152-65535
Network Address Translation (NAT)

- NAT, as defined by RFC 1631, is a router function where IP addresses (and possibly port numbers) of IP datagrams are replaced at the boundary of a private network.

- NAT is a method that enables hosts on private networks to communicate with hosts on the Internet.

- NAT is run on routers that connect private networks to the public Internet, to replace the IP address-port pair of an IP packet with another IP address-port pair.

<table>
<thead>
<tr>
<th>Inside Local IP addresses</th>
<th>Outside Global IP addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.4.4.5</td>
<td>2.2.2.2</td>
</tr>
<tr>
<td>10.4.1.1</td>
<td>2.2.2.2</td>
</tr>
</tbody>
</table>
Network Address Translation (NAT) Addresses

Used to address Inside hosts

- Inside Local
  - used by host on the private side
- Inside Global
  - public often registered IP address into which the inside local is translated

Used to address Outside hosts

- Outside Global
  - actual IP address of a host that resides on the outside public network
- Outside Local
  - IP address used to translate an outside global IP address
NAT Inside/Outside & Local/Global Relationship

Inside Network

SA Inside Local  DA Outside Local

DA Inside Local  SA Outside Local

Packet Direction

Outside Network

SA Inside Global  DA Outside Global

DA Inside Global  SA Outside Global

SA=Source Address  DA=Destination Address
NAT/PAT → IP masquerading

- the process of **network address translation (NAT)** involves re-writing the source and/or destination **addresses** of **IP packets** as they pass through a **router** or **firewall**.

- the process of **port address translation (PAT)** involves re-writing both **IP addresses** and **protocol ports** of the source and/or destination fields.

- NAT and PAT are generically known as **network masquerading or IP-masquerading**
Another point of view

Network Address Translation

- **One to one translation** of IP addresses from inside local IP address.

How to conserve IP addresses by translating a large pool of private addresses into a small pool of public addresses? → **PAT**

- Port Address Translation: **Many-to-one translation**, takes multiple inside local IP addresses and translates them to one (or few) inside global address.
- With PAT, or **address overloading**, hundreds of privately addressed nodes can access the Internet using only one global address.

**Connection tracking and timeouts**

- The NAT router monitors the different conversations and clear the mapping of a conversation after a timeout (2 minutes) or after the conversation has been closed.
Firewall

- A Firewall is a system that filters TCP/IP UDP/IP packet according to rules
- It can be a software running in the user machine or in a network router
Firewall
Translating Inside Local Addresses

<table>
<thead>
<tr>
<th>Inside local IP</th>
<th>Inside global IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.2.25</td>
<td>200.1.1.25</td>
</tr>
</tbody>
</table>

LAN switch

NAT Border Router

Internet

206.100.29.1
Simple NAT

NAT

Main Internet

(Public IP addresses)

(Public IP addresses)

(Private IP addresses)
Multiple NAT

Main Internet

ISP NAT

ISP network

Home NAT

Home network

156.148.70.32

192.168.2.12

192.168.2.99

10.0.0.12

(Public IP addresses)

(Private IP addresses)
NAT Mappings

Inside SRC | Outside SRC | Outside DST
---|---|---
Address  | Address | Address  
Port    | Port   | Port  
192.168.2.2 | 1.1.1.5 | 12.1.1.3  
4445     | 10000  | 7777   
Why NAT is so popular

NAT is an Internet standard that enables a local-area network (LAN) to use one set of IP addresses for internal traffic and a second set of addresses for external traffic.

NAT serves four main purposes:
- Provides a type of firewall by hiding internal IP addresses
- Enables a company to use more internal IP addresses. Since they're used internally only, there's no possibility of conflict with IP addresses used by other companies and organizations.
- Overcomes IPv4 addresses depletion.
- Enhances the reliability of local systems by stopping worms and enhance privacy by discouraging scans.

For these reasons, NAT is a standard feature in routers for home and small-office Internet connections.
Forms of NAT

NAT has many forms and can work in several ways:

- Static NAT
- Dynamic NAT
- Overloading NAT (PAT)
Static NAT

- Mapping an unregistered IP address to a registered IP address on a one-to-one basis. Particularly useful when a device needs to be accessible from outside the network.

Source: http://computer.howstuffworks.com/nat1.htm
Dynamic NAT

- Maps an unregistered IP address to a registered IP address from a group of registered IP addresses.

Source: http://computer.howstuffworks.com/nat1.htm
Overloading NAT (PAT)

- A form of dynamic NAT that maps multiple unregistered IP addresses to a single registered IP address by using different ports. This is known also as PAT (Port Address Translation), single address NAT or port-level multiplexed NAT.

In overloading, each computer on the private network is translated to the same IP address (213.18.123.100), but with a different port number assignment.
Advantages of NAT

- Allows you to increase or decrease the number of registered IP addresses without changing devices in the network.

- Static translations are manually configured to translate a single global IP address to a single local IP address.

- Dynamic mappings are configured on the NAT border router by using a pool of one or more registered IP addresses. Devices on the inside of the network that wish to communicate with a host on the outside network can use these addresses in the pool.
Advantages of NAT (Continued)

- NAT can be configured to allow the basic load sharing of packets among multiple servers using the TCP load distribution feature. TCP load distribution uses a single outside IP address, which is mapped to multiple addresses. Incoming connections are distributed in a round-robin fashion among IP addresses in the internal pool.

- If you switch ISPs and need to change the registered IP addresses you are using, NAT makes it so you don’t have to renumber every device in your network. The only change is the addresses that are being used in the NAT pool.
Disadvantages of NAT

- NAT increases latency. Every packet must be processed to see if it needs translation.
- NAT hides end-to-end IP addresses that render some applications unusable.
- Disadvantages include increased latency and difficulties with protocols or applications that put IP address in data portion of IP packet.
- NAT changes IP addresses making it unable to track IP flow end-to-end. The good thing is this eliminates a hacker’s ability to identify the packet’s true source.
Disadvantages of NAT (Continued)

- Because a host needs to be accessed from the outside network will have two IP addresses, one inside and one outside, this creates a problem called split DNS. You need to set up two DNS servers, one for external and one for internal addresses.
The problem

- The large deployment of NAT builds a barrier to the development of peer-to-peer and IoT applications because:
  - Host behind a NAT are only authorized to initiate outgoing traffic through a limited set of ports (UDP/TCP)
    - configurable
  - Host behind a NAT are never authorized to receive incoming TCP or UDP traffic initiated by a foreign host.
    - this rule depends on the NAT type
NAT types

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

- The NAT maintains a 'table' that links private and public addresses and port numbers.
- It is important to note that these 'bindings' can only be initiated by outgoing traffic.
- NAT breaks end-to-end semantics.
Methods to solve the NAT Problem

The current methods for solving NAT traversal are:

- Session (ex Simple) Traversal of UDP Through Network Address Translation devices (STUN)
- Traversal Using Relay NAT (TURN)
- Interactive Connectivity Establishment (ICE)

- Universal Plug and Play (UPnP)
- Application Layer Gateway
- Manual Configuration
- Tunnel Techniques
NAT Problem scenario

- Two peers want setting up a UDP channel.
- Each of them:
  1. is behind a different NAT/firewall.
  2. holds credential to access to a given rendezvous server.
  3. At the same time, initiates a connecting procedure
- The hypotheses 2) and 3) may be satisfied if the two peers share a signaling channel through which they exchange synchronization messages and credentials.
NAT Problem scenario - example

- Two peers connected with a SIP (Session Initiation Protocol) channel.
- They want setting up a VoIP communication
  - a bidirectional UDP-based channel
  - that transport RTP/UDP packet (voice pkt)
- Each of them must know UDP ports and IP addresses to which send UDP datagrams.
- But .... NAT change ports and addresses
- Thus ... Each of these peers must detect Inside Global IP address and UDP port of the other peer, to which to send UDP datagrams.
VoIP scenario

1) I want to talk with You

2) OK: my address is 10.0.0.2 : 57000

3) thank: my address is 192.168.2.99 : 56000

4) SRC 192.168.2.99:56000 DST 10.0.0.12:57000

5) SRC 10.0.0.12:57000 DST 192.168.2.99:56000
VoIP scenario - rendezvous server

SIP Proxy

NAT

Local network

SIP channel

NAT

Local network

host A

192.168.2.99
56000

host B

host C

10.0.0.12
57000

SIP Proxy

rendezvous server

Local network

Local network
VoIP scenario - rendezvous server

- The "rendezvous server" cooperates with each peer to solve NAT traversal problem:
  - imposes the **other peer's NAT to map Inside Local address** (and port) **into Inside Global address**,  
  - detects **Inside Global IP address and UDP port** of the other peer.

- Depending on NAT traversal solution, the rendezvous can remain in the transmission path or can not.

- Rendezvous server types include:
  - STUN
  - TURN

...
VoIP scenario - rendezvous server

SIP Proxy - public IP

1) what is my Inside Global address?

2) your Inside Global address is X Y

3) my Inside Global address is X Y

4) thanks, call to X Y
Traversing a NAT that does not collaborate

argh!
Rendezvous server types

- Traversing a NAT that does not collaborate requires a Rendezvous server.

- There are two types of Rendezvous server that perform two different operations:
  
  - Connection Reversal
    - STUN server
  
  - Relaying
    - TURN server
Connection reversal

- A public-IP STUN server (*rendezvous*) detect the NAT and collaborates in setting up a direct connection.
- All the subsequent packets flow directly.
- The rendezvous detects the public IP address and port to which are mapped the internal IP address and port.
Relaying

A public-IP relay (anchor point) (a TURN server) redirects all the packets using NAT techniques or other approaches.

All the packets flow through this relay.
NAT policies (1/2)

- **Full cone NAT**: is NAT where all requests from the same internal IP address and port are mapped to the same public IP address and port.
  - Once a mapping is created, all incoming traffic to the public address is routed to the internal host without checking the address of the remote host.

- **A restricted cone NAT**: like full cone all requests from the same internal IP address and port are mapped to the same public IP address and port.
  - Unlike a full cone NAT, a remote host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.
NAT policies (2/2)

- **A port restricted cone NAT** is like a restricted cone NAT, but the restriction includes port numbers. All requests from the same internal IP address and port are mapped to the same public IP address and port.

  - In addition, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.

- **A symmetric NAT** is a NAT where all requests from the same internal IP address and port to a specific destination IP address and port are mapped, within a conversation, to the same external source IP address and port. The port is chosen in a random way, one for conversation. If the same internal host sends a packet with the same source address and port to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a UDP packet back to the internal host.

  - Only the external host that receives a packet can send a UDP packet back to the internal host.
UDP Hole Punching

- Hole punching is a technique to allow traffic from/to a host behind a firewall/NAT without the collaboration of the NAT itself.

- A rendezvous detects the public IP address and port to which are mapped the internal IP address and port.

- The simplest way is to use UDP packets
Policy: all incoming traffic to the public address is routed to the internal host without checking the address of the remote host.
Full cone mapping and policy

- **Mapping**
  - 192.168.2.2:4445 <-> 1.1.1.4:10100

- **Policy**
  - ALLOW ALL TO 1.1.1.4:10100
Policy: all incoming traffic to the public address is routed to the internal host without checking the address of the remote host.
Policy: a remote host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X (even if that packet does not reached its destination)
Restricted cone mapping and policy

**Mapping**
- 192.168.2.2:4445 <-> 1.1.1.4:10100

**Policy**
- **ALLOW** 1.1.1.5 TO 1.1.1.4:10100
- **ALLOW** 1.1.1.6 TO 1.1.1.4:10100
Policy: an external host can send a packet, with source IP address X (and any source port P), to the internal host only if the internal host had previously sent a packet to IP address X (even if that packet does not reached its destination).
Port restricted cone

Policy: an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P (even if that packet does not reached its destination).
Port restricted cone mapping and policy

- **Mapping**
  - 192.168.2.2:4445 - 1.1.1.4:10100

- **Policy**
  - `ALLOW 1.1.1.5:7777 TO 1.1.1.4:10100`
  - `ALLOW 1.1.1.6:7777 TO 1.1.1.4:10100`
Policy: an external host C can send a packet, with **source IP address X** and **source port P**, to the internal host only if the internal host had previously sent a packet to IP address X and port P (even if that packet does not reached its destination).
Symmetric NAT

Policy: If the same internal host sends a packet with the same source address and port to a different destination, a different mapping is used. Only the external host that receives a packet can send a UDP packet back to the internal host.
Symmetric NAT mapping and policy

- **Mapping**
  - 192.168.2.2:4445 <-> 1.1.1.4:10100
  - 192.168.2.2:4445 <-> 1.1.1.4:10179

- **Policy**
  - `ALLOW 1.1.1.5:7777 TO 1.1.1.4:10100`
  - `ALLOW 1.1.1.6:7777 TO 1.1.1.4:10179`
Symmetric NAT - dynamic mapping

- The mapping expires after the connection has been closed (no packets received or transmitted after a given timeout).

- Recent implementation of Symmetric NAT changes the Inside Global (address:port) pair assigned to a given ((srcIP:srcport),(destIP:destport)) association in any subsequent mapping request.

  - This prevents us from predicting which Inside Global (address:port) pair will be assigned to a given ((srcIP:srcport),(destIP:destport)) association.
Holes in Symmetric NATs

The only way to traverse this NAT is by using a Relay.
Holes in Symmetric NATs - details

- If a peer C is behind a symmetric NAT and the other peer A is behind a full cone or restricted cone NAT.
  - the C peer can initiate the connection after a STUN server has detected the Inside Global (address:port) pair of the host A.
  - The C host sends packets to that Inside Global (address:port) pair.
  - The A host receives those packets and responds to the sender (the Inside Global (address:port) pair of C).

- If a peer C is behind a symmetric NAT and the other peer A is behind a port restricted cone NAT or symmetric NAT.
  - There is no way to setup a direct connection.
STUN protocol (to simplify hole punching)

- Protocol to discover the presence and types of NAT and firewalls between them and the public Internet

- STUN client in application (in private network)
- STUN server in the internet, which echo's information back.

- STUN allows applications to determine:
  - what type of NAT is running on your broadband router,
  - what public IP addresses and ports are allocated to these applications by the NAT
STUN protocol

- STUN is originally specified in RFC 3489 (updated in RFC 5389 "Session Traversal Utilities for NAT (STUN)", and RFC 7350 "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)"

- STUN defines the operations and the message format needed to understand the type of NAT

- To allow NAT behaviour detection, the STUN server listens on two IP addresses and two UDP ports (two instances of STUN)
How STUN detects NAT (1)

- The client sends a request to the server
- The server returns a response which contains (inside the payload) the source IP of the packet received from the client, i.e. the mapped IP address
- The client compares its IP address with the mapped IP address returned by the server
- If they are different then the client is behind a NAT
- The client can set some flags in the message which tell the server to send a packet from another IP/port or to another IP/port (the STUN server has 2 public IP addresses)
How STUN detects NAT (2)

- basic procedure

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 FlowProceeds 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]

The STUN Message structure consists of an STUN Header followed by an STUN Payload (which can have none to many blocks).

### STUN Header

- **Message Type** (16 bits)
- **Message Length** (16 bits)
- **Transaction ID** (128 bits)

### 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]

### Attribute Types

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>MAPPED-ADDRESS</td>
</tr>
<tr>
<td>0x0002</td>
<td>RESPONSE-ADDRESS</td>
</tr>
<tr>
<td>0x0003</td>
<td>CHANGE-REQUEST</td>
</tr>
<tr>
<td>0x0004</td>
<td>SOURCE-ADDRESS</td>
</tr>
<tr>
<td>0x0005</td>
<td>CHANGED-ADDRESS</td>
</tr>
<tr>
<td>0x0006</td>
<td>USERNAME</td>
</tr>
<tr>
<td>0x0007</td>
<td>PASSWORD</td>
</tr>
<tr>
<td>0x0008</td>
<td>MESSAGE-INTEGRITY</td>
</tr>
<tr>
<td>0x0009</td>
<td>ERROR-CODE</td>
</tr>
<tr>
<td>0x000a</td>
<td>UNKNOWN-ATTRIBUTES</td>
</tr>
<tr>
<td>0x000b</td>
<td>REFLECTED-FROM</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>
STUN Decision Tree

STUN Flow for type discovery process

Figure 2
RFC 3489
# Example of a STUN Session

<table>
<thead>
<tr>
<th>Frame</th>
<th>SRC IP</th>
<th>SRC PORT</th>
<th>Message &amp; Direction / Response</th>
<th>STUN SERVER DEST IP</th>
<th>DEST PORT</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Request ==&gt;</td>
<td>69.90.168.13</td>
<td>3478</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Response &lt;=: public IP is 209.150.62.132:17507</td>
<td>69.90.168.13</td>
<td>3478</td>
<td>ok - now we know what the public IP address of the router in front of the SIP client is, along with the port it's sending on</td>
</tr>
<tr>
<td>15</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Request ==&gt;</td>
<td>69.90.168.13</td>
<td>3479</td>
<td>ah .. We are sending to the +1 dest port</td>
</tr>
<tr>
<td>16</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Response &lt;=: public IP is 209.150.62.132:17507</td>
<td>69.90.168.13</td>
<td>3479</td>
<td>sending to the +1 port with the same dest ip doesn't seem to change the public port on the router.</td>
</tr>
<tr>
<td>17</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Request ==&gt; (change the dest ip and dest port +1)</td>
<td>69.90.168.14</td>
<td>3479</td>
<td>ok .. We've +1 on the dest IP address and left the port as above</td>
</tr>
<tr>
<td>18</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Response &lt;=: public IP is 209.150.62.132:17507</td>
<td>69.90.168.14</td>
<td>3479</td>
<td>Doesn't seem to matter - the public IP:Port only seems to depend on the src ip:port, not the</td>
</tr>
</tbody>
</table>
## STUN Debug (continued)

<table>
<thead>
<tr>
<th>Frame</th>
<th>IP Source</th>
<th>Port</th>
<th>Method</th>
<th>IP Destination</th>
<th>Port</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>192.168.0.12</td>
<td>8377</td>
<td>Binding Request ==&gt;</td>
<td>69.90.168.13</td>
<td>3478</td>
<td>OK .. Let's go back to where we began; this is the same as frame 13</td>
</tr>
</tbody>
</table>
| 20    | 192.168.0.12    | 8377 | Binding Response <=:  
public IP is 209.150.62.132:17507 | 69.90.168.13 | 3478 | and not surprisingly, this is the same as frame 14 |
| 21    | 192.168.0.12    | 5060 | Binding Request ==> | 69.90.168.14 | 3478 | OK .. Now we'll try changing the port on the src side .. Surprise, it's the same port as we use for SIP |
| 22    |                 |      |          | This turns out to be a http packet .. Nothing related to STUN |
| 23    | 192.168.0.12    | 5060 | Binding Response <=:  
public IP is 209.150.62.132:17508 | 69.90.168.14 | 3478 | ahhh. So what counts is the SRC port .. The public port on the router has changed!! So, now we know everything we need to know to figure out how to send SIP packets |
Is STUN suitable for Symmetric NAT?

- Absolutely not

Mapping Table

10.0.0.1:21 \(\rightarrow\) 12345 (for 222.111.99.1:20202)
STUN Limitations

- Does not work with symmetric NATs used by most corporate environments
- Does not work if both clients are behind the same NAT
- Once a client has discovered its external addresses, it can relate it to its peers, but ...
  - If the NATs are full cone then either side can initiate communication.
  - If they are restricted cone or restricted port cone both sides must start transmitting together.
- The trick is using STUN to discover the presence of NAT, and to use the bindings they allocate.
Revised STUN

- Revised specification in RFC 5389
  - "Session Traversal Utilities for NAT"
  - STUN is not a complete solution, rather it is a tool
    - ICE
    - SIP Outbound proxy, ...
  - Binding discovery, NAT keep-alives, Short-term password,
  - STUN Relay (previously TURN)
- Can run on UDP, TCP, TLS
**TURN protocol (also STUN Relay)**

- **TURN** is a protocol for UDP/TCP relaying behind a NAT.
- Unlike STUN, there is no hole punching and data are bounced to a public server called the TURN server.
- **TURN** is the last resource.
  - For instance, behind a symmetric NAT.

**Simplest solution:** traffic goes through relay

TURN is the last resource.

- For instance, behind a symmetric NAT.
Role in TURN

- A TURN client is an entity that generates TURN requests.
- A TURN Server is an entity that receives TURN requests, and sends TURN responses.
- The server is the data relay, receiving data on the address it provides to clients, and forwarding them to the clients.
More complex scenario

- Both client behind a symmetric NAT
- Using two TURN server
To be continued

- TURN details
ICE

Interactive Connectivity Establishment
ICE

“Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT)”

- Allows peers to discover NAT types and client capabilities
- Provide alternatives for establishing connectivity, using STUN and TURN
- Works with all types of NATs
- Designed for SIP and VoIP. Can be applied to any session-oriented protocol
- The detailed operation of ICE can be broken into six steps: gathering, prioritizing, encoding, offering and answering, checking, and completing.
ICE scenario

- The two UAs are each behind a NAT with unknown properties.
- They are capable of exchanging SDP messages through an offer/answer exchange used to setup media sessions between the UAs through a SIP server.
- ICE uses STUN/TURN server(s)
- Each UA can have its own or they can use the same one.
- Both UAs can use different transport addresses to communicate with another agent.
- ICE is used to discover which addresses can connect to each other and the method used to make that connection through the NAT.
ICE address candidates (1)

- To execute ICE UAs have to identify all address candidates, transport addresses. Transport addresses are a combination of IP address and port for a particular transport protocol.

- There are three types of candidates:
  - **Host Candidate** - transport address associated with a UA’s local interface
  - **Relayed Candidate** - transport address associated with a TURN server (can only be obtained from a TURN server)
  - **Server Reflexive Candidate** - translated address on the public side of the NAT (obtained from either a STUN server or a TURN server)
ICE address candidates (2)

- candidates

Relay

Relayed candidates reside on a host acting as a relay towards the agent

Server Reflexive candidates are addresses residing at the public side of a NAT

NAT

NAT

Host Candidates reside on the agent itself
Using TURN to Obtain Candidates

- Server reflexive and relayed candidates are learned jointly by talking to a TURN server.
- Client sends query to TURN server.
- Query passes through NAT, creates bindings.
- TURN server allocates a relayed address and also reports back source address of request to client.
  - This will be the server reflexive address.
ICE steps (simplified)

- Gathering address candidates
- Prioritizing candidates
- Offering candidates and Answering (using SDP)
- Pairing
- Pairs Verification (using STUN)
- Completing.
ICE steps (simplified)

- After UA1 has gathered all of its candidates, it arranges them in order of priority from highest to lowest and sends them to UA2 in attributes in an SDP offer message.

- UA2 performs the same candidate gathering and sends a SDP response with it’s list of candidates.

- Each UA takes the two lists of candidates and pairs them up to make candidate pairs.

- Each UA schedules connectivity check, STUN request/response transaction, to see which pairs work.
Using ICE in WebRTC applications

see https://www.w3.org/TR/webrtc/#simple-peer-to-peer-example for webrtc code example