CS514: Intermediate Course in Computer Systems

Lecture 10: February 10, 2003
“P2P using STUN and DNS SRV”

Reading Materials (1/2)

For the project:
- draft-ietf-midcom-stun-05.txt
  - STUN (Simple Traversal of UDP through NATs)
- RFC 2782
  - ”A DNS RR for specifying the location of services (DNS SRV)”
- RFC 2136
  - “Dynamic Updates in the Domain Name System (DNS UPDATE)”
### Reading Materials (2/2)

- **NAT extra reading:**
  - RFC 3022
    - "Traditional IP Network Address Translator (Traditional NAT)"
  - RFC 3027
    - "Protocol complications with the IP network address translator"
  - RFC 3235
    - "NAT-Friendly Application Design Guidelines"
  - RFC 3424
    - "IAB considerations for UNilateral self-address fixing (UNSAF) across network address translation"

### Attempts to fix NAT (1/2)

- **RSIP (Realm Specific IP)**
  - IETF work
  - Host can request an address and address+port assignment from the NAT box
  - Didn’t go anywhere

- **Microsoft UPnP (Universal Plug and Play)**
  - Broad initiative to allow cross-vendor plug-and-play in local network environment
    - Auto-configure into net, advertise its capabilities
  - NAT aspect: Client can learn of address/port mappings from NAT box, add new port mappings
  - I don’t know if this is taking off or not
Attempts to fix NAT (2/2)

- midcom (middlebox communications)
  - IETF working group
  - Broad effort to deal with all kinds of (now opaque) middle boxes (NATs, firewalls, Intrusion Detection Systems (IDS), etc.)
  - Usual standards committee trashing about
- STUN (Simple Traversal of UDP through NAT)
  - Bad name…try searching for it with Google!
  - Simple method for host to learn what port it got assigned (transparent to NAT box)
  - Then application can use this knowledge as it sees fit

I like STUN

- I think it will succeed
  - Note that, of these options, STUN is the only one that we can use!
  - Because doesn’t require NAT box cooperation
- I think it will be another nail in the coffin of IPv6
- I wish I had thought of it
### Types of NAT behaviors

<table>
<thead>
<tr>
<th>Types of cone</th>
<th>Port assignment policy</th>
<th>Firewall policy for incoming packets (from dest address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full cone</td>
<td>Same addr and port for every flow (from a given internal host)</td>
<td>Accept all flows to assigned address and port from any dest address</td>
</tr>
<tr>
<td>Restricted cone</td>
<td></td>
<td>Accept if packet previously sent to dest address</td>
</tr>
<tr>
<td>Port-restricted cone</td>
<td></td>
<td>Accept if packet previously sent to dest address and port</td>
</tr>
<tr>
<td>Symmetric</td>
<td>Different addr/port for every flow</td>
<td></td>
</tr>
</tbody>
</table>

Terminology partly applies to firewalls as well

- Full cone = no firewall
- Symmetric = firewall only allows outgoing flows
  - (typical)
- Firewall may also disallow UDP
What STUN does

- Tells you if you are behind a NAT
- If so:
  - Tells you the assigned address(es) and port(s)
  - Tells you what type of NAT
- If not:
  - Can still tell you what kind of firewall you are behind
    - (UDP blocking, symmetric UDP)

Typical STUN deployment

[Diagram showing STUN client, NAT1, private network, NAT2, private network, Global Internet, STUN Server, with labels: Your home NAT, ISP’s NAT]
Basic operation: query/reply

STUN Client → NAT1 → Global Internet → STUN Server

What is my global address and port?

sA=10.1.1.1, sP=5555
sA=10.1.1.2, sP=6666
sA=20.1.1.1, sP=7777

Your global address is 20.1.1.1, port is 7777
dA=10.1.1.1, dP=5555
dA=10.1.1.2, dP=6666
dA=20.1.1.1, dP=7777

Use learned address/port to tell peer how to reach you

Open port 5555

STUN Client → NAT1 → Global Internet → STUN Server

STUN Client → NAT1 → Global Internet → App Server
Use learned address/port to tell peer how to reach you

I’m at 20.1.1.1:7777

Voila, it works!

dA=20.1.1.1, dP=7777

dA=10.1.1.1, dP=5555

dA=10.1.1.2, dP=6666
Unless NAT is restricted!

If restricted NAT, must “punch hole” first
How to determine if NAT is restricted

- STUN server can send packets from two addresses and two ports
  - Primary and secondary
  - pA and pP, sA and sP
- STUN client can ask the STUN server to use the secondary port or address and port.

---

How to determine if NAT is restricted

Diagram showing the process:
- STUN Client
- NAT1
- NAT2
- Global Internet
- STUN Server

1. What is my addr/port?
2. Your addr/port is A/P. Here is sA and sP.
3. This time reply from sA and sP.
4. Ok, here is my reply.
5. Blocked if restricted NAT.
What if two ports needed?

- Two flows between peers

Two ports are needed for two flows between peers. This is illustrated in the diagram with two separate connections:

1. **STUN Client** to **NAT1** to **Global Internet** to **STUN Server**
   - Source Port (sP) = 1111, Destination Port (dP) = 2222
   - Source Port (sP) = 3111, Destination Port (dP) = 3333

2. **STUN Client** to **NAT2** to **Global Internet** to **App Server**
   - Source Port (sP) = 1112, Destination Port (dP) = 2222
   - Source Port (sP) = 3112, Destination Port (dP) = 3333

Learning two ports

The process of learning two ports involves the following steps:

1. **STUN Client** to **NAT1** to **Global Internet** to **STUN Server**
   - Source Port (sP) = 1111, Destination Port (dP) = 3478 (What is my addr/port?)
2. **STUN Server** to **NAT2** to **Global Internet** to **App Server**

   - Source Port (sP) = 1112, Destination Port (dP) = 3478 (What is my addr/port now?)

   - Your addr/port is A1/P1.

   - Source Port (sP) = 1112, Destination Port (dP) = 3478 (What is my addr/port now?)

   - Your addr/port is A1/P2.

This process causes a second port assignment, as indicated in the diagram.
Issues with learning two ports?

- If two ports assigned, which is used for a given peer-to-peer flow???
- STUN spec doesn’t mention two ports
  - You should experiment with this in the project
- Note that a given App may be smart enough to use a single port
  - But multiple distinct Apps may use STUN!

Keeping NAT assignments alive

- NAT box will time-out port assignment after inactivity (if UDP)
  - At end of TCP connection if TCP
- App must periodically send packets to keep NAT state alive
  - Every minute or so?
- Note that client can try to learn NAT box time-out value
  - But this takes time, and is prone to failure
What about this????

Don’t really want this…
And some NATs don’t allow it!

May use heuristics to decide if on same private network:

- Peers have same global IP address
  - But this may not happen
- Peers have same domain name
  - Doesn’t mean peers are in the same private network though
- Doesn’t hurt (much) to try local address and global address
What about this????

This is the only choice. No way to learn these addresses.

Discovering STUN servers

- Two ways:
  - By address
  - By name
    - By SRV record (preferred)
    - By A record (if SRV doesn’t work)
STUN limitations

- Doesn’t work with all NAT types
  - Esp. “symmetric NAT”, apparently common in enterprise networks
    - I believe NATs will change to accommodate STUN
- Only UDP, not TCP
- Does not help discovery of peers behind same NAT
  - But this easy to solve

Stuff I didn’t talk about

- Before the query/reply, there is a security phase over TCP using TLS
  - The STUN server securely gives you a temporary name and password
- Other details to overcome security problems
Remember this?

I'm at 20.1.1.1:7777

STUN Client

private net

NAT1

private net

NAT2

Global Internet

STUN Server

How did the App Server know the addr/port of the lower client?

Some kind of rendezvous is needed

I'm at 20.1.1.1:7777

Paul

10.1.1.1

I'm at 20.1.1.1:7777

20.1.1.1

STUN Server

App Server

I'm at 30.1.1.1:6666

Ken

10.1.1.1
Some kind of rendezvous is needed

Ken is at 30.1.1.1:6666
Paul is at 20.1.1.1:7777

STUN Server
App Server

DNS as a rendezvous service

- Dynamic DNS update
- SRV record
- Advantages:
  - Existing software and standards
  - DNS is robust
- Disadvantages:
  - No presence or other application-specific information
  - Limited types of queries
    - Works if you know who you want to talk to, but not good for discovering new peers
**DNS SRV Record (RFC 2782)**

- `_Service._Proto.Name SRV Priority Weight Port Target`

  - Determines load balancing
  - Host name (leads to IP address)
  - Provides port assignment flexibility. (This is why we are interested in this)

`_myapp._udp.host.domain.com`

**DNS SRV rendezvous operation (1/2)**

- In advance:
  - Make up a service name (“game”)
  - Make up a host name (“me.gamesite.com”)

- At host config time:
  - Configure addresses of dynamic DNS servers
DNS SRV rendezvous operation (2/2)

- At app execute time:
  - Learn STUN address (20.1.1.1) and port (7777)
  - Dynamically update DNS server
    - _game._udp.me.gamesite.com SRV X Y 7777 me.gamesite.com
    - me.gamesite.com A 20.1.1.1
  - Query DNS for game partner
    - SRV _game._udp.him.gamesite.com