ICE, TURN and STUN

Stephen Strowes
31/Oct/2008
NATs

- NAT Terminology
  - Full cone
  - Restricted cone
  - Port-restricted cone
  - Symmetric
  - Guarantees...
  - Packet rewriting (ALGs)...

INVITE
Send media to 192.168.1.2:4321
ICE, Interactive Connectivity Establishment

- ICE is a mechanism to permit media streams to flow between two peers in a NATed environment
- An extension to SIP, it can be used by other signalling mechanisms
ICE, Interactive Connectivity Establishment

- Really high-level of how ICE-enabled peers enable comms:
  1. Discover information about network, be pessimistic
  2. Exchange information about network (signalling)
  3. Systematically probe possibilities to find useful connection
ICE, Interactive Connectivity Establishment

- Allows hosts in same NAT realm to communicate directly...
  - ... and also ...
- Allows hosts behind symmetric NATs to communicate via a relay
- And variations in-between...
ICE, Terminology

• ICE deals with components
  • 1 component per media stream
    • e.g., 1 for RTP, 1 for RTCP
  • Each media stream may nominate multiple candidate addresses
• Candidate: A transport address (ip:port) which may offer reachability for data incoming from an opposing peer
ICE, Sequence of Events

• In a little more detail:
  1. Candidate gathering
     • STUN
     • TURN
  2. Prioritisation
  3. Exchange
  4. Connectivity checks
  5. Coordination
  6. Communication
ICE, Candidate Gathering

- Uses STUN & TURN
- Each host possibly has multiple candidates...
  - Host
  - Server reflexive
  - Relay
  - Peer reflexive (later...)

Diagram:

- STUN Server
- NAT
- Client
STUN: Session Traversal Utilities for NAT

  - Returns the public-side of the binding
  - XOR-mapped address
**STUN: Session Traversal Utilities for NAT**

- Returns the public-side of the binding
- XOR-mapped address
TURN: Traversal Using Relays around NAT

  - Allocations
    - Allocate a socket on the relay...
  - Permissions
    - Inform relay which locations it should accept packets from for relaying back to client
TURN: Traversal Using Relays around NAT

  - Allocations
    - Allocate a socket on the relay...
  - Permissions
    - Inform relay which locations it should accept packets from for relaying back to client
ICE, Candidate Gathering

- Uses STUN & TURN
- Possibly multiple candidates...
  - Relay
  - Server reflexive
  - Host
  - Peer reflexive (later...)

STUN server

NAT

Client
ICE, Prioritisation

- $\text{prio} = 2^{24}(\text{type\_pref}) + 2^{8}(\text{local\_pref}) + (256 - \text{component\_ID})$
- **Type preference:**
  - 0 Relayed candidates
  - 100 Server reflexive candidates
  - 110 Peer reflexive candidates
  - 126 Host candidates
- **Local preference:**
  - Preference by interface, by STUN server...
- **Component ID:**
  - As described (RTP=1; RTCP=2)
**ICE, Candidate Exchange**

- Signalling carries the gathered candidates
  - In SIP, INVITE & response
- SDP carries the candidates for ICE usage...
ICE, Candidate Exchange

- Signalling carries the gathered candidates
  - In SIP, INVITE & response
- SDP carries the candidates for ICE usage...

```
a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr 10.0.1.1 rport 8998
```
ICE, Candidate Exchange

- Signalling carries the gathered candidates
  - In SIP, INVITE & response
- SDP carries the candidates for ICE usage...

```
a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr 10.0.1.1 rport 8998
```
ICE, Candidate Exchange

- Signalling carries the gathered candidates
- SIP response carrying opposing peer's candidate set
ICE, Connectivity Checks

- Pair the local candidates off against the remote candidates
- Calculate pair priority as:
  \[ 2^{32} \min(P_L, P_R) + 2 \max(P_L, P_R) + (P_L > P_R ? 1 : 0) \]
- Order the list by priority...
- Prune duplicates
ICE, Connectivity Checks

Pair the local candidates off against the remote candidates

- HostL -- HostR
- SrflxL -- HostR
- RelayL -- HostR
- HostL -- SrflxR
- SrflxL -- SrflxR
- RelayL -- SrflxR
- HostL -- RelayR
- SrflxL -- RelayR
- RelayL -- RelayR
ICE, Connectivity Checks

- Prioritise and order candidates...
  
  \[ 2^{32} \min(P_L, P_R) + 2 \max(P_L, P_R) + (P_L > P_R \? 1:0) \]

  126-126  HostL  --  HostR
  100-126  SrflxL  --  HostR
  0-126    RelayL  --  HostR
  100-126  HostL  --  SrflxR
  100-100  SrflxL  --  SrflxR
  0-100    RelayL  --  SrflxR
  0-126    HostL  --  RelayR
  0-100    SrflxL  --  RelayR
  0-0      RelayL  --  RelayR
ICE, Connectivity Checks

- Prioritise and order candidates...
  - $2^{32} \min(P_L, P_R) + 2 \max(P_L, P_R) + (P_L > P_R ? 1 : 0)$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>126-126</td>
<td>HostL</td>
<td>HostR</td>
</tr>
<tr>
<td>100-126</td>
<td>SrflxL</td>
<td>HostR</td>
</tr>
<tr>
<td>100-126</td>
<td>HostL</td>
<td>SrflxR</td>
</tr>
<tr>
<td>100-100</td>
<td>SrflxL</td>
<td>SrflxR</td>
</tr>
<tr>
<td>0-126</td>
<td>RelayL</td>
<td>HostR</td>
</tr>
<tr>
<td>0-126</td>
<td>HostL</td>
<td>RelayR</td>
</tr>
<tr>
<td>0-100</td>
<td>RelayL</td>
<td>SrflxR</td>
</tr>
<tr>
<td>0-100</td>
<td>SrflxL</td>
<td>RelayR</td>
</tr>
<tr>
<td>0-0</td>
<td>RelayL</td>
<td>RelayR</td>
</tr>
</tbody>
</table>
ICE, Connectivity Checks

- Prune duplicates...
- Replace local candidates with their bases

126-126  HostL  --  HostR
100-126  SrflxL  --  HostR
100-126  HostL  --  SrflxR
100-100  SrflxL  --  SrflxR
0-126   RelayL  --  HostR
0-126   HostL  --  RelayR
0-100   RelayL  --  SrflxR
0-100   SrflxL  --  RelayR
0-0     RelayL  --  RelayR
ICE, Connectivity Checks

- Prune duplicates...
- Replace local candidates with their bases

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>126-126</td>
<td>HostL -- HostR</td>
<td></td>
</tr>
<tr>
<td>100-126</td>
<td>HostL -- HostR</td>
<td></td>
</tr>
<tr>
<td>100-126</td>
<td>HostL -- SrflxR</td>
<td></td>
</tr>
<tr>
<td>100-100</td>
<td>HostL -- SrflxR</td>
<td></td>
</tr>
<tr>
<td>0-126</td>
<td>RelayL -- HostR</td>
<td></td>
</tr>
<tr>
<td>0-126</td>
<td>HostL -- RelayR</td>
<td></td>
</tr>
<tr>
<td>0-100</td>
<td>RelayL -- SrflxR</td>
<td></td>
</tr>
<tr>
<td>0-100</td>
<td>HostL -- RelayR</td>
<td></td>
</tr>
<tr>
<td>0-0</td>
<td>RelayL -- RelayR</td>
<td></td>
</tr>
</tbody>
</table>
ICE, Connectivity Checks

- Prune duplicates...
- Remove duplicates, retain highest priority duplicate

126-126  HostL  --  HostR
100-126  HostL  --  SrflxR
  0-126  RelayL  --  HostR
  0-126  HostL  --  RelayR
  0-100  RelayL  --  SrflxR
  0-0   RelayL  --  RelayR
ICE, Connectivity Checks

- Series of STUN *requests* and *responses* between these pairs
- Checks are paced
  - 1 every 20 ms
- Frozen Algorithm
- Normal checks (following prioritisation)
- Triggered checks (optimisation)
ICE, Connectivity Checks

- Series of STUN requests and responses between these pairs
ICE, Connectivity Checks

- Series of STUN *requests* and *responses* between these pairs
ICE, Connectivity Checks

• Series of STUN *requests* and *responses* between these pairs
ICE, Connectivity Checks

- Series of STUN requests and responses between these pairs
ICE, Connectivity Checks

- Series of STUN *requests* and *responses* between these pairs
ICE, Connectivity Checks

- Series of STUN *requests* and *responses* between these pairs
ICE, Connectivity Checks

- ... and host R does the same ...
ICE, Frozen Algorithm

- Generally, have multiple components (RTP, RTCP...), each with their own candidate sets
- ICE assumes that similar candidate pairs between components will exhibit similar characteristics
  - Initially all pairs are frozen; highest priority pair “unfrozen” and checked
  - If a STUN request comes in from one of the frozen pairs, unfreeze it such that it's the next check to be dispatched (triggered check)
ICE, Connectivity Checks

• 4-way handshake
ICE, Connectivity Checks

- Peer Reflexive candidate discovery:
- A STUN check through a symmetric NAT will reveal to the receiving peer a new candidate address
ICE, Coordination

- Signal completion (achieved directly between peers)
- Regular Nomination by controlling peer
  - Re-send a STUN check, with a flag set
- Aggressive nomination by controlling peer
  - Set flag in all STUN checks, such that the first working candidate is chosen
ICE, Communication

*joy*
Security Mechanisms

- TURN:
  - Long-term credentials
    - Digest challenge
- Connectivity checks:
  - Short-term credentials
    - Time-limited
MIMP: Mobile Internet Measurement Platform
MIMP: Mobile Internet Measurement Platform

• Aim is to support multiple different kinds of tests...
• Collect data from cellphones (etc...) in the real-world
• Server hardware located at Nokia; fit.nokia.com
MIMP: Mobile Internet Measurement Platform

• Downloadable client for Symbian
  • Updateable
  • Presents a list of tests to run
• Test-specific configuration via HTTP
• On test completion, submit results over HTTP
MIMP: Mobile Internet Measurement Platform
MIMP: ICE

- SIP server (OpenSER), STUN server/TURN relay (turnserver)
- ICE implementation: pjnath (part of the pjsip project)
  - http://pjsip.org/
- Symbian client grabs test configuration, e.g.,
  - SIP username & password
  - STUN/TURN server
    - SIP agent to contact (located on our machine)
    - Submits logged results to known location over HTTP
- Server side of comms also logs ICE interactions and submits
- Post-processing will take place to generate pretty pictures, graphs, etc
ICE: What don't we know?

• Actual quantifiable data on success rates for ICE
  • These protocols, or the ideas behind them, are being used in the real world, but perhaps they need tweaking
• Performance of connectivity checks
  • Analysis of quality of chosen candidates
• ... and then there's the possibility of collecting information on the type of NATs widely deployed in the Internet
Resources


Questions?
stephen.strowes@nokia.com