Peer-to-Peer Networks
Hole Punching
7th Week

Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer
Summer 2008
Peer-to-Peer Networks

NAT, PAT & Firewalls
Network Address Translation

- **Problem**
  - too few (e.g. one) IP addresses for too many hosts in a local network
  - hide hosts IP addresses from the outer world

- **Basic NAT (Static NAT)**
  - replace internal IP by an external IP

- **Hiding NAT**
  - = PAT (Port Address Translation)
  - = NAPT (Network Address Port Translation)
  - Socket pair (IP address and port number) are transformed to a single outside IP address

- **Hosts in local network cannot be addressed from outside**
DHCP Dynamic Host Configuration Protocol

- **DHCP (Dynamic Host Configuration Protocol)**
  - manual binding of MAC address
    - e.g. for servers
  - automatic mapping
    - fixed, yet not pre-configured
  - dynamic mapping
    - addresses may be reused

- **Integration of new hosts without configuration**
  - hosts fetches IP address from DHCP server
  - sever assigns address dynamically
  - when the hosts leaves the network the IP address may be reused by other hosts

- **P2P**
  - for dynamic mapping addresses must be refreshed
  - if a hosts tries to reuse an outdated address the DHCP server denies this request
  - problem: stealing of IP addresses
  - DHCP is good for anonymity
    - if the DHCP is safe
  - DHCP is bad for contacting peers in local networks
Firewalls

- **Types of Firewalls**
  - Host Firewall
  - Network Firewall

- **Network Firewall**
  - differentiates between
    - external net
      * Internet, hostile
    - internal net
      * LAN, trustworthy
    - demilitarized zone
      * servers reachable from the external net

- **Host Firewall**
  - e.g. personal firewall
  - controls the complete data traffic of a host
  - protection against attacks from outside and inside (trojans)

- **Methods**
  - Packet Filter
    - blocks ports and IP addresses
  - Content Filter
    - filters spam mails, viruses, ActiveX, JavaScript from html pages
  - Proxy
    - transparent (accessible and visible) hots
    - channels the communication and attacks to secured hosts
  - Stateful Inspection
    - observation of the state of a connection

- Firewalls can prevent Peer to Peer connections
  - on purpose or as a side effect
  - are treated here like NAT
Types of Firewalls & NATs (RFC 3489)

- **Open Internet**
  - addresses fully available

- **Firewall that blocks UDP**
  - no UDP traffic at all
  - hopeless, maybe TCP works?

- **Symmetric UDP Firewall**
  - allows UDP out
  - responses have to come back to the source of the request
  - like a symmetric NAT, but no translation

- **Full-cone NAT**
  - if an internal address is mapped to an external address all packets from will be sent through this address
  - External hosts can send packets to the external address which are delivered to the local address

- **Symmetric NAT**
  - Each internal request is mapped to a new port
  - Only a contacted host can send a message inside
    - on the very same external port arriving on the internal port

- **Restricted cone NAT**
  - Internal address are statically mapped to external addresses
  - All such UDP packets of one internal port use this external port
  - All external hosts can use this port to sent a packet to this host if they have received a packet recently from the same internal port (to any external port)

- **Port restricted cone NAT**
  - All UDP packets from one internal address use the same external port
  - External hosts must use this port to sent a packet to this host if they have received a packet recently from the same internal port to the same external port
Combination of NATs

Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel
Overcoming NAT by Relaying

- Relaying
  - use an open (non-NATed) server to relay all UDP or TCP connections
  - first both partners connect to the server
  - then, the server relays all messages
Connection Reversal

- If only one peer is behind NAT
  - then the peer behind NAT always starts connection
- Use a server to announce a request for connection reversal
  - periodic check for connection requests is necessary

Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

Peer-to-Peer-Networks
Summer 2008

Dienstag, 10. Juni 2008
Peer-to-Peer Networks

UDP Hole Punching
UDP Hole Punching


A does not know B’s address

**Algorithm**

- A contacts rendezvous server S and tells his local IP address
- S replies to A with a message containing
  - B’s public and private socket pairs
- A sends UDP packets to both of this addresses
  - and stays at the address which works
UDP Hole Punching

- Peers Behind a Common NAT
  - Rendezvous server is used to tell the local IP addresses
  - Test with local IP address establish the connections in the local net

Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel
Peer-to-Peer-Networks
Summer 2008

Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

13

Peer-to-Peer-Networks
Summer 2008

Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

13

Computer Networks and Telematics
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

Dienstag, 10. Juni 2008

UDP Hole Punching

› Peers Behind Different NATs
• Rendezvous server is used to tell the NAT IP addresses
• Test with NAT IP address establishes the connections
• Peers reuse the port from the Rendezvous server
Über UDP Hole Punching

- Peers Behind Multiple Levels of NAT
  - Rendezvous server is used to tell the NAT IP addresses
  - Test with NAT IP address establishes the connections
  - Relies on loopback translation of NAT C
Simple traversal of UDP over NATs (STUN)


- Client-Server Protocol
  - Uses open client to categorize the NAT router

- UDP connection can be established with open client
  - Tells both clients the external ports and one partner establishes the connection

- Works for Full Cone, Restricted Cone and Port Restricted Cone
  - Both clients behind NAT router can initialize the connection
  - The Rendezvous server has to transmit the external addresses

- Does not work for Symmetric NATs
STUN Test

- Client communicates to at least two open STUN server

from: http://en.wikipedia.org/wiki/STUN
Peer-to-Peer Networks

TCP Hole Punching
## TCP versus UDP Hole Punching

<table>
<thead>
<tr>
<th>Category</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection?</strong></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Symmetry</strong></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>client uses „connect“, server uses „accept“ or „listen“</td>
</tr>
<tr>
<td><strong>Acknowledgments</strong></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>must have the correct sequence numbers</td>
</tr>
</tbody>
</table>
P2P-NAT
Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

Prerequisite
- change kernel to allow to listen and connect TCP connections at the same time
- use a Rendezvous Server S
- Client A and client B have TCP sessions with S

P2P-NAT
- Client A asks S about B’s addresses
- Server S tells client A and client B the public and private addresses (IP-address and port number) of A and B
- From the same local TCP ports used to register with S
- A and B synchronously make outgoing connection attempts to the others’ public and private endpoints
- A and B
  - wait for outgoing attempts to succeed
  - wait for incoming connections to appear
  - if one outgoing connection attempt fails ("connection reset", "host unreachable") then the host retries after a short delay
- Use the first established connection
- When a TCP connection is made the hosts authenticate themselves
P2P-NAT
Peer-to-Peer Communication Across Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

Figure 7: Sockets versus Ports for TCP Hole Punching
Behavior for nice NAT-routers of A

- The NAT router of A learns of outgoing TCP-connection when A contacts B using the public address
  - A has punched a hole in its NAT

- A’s first attempts may bounce from B’s NAT router
- B’s connection attempt through A’s NAT hole is successful
- A is answering to B’s connection attempt
- B’s NAT router thinks that the connection is a standard client server

Some packets will be dropped by the NAT routers in any case

This connection attempt may also work if B has punched a hole in his NAT router before A

- The client with the weaker NAT router is the server in the TCP connection
P2P-Nat
Problems with Acks?

- Suppose A has punched the hole in his router
  
- A sends SYN-packet
  - but receives a SYN packet from B without Ack
    - so the first SYN from A must be ignored
  
- A replies with SYN-ACK to B
  
- B replies with ACK to A
  - all is fine then

- Alternatively:
  - A might create a new stream socket associated with B’s incoming connection start
    - a different stream socket from the socket that A hole punching TCP SYN message
    - this is regarded as a failed connection attempt
  - Also results in a working connection
P2P-NAT
The Lucky (?) Case

- What if both clients A and B succeed synchronously?
- When both clients answered to the SYN with a SYN-ACK
  - results in simultaneous TCP open
- Can result in the failure of the connection
  - depends on whether the TCP implementation accepts a simultaneous successful "accept()“ and "connect()“ operation
- Then, the TCP connection should work correctly
  - if the TCP implementation complies with RFC 793

- The TCP connection has been „magically“ created itself from the wire
  - out of nowhere two fitting SYN-ACKs have been created.
P2P-NAT Working Principle

(d) P2PNAT

Picture from
Characterization and Measurement of TCP Traversal through NATs and Firewalls
Saikat Guha, Paul Francis
# Success Rate of UDP Hole Punching and P2P-NAT (2005)

<table>
<thead>
<tr>
<th>NAT Hardware</th>
<th>UDP Hole Punching</th>
<th>Hairpin</th>
<th>TCP Hole Punching</th>
<th>Hairpin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linksys</td>
<td>45/46 (98%)</td>
<td>5/42 (12%)</td>
<td>33/38 (87%)</td>
<td>3/38 (8%)</td>
</tr>
<tr>
<td>Netgear</td>
<td>31/37 (84%)</td>
<td>3/35 (9%)</td>
<td>19/30 (63%)</td>
<td>0/30 (0%)</td>
</tr>
<tr>
<td>D-Link</td>
<td>16/21 (76%)</td>
<td>11/21 (52%)</td>
<td>9/19 (47%)</td>
<td>2/19 (11%)</td>
</tr>
<tr>
<td>Draytek</td>
<td>2/17 (12%)</td>
<td>3/12 (25%)</td>
<td>2/7 (29%)</td>
<td>0/7 (0%)</td>
</tr>
<tr>
<td>Belkin</td>
<td>14/14 (100%)</td>
<td>1/14 (7%)</td>
<td>11/11 (100%)</td>
<td>0/11 (0%)</td>
</tr>
<tr>
<td>Cisco</td>
<td>12/12 (100%)</td>
<td>3/9 (33%)</td>
<td>6/7 (86%)</td>
<td>2/7 (29%)</td>
</tr>
<tr>
<td>SMC</td>
<td>12/12 (100%)</td>
<td>3/10 (30%)</td>
<td>8/9 (89%)</td>
<td>2/9 (22%)</td>
</tr>
<tr>
<td>ZyXEL</td>
<td>7/9 (78%)</td>
<td>1/8 (13%)</td>
<td>0/7 (0%)</td>
<td>0/7 (0%)</td>
</tr>
<tr>
<td>3Com</td>
<td>7/7 (100%)</td>
<td>1/7 (14%)</td>
<td>5/6 (83%)</td>
<td>0/6 (0%)</td>
</tr>
<tr>
<td>OS-based NAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>31/33 (94%)</td>
<td>11/32 (34%)</td>
<td>16/31 (52%)</td>
<td>28/31 (90%)</td>
</tr>
<tr>
<td>Linux</td>
<td>26/32 (81%)</td>
<td>3/25 (12%)</td>
<td>16/24 (67%)</td>
<td>2/24 (8%)</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>7/9 (78%)</td>
<td>3/6 (50%)</td>
<td>2/3 (67%)</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>All Vendors</td>
<td>310/380 (82%)</td>
<td>80/335 (24%)</td>
<td>184/286 (64%)</td>
<td>37/286 (13%)</td>
</tr>
</tbody>
</table>

Table 1: User Reports of NAT Support for UDP and TCP Hole Punching

**Peer-to-Peer Communication Across Network Address Translators**

Bryan Ford, Pyda Srisuresh, Dan Kegel
TCP Hole Punching with Small TTL

- NAT Servers can be punched with TCP Sync packets of small TTL
  - message passes NAT server
  - listening to outgoing messages help to learn the Sequence Number

- Technique used by
  - STUNT#1, #2
  - NATBlaster
Both endpoints produce a SYN packet with small TTL
  • Packet passes NAT-router, yet does not reach target
Both clients learn their own (!) sequence number
STUNT (Rendezvous) server produces a spoofed SYNACK
  • with correct sequence number to both clients
Both clients respond with ACK
Hopefully, connection is established
Problems:
  • Choice of TTL. Not possible if the two outermost NATs share an interface
  • ICMP-packet can be interpreted as fatal error
  • NAT may change the sequence number, spoofed SYNACK might be „out of window“
  • Third-party spoofer is necessary

---

Endpoints A produce a SYN packet with small TTL
- Packet passes NAT-router, yet does not reach target

Client A aborts attempt connect
- accepts inbound connections

Client B
- learns address from Rendezvous server
- initiates regular connection to A

Client A answers with SYNACK
- Hopefully, connection is established

Problems:
- Choice of TTL.
- ICMP-packet must not be interpreted as fatal error
- NAT must accept an inbound SYN following an outbound SYN
  - unusual situation

Both endpoints produce low TTL SYN-packets
- passes NAT router, but does not reach other NAT router

Learn sequence number for own connection
- exchange this information using Rendezvous server

Both endpoints produce SYN-ACK packets
- Both endpoints answer with ACKs
- Connection established

Problems
- Choice of TTL
- NATs must ignore ICMP-packet
- NAT may change sequence numbers
- NAT must allow symmetric SYN-Acks after own SYN packet
  - unusual

---

## OS Issues of TCP Hole Punching

<table>
<thead>
<tr>
<th>Approach</th>
<th>NAT/Network Issues</th>
<th>Linux Issues</th>
<th>Windows Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUNT #1</td>
<td>• Determining TTL</td>
<td>• Superuser priv.</td>
<td>• Superuser priv.</td>
</tr>
<tr>
<td></td>
<td>• ICMP error</td>
<td></td>
<td>• Setting TTL</td>
</tr>
<tr>
<td></td>
<td>• TCP Seq# changes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• IP Address Spoofing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STUNT #2</td>
<td>• Determining TTL</td>
<td></td>
<td>• Setting TTL</td>
</tr>
<tr>
<td></td>
<td>• ICMP error</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SYN-out SYN-in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NATBlaster</td>
<td>• Determining TTL</td>
<td>• Superuser priv.</td>
<td>• Superuser priv.</td>
</tr>
<tr>
<td></td>
<td>• ICMP error</td>
<td></td>
<td>• Setting TTL</td>
</tr>
<tr>
<td></td>
<td>• TCP Seq# changes</td>
<td></td>
<td>• RAW sockets (post WinXP SP2)</td>
</tr>
<tr>
<td></td>
<td>• SYN-out SYNACK-out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2PNAT</td>
<td>• TCP simultaneous open</td>
<td></td>
<td>• TCP simultaneous open (pre WinXP SP2)</td>
</tr>
<tr>
<td></td>
<td>• Packet flood</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STUNT #1 no-TTL</td>
<td>• RST error</td>
<td>• Superuser priv.</td>
<td>• Superuser priv.</td>
</tr>
<tr>
<td></td>
<td>• TCP Seq# changes</td>
<td></td>
<td>• TCP simultaneous open (pre WinXP SP2)</td>
</tr>
<tr>
<td></td>
<td>• Spoofing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STUNT #2 no-TTL</td>
<td>• RST error</td>
<td>• Superuser priv.</td>
<td>• TCP simultaneous open (pre WinXP SP2)</td>
</tr>
<tr>
<td></td>
<td>• SYN-out SYN-in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NATBlaster no-TTL</td>
<td>• RST error</td>
<td>• Superuser priv.</td>
<td>• Superuser priv.</td>
</tr>
<tr>
<td></td>
<td>• TCP Seq# changes</td>
<td></td>
<td>• RAW sockets (post WinXP SP2)</td>
</tr>
<tr>
<td></td>
<td>• SYN-out SYNACK-out</td>
<td></td>
<td>• TCP simultaneous open (pre WinXP SP2)</td>
</tr>
</tbody>
</table>

*from* Characterization and Measurement of TCP Traversal through NATs and Firewalls, Saikat Guha, Paul Francis
Peer-to-Peer-Networks
Summer 2008

Port Prediction

- NAT router changes port addresses for incoming connections
- A knows the type of NAT
  - learns the mapping from the Rendezvous (STUNT) server
  - predicts its mapping
- B also predicts his mapping
- Both clients send SYN packets to the predicted ports
- Usually, NAT servers can be very well predicted, e.g.
  - outgoing port is 4901.
  - then the incoming port is 4902
    - if 4902 is not used, then it is 4903
    * and so on....

Figure 6: Port-prediction in TCP NAT-Traversal approaches. from Characterization and Measurement of TCP Traversal through NATs and Firewalls, Saikat Guha, Paul Francis

Dienstag, 10. Juni 2008
How Skype Punches Holes

- An Experimental Study of the Skype Peer-to-Peer VoIP System, Saikat Guha, Neil Daswani, Ravi Jain
  - Skype does not publish its technique
  - Yet, behavior can be easily tracked

- Techniques
  - Rendezvous Server
  - UDP Hole Punching
  - Port scans/prediction
  - Fallback: UDP Relay Server
    - success rate of Skype very high, seldomly used
Peer-to-Peer Networks
End of 7th Week