

Teredo, the Internet shipworm

Peer-to-peer NAT traversal

Rémi Denis-Courmont

Nokia Devices R&D
Maemo Software

Future Internet pizza - October 15th 2008

Outline

- 1 A brief history of NATs
- 2 NATs today
- 3 Teredo

Engineering advisory

Explicit content

The opinion expressed in this slideset are my personal own.

Outline

- 1 A brief history of NATs
- 2 NATs today
- 3 Teredo

Early consumer Internet access

In the old days. . .

- Initially, one IP address per host.
- Dial-up Internet connectivity.

Early consumer Internet access

In the old days. . .

- Initially, one IP address per host.
- Dial-up Internet connectivity.
- Slow.
- ISP contractual terms restricted to one host at a time.

Broadband and connection sharing

Emergence of the CPE concept

- Broadband "fast" connectivity
 - DSL,
 - cable (DOCSIS),
 - even fibre for the luckiest (FTTH).

Broadband and connection sharing

Emergence of the CPE concept

- Broadband "fast" connectivity
 - DSL,
 - cable (DOCSIS),
 - even fibre for the luckiest (FTTH).
- Local area networking:
 - several computers at home
 - built-in network interface cards
 - wireless (802.11) networking
 - cheap (or rented) NAT/routers

Broadband and connection sharing

Emergence of the CPE concept

- Broadband "fast" connectivity
 - DSL,
 - cable (DOCSIS),
 - even fibre for the luckiest (FTTH).
- Local area networking:
 - several computers at home
 - built-in network interface cards
 - wireless (802.11) networking
 - cheap (or rented) NAT/routers

⇒ Only one IP address per home.

Uses and abuses of NATs

Convenient hack

- easy upgrade path for ISP and users:
 - plug-in the WAN port and surf the Web!
 - no routing and prefix provisioning for the ISP

Uses and abuses of NATs

Convenient hack

- easy upgrade path for ISP and users:
 - plug-in the WAN port and surf the Web!
 - no routing and prefix provisioning for the ISP
- not enough IP addresses anyway
- inbound firewall against Internet worms
- topology hiding, privacy(?)

Uses and abuses of NATs

Convenient hack

- easy upgrade path for ISP and users:
 - plug-in the WAN port and surf the Web!
 - no routing and prefix provisioning for the ISP
- not enough IP addresses anyway
- inbound firewall against Internet worms
- topology hiding, privacy(?)

⇐ And it works fine for web & email.

Outline

- 1 A brief history of NATs
- 2 NATs today
- 3 Teredo

What works?

The client-server model

- What was there when NAT was invented!

What works?

The client-server model

- What was there when NAT was invented!
- Client-server protocols:
 - domain name service (DNS)
 - web (HTTP),
 - email (SMTP, POP, IMAP),
 - instant messaging (MSNP, XMPP...),
 - remote access (SSH, RDP...),
 - file sharing (FTP, SMB/CIFS),
 - games...

What works?

The client-server model

- What was there when NAT was invented!
- Client-server protocols:
 - domain name service (DNS)
 - web (HTTP),
 - email (SMTP, POP, IMAP),
 - instant messaging (MSNP, XMPP...),
 - remote access (SSH, RDP...),
 - file sharing (FTP, SMB/CIFS),
 - games...

Connection from client to server.

The two connectivity models

- Client-server model:
Asymmetric network
 - Client at home, server in rack
 - Clients = second class Internet nodes
- Peer-to-peer model:
Direct person to person communications
 - Peer-to-peer file sharing
 - Audio/video communications (not just IM/text)
 - Mobile to home, remote home access

The two connectivity models

- Client-server model:
Asymmetric network
 - Client at home, server in rack
 - Clients = second class Internet nodes
- Peer-to-peer model:
Direct person to person communications
 - Peer-to-peer file sharing
 - Audio/video communications (not just IM/text)
 - Mobile to home, remote home access

Connection from "client" to another "client"!

NAT operations

Outgoing packet

NAT maintains a mapping table:
client (*IP, port*) \leftrightarrow NAT *port*

NAT operations

Outgoing packet

NAT maintains a mapping table:

client (*IP, port*) \leftrightarrow NAT *port*

- Look-up *source = client (IP, port)* in the table.
- If no match, allocate a new NAT *port*.
- Replace the source port with the matching NAT port.
- Replace the source address with the NAT address.
- Send the packet to the Internet.

NAT operations (cont'd)

Incoming packet

NAT remembers the mapping table:

client (*IP, port*) \leftrightarrow NAT *port*

- Look-up *destination = NAT port* in the table.
- If no match, *discard the packet*. Otherwise...
- Replace the destination port with the client port.
- Replace the destination address with the client address.
- Send the packet to the internal network.

Outline

- 1 A brief history of NATs
- 2 NATs today
- 3 Teredo**

What is it?

Depending on the perspective:

- An automatic IPv6 transition mechanism.
- A (pseudo-)tunneling IPv6-over-UDP-over-IPv4 protocol.
- An *overlay* peer-to-peer network on top of IPv4.

Timeline

- 2001: first draft published (Microsoft).

Timeline

- 2001: first draft published (Microsoft).
- 2003: peer-to-peer kit for Windows XP SP1.
- 2003: FreeBSD implementation by 6WIND.

Timeline

- 2001: first draft published (Microsoft).
- 2003: peer-to-peer kit for Windows XP SP1.
- 2003: FreeBSD implementation by 6WIND.
- 2004: Linux implementation by yours.
- 2004: IPv6 stack for Windows XP SP2.

Timeline

- 2001: first draft published (Microsoft).
- 2003: peer-to-peer kit for Windows XP SP1.
- 2003: FreeBSD implementation by 6WIND.
- 2004: Linux implementation by yours.
- 2004: IPv6 stack for Windows XP SP2.
- 2006: IETF RFC4380 standardized.
- 2006: IPv6 on-by-default stack for Windows Vista.

Timeline

- 2001: first draft published (Microsoft).
- 2003: peer-to-peer kit for Windows XP SP1.
- 2003: FreeBSD implementation by 6WIND.
- 2004: Linux implementation by yours.
- 2004: IPv6 stack for Windows XP SP2.
- 2006: IETF RFC4380 standardized.
- 2006: IPv6 on-by-default stack for Windows Vista.
- 2008: draft protocol updates (Microsoft).
- 2008: some BitTorrent client starts using it.

Teredo addressing

- Ask an outside server for your NAT IP and port.
- Determine your Teredo IPv6 address (128-bits):
2001 : 0000 : *serverIP* : *flags* : *port* : *IP*
- "Refresh" the NAT mapping regularly.

Establishing communication

- Assume you know the other IPv6 address.
- Infer peer's server, NAT IP and port from its address.

Establishing communication

- Assume you know the other IPv6 address.
- Infer peer's server, NAT IP and port from its address.
- Send *kamikaze* "bubble" to the peer NAT IP/port
→ allocate a mapping on your NAT.

Establishing communication

- Assume you know the other IPv6 address.
- Infer peer's server, NAT IP and port from its address.
- Send *kamikaze* "bubble" to the peer NAT IP/port
→ allocate a mapping on your NAT.
- Send *out-of-band* "bubble" through the peer server.
- Server can forward the bubble to the peer, because the peer is communicating with the server.

Establishing communication

- Assume you know the other IPv6 address.
- Infer peer's server, NAT IP and port from its address.
- Send *kamikaze* "bubble" to the peer NAT IP/port
→ allocate a mapping on your NAT.
- Send *out-of-band* "bubble" through the peer server.
- Server can forward the bubble to the peer, because the peer is communicating with the server.
- Wait for "bubble" from the peer NAT IP/port. → allocate a mapping on the peer NAT.

Establishing communication

- Assume you know the other IPv6 address.
- Infer peer's server, NAT IP and port from its address.
- Send *kamikaze* "bubble" to the peer NAT IP/port
→ allocate a mapping on your NAT.
- Send *out-of-band* "bubble" through the peer server.
- Server can forward the bubble to the peer, because the peer is communicating with the server.
- Wait for "bubble" from the peer NAT IP/port. → allocate a mapping on the peer NAT.
- Exchange any UDP packet with the peer, via its NAT IP/port.

Programmer's perspective

- In theory, could be implemented in every application.
- In practice, background operating system service.

Programmer's perspective

- In theory, could be implemented in every application.
- In practice, background operating system service.
- An IPv6-capable (virtual) interface (a.k.a. tunnel).
- Normal INET6 sockets.

Complications

Teredo predicts NAT address & port.

- IP prediction almost always works.
- NAT port prediction often works, not always :(
- Teredo disables itself if NAT problem detected.

¹Apple NAT-PMP could be used all the same

Complications

Teredo predicts NAT address & port.

- IP prediction almost always works.
- NAT port prediction often works, not always :(
- Teredo disables itself if NAT problem detected.

To improve the situation:

- UPnP IGD: try to control the NAT directly¹.
- Multicast: detect Teredo nodes on the local network.

¹Apple NAT-PMP could be used all the same

Relaying

When everything else fails, use Teredo *relays*:

- Dedicated servers.
- Relay: Teredo ↔ rest of the IPv6 Internet.
- Questionable deployment model: who pays for the bandwidth?

Comparison with ICE

	TEREDO	ICE
Lower-layer protocol	UDP/IPv4	UDP/IPv4
Payload format	IPv6	RTP (& STUN)
Higher-layer	any IP protocol	one RTP session
Implementation	the IP stack	each application
Signaling	Teredo server	SIP proxy
Discovery	Teredo server	STUN server
Fallback relay	Teredo relay	TURN server
Path MTU discovery	Not currently	Not yet
Mobile IPv6, HIP	Yes	W.I.P.
TCP, DCCP, other	Yes, Yes, Yes	Not really, No, No
UDP-Lite	Not really	No
HTTP tunneling	No	No

References

- Teredo overview
<http://www.microsoft.com/technet/network/ipv6/teredo.msp>
- Miredo, Teredo for Linux <http://www.remlab.net/miredo/>
- RFC4380 "Teredo"
- RFC3489 "STUN"
- draft-ietf-mmusic-ice "ICE", draft-ietf-mmusic-ice-tcp

Any questions?