Teredo: IPv6 through NAT, over UDP

Christian Huitema
Architect
Windows Networking & Communications
Microsoft Corporation
Presenting Teredo

- Model of operation
- Adapting to various NAT forms
- Operational considerations
- Security considerations
- What is in a name
Teredo: IPv6 behind a NAT

- **Teredo server:**
  - Helps host discover its “mapped” address
  - Provides prefix: xxxx:IPv4:port::/64
  - Example: xxxx:4050:607:401::/64

- **Teredo relay:**
  - Advertises xxxx://16
  - Tunnel over UDP to “IPv4:port”
  - NAT will relay to host

- **Between host:**
  - First packet through server,
  - Use “bubbles” to pierce the NAT, enable transmission
Teredo objects & entities

- **Client**: the node behind a NAT
- **Server**: helps the client connect
- **Relay**: forwards IPv6 packets to clients
- **Teredo IPv6 prefix**: xx:://n (TBD IANA)
  - Used to construct all Teredo addresses
- **Teredo address prefix**: xx:IPv4:port::/m
  - Embeds the “mapped address & port” of the client
- **Teredo IPv4 anycast address**: x.y.z.t
  - Used by relays and servers
- **Teredo UDP port**: pppp
  - Used by relays and servers
Qualification Procedure

- **Client sends “Router Solicit”:**
  - Over UDP
  - Fixed “anycast” address of server

- **Server replies with “Router advertisement”:**
  - Prefix includes “mapped IP address” and “mapped port.”
  - Example: xx:4005:607:401::

- **Client is qualified!**
  - Address: prefix + Identifier
Transmission between Teredo & regular IPv6 node

- **Teredo to IPv6 (A-B)**
  - A sends to server,
  - Server relays to IPv6.

- **IPv6 to relay (B-A)**
  - B sends to A (IPv6)
  - Packet routed to relay
  - Relay sends to A (UDP), same source address, port as server
Transmission between Teredo nodes

- **First packet, A → B**
  - A sends bubble to B
  - Packet through server,
  - Server relays with “anycast” address, use existing hole

- **Reply, B → A**
  - If bubble received, send direct to A
  - Else, send through S, send bubble to A

- **Follow on packets**
  - If bubble or direct packet received, direct path.
  - Else, through server.
Transmission between Local Teredo nodes

- Qualification:
  - A & B get address from server S

- Advertisement
  - A & B send multicast bubble, advertise their local address
  - Multicast bubbles are cached

- Direct transmission
  - Packets are sent directly, over the local network, using UDP encapsulation

- Limitation
  - Single link!
Adapting to all NAT forms

- Four forms of NAT
  - Cone NAT
  - Restricted Cone
  - Port Restricted Cone
  - Symmetric

- Additional parameter: delay
- UDP packet creates a mapping in NAT
  - Inside: 10.0.0.2:3456
  - Outside: 64.5.6.7:1025
- Target (B) can reply
- Third party (C) can also reply
Restricted Cone NAT

- UDP packet creates a mapping in NAT
- Target (B) can reply
- Random third party (C) cannot reply...
- Traffic to other party use the same mapping
- If spoken to, can respond
Port Restricted Cone NAT

- Same behavior as restricted cone, with one difference, more restrictive:
  - If traffic send to "port-x", return is only authorized from same port.
Symmetric NAT

- Mapping of internal port varies as a function of target
- Generally coupled with “port restricted” behavior.
Design goal: passing all types of NAT

- Use a single address & port for servers and relays:
  - Only one hole to maintain in the NAT
  - See maintenance procedure, next slide

- Wait for receiving a “bubble” or a “direct packet” before sending on direct path
  - 1st packet goes through server for all NAT
  - 2nd packet goes direct in 40% of NAT (open)
  - 3rd packet goes direct in 95% of NAT (protected)
  - Keep using a server as relay in 5% of cases (weird)
Maintenance procedure

- **NAT mapping will time out after “some period”**
  - Maintain a timer: last packet from “server”
  - Refresh the mapping if timer elapses
  - Detect possible change of mapping during refresh
- **Period is variable**
  - Assume 30 seconds initially
- **Use secondary port to test the timer:**
  - Get mapping for secondary port
  - Test packet from primary port to secondary through server after “candidate timer”
  - If packet received: try larger value (2 minute max)
  - Else: try smaller, or stop.
Operation issues: routing

- **Teredo network determined by**
  - Teredo IPv6 address prefix,
  - Teredo IPv4 anycast address,
  - Teredo UDP port

- **Restriction on IPv4 anycast**
  - Must be “topologically correct”
  - must advertise “reachability” to all

- **Restriction on IPv6 source**

- **Option: run separate networks**
Security issues

- Big concern: address spoofing
  - Relays can be abused to source “funny” traffic, hide the source
  - Teredo address only as “proven” as IPv4 source address (i.e. not much)

- Mitigating factors
  - Teredo enables IPSEC end-to-end
  - Teredo traffic to third parties can easily be filtered-out, preventing DDOS attack
What is in a name?

- **Teredo Navalis:**
  - Wood boring salt water mollusk
  - 10-15 cm in length, 10 mm in diameter

- Looks nasty, but
  - “the animal only survives in relatively clean and unpolluted water; its recent comeback in several Northern American harbors is a testimony to their newly retrieved cleanliness”
Teredo: what is the timeline?

- Spec passed WG last call, received IESG comments
  - Expect RFC in 2002
- Development of code in Windows XP
  - Starting now; relatively simple.
  - Availability as some form of Windows XP update
- Other developments
  - Expect 6 months after RFC for routers (Cisco),
  - Maybe some NAT
- Deployment of servers
  - Deploy Windows based test server by December 2001 in Redmond (done)
  - Test by ISP in early 2002 – hopefully!
Where do you want to go today?