

Internet Technology

15. VoIP, NAT Traversal, and auto configuration

Paul Krzyzanowski
Rutgers University
Spring 2016

1

Session Initiation Protocol (SIP)

- Dominant protocol for Voice over IP (VoIP): RFC 3261
- Allows a call to be established between multiple parties
 - Notify a callee of a call request
 - Agree on media encodings
 - Allow a participant to end the call
 - Determine IP address of callee
 - No assumption on the callee having a fixed IP address
 - Add new media streams, change encoding, add/drop participants
- Messages are HTTP style (line-oriented text) using UDP or TCP



2

Proxies

- SIP proxy server
 - Helps route requests
 - Forwards requests to one or more destinations and sends responses to the requester
 - Contacts remote registrar to look up addresses
 - Often run on the same server as a registrar
- Usually a proxy at each SIP domain

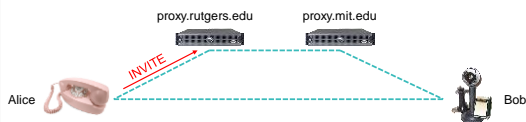
3

Registration

- A user's SIP address is an IP address & port number
 - In many cases, this changes over time
- Registration
 - When a phone is switched on (or phone software is run)
 - A registration process takes place
 - Registrations expire, so re-register periodically
- Location Server
 - Stores a mapping between the user's address and the address of their phone
 - user's address = Address of Record (AOR): sip:alice@sip.rutgers.edu
- SIP Registrar:
 - Accepts REGISTER requests and interacts with the Location Server
- SIP proxy, registrar, & location server normally run on the same system

4

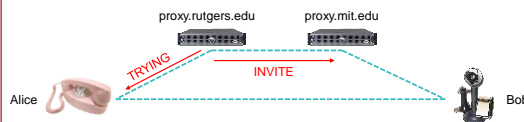
SIP Example



- Alice wants to call bob@sip.mit.edu
- She sends an INVITE message to her proxy server
 - HTTP-style
 - Identifies destination: Bob (bob@sip.mit.edu)
 - Specifies:
 - Alice's current IP address
 - Media type (e.g., PCM-encoded audio via RTP)
 - Port on which she'd like to receive the message

5

SIP Example



- Alice's SIP proxy server needs to look up bob@sip.mit.edu
 - Uses DNS to look up Bob's SIP server (NAPTR and/or SVR records)
 - Forwards the Alice's INVITE to Bob's SIP proxy
 - Tells Alice that it's TRYING to contact the party

NAPTR = Name Authority Pointer
– designed to get a list of protocols and regular expression rewrite rule to create a SIP URN

SVR = Service Record – designed to map service names to hostname:port

6

SIP Example

- Routing
 - SIP INVITE requests are sent from proxy to proxy until it reaches one that knows the location of the callee
 - A Proxy may respond with a REDIRECT message

SIP Example

- Bob's proxy server
 - Forwards the INVITE to Bob's phone
 - Tells Alice's proxy server that it's trying to reach Bob

SIP Example

- Bob's phone gets the INVITE message
 - Starts ringing
 - Sends RINGING response

SIP Example

- Bob can accept or decline the call
 - If he accepts it, the INVITE is acknowledged with a 200 OK
 - INVITE feedback is propagated back to Alice

SIP Example

- Now Alice & Bob talk point-to-point
 - Alice sends an ACK to confirm setup
 - Both sides exchange media streams (usually RTP)

SIP Example

- To disconnect, one party sends a BYE message
- The other side confirms with a 200 OK
- SIP is an **out-of-band protocol**
 - SIP messages are sent on different sockets than media data
 - All messages are acknowledged, so either TCP or UDP can be used

NAT Traversal

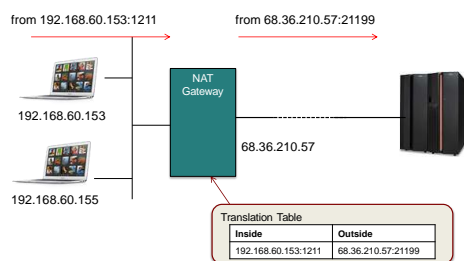
13

NAT traversal & why do we need it?

- Remember NAT?
 - Private IP addresses
 - NAT gateway (usually on a gateway router)
 - Translates between internal addresses/ports & external ones
- It's awesome!
 - Cut down on lots of wasted addresses – usually, you need just one
- But it breaks end-to-end connectivity!
 - What if you want to contact a service behind NAT?
 - Consider two VoIP clients that want to communicate
 - *No foolproof solution*

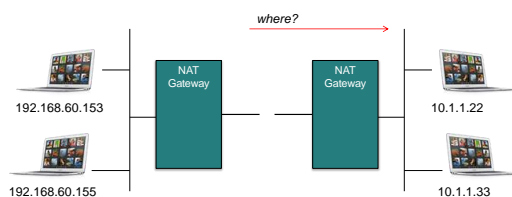
14

NAT: This is easy



15

NAT: This is tricky



16

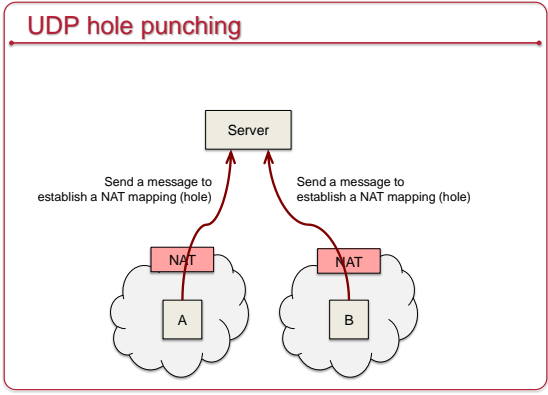
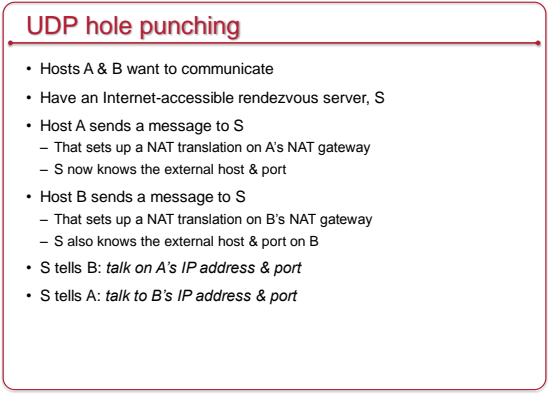
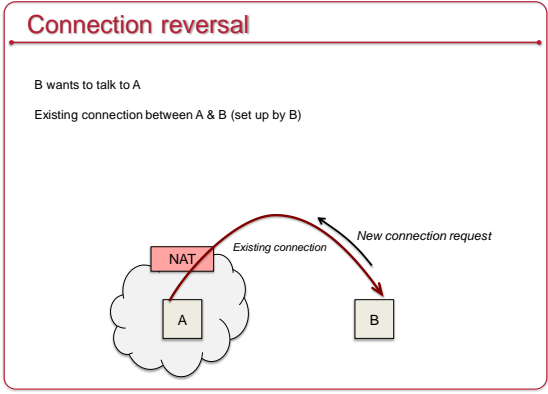
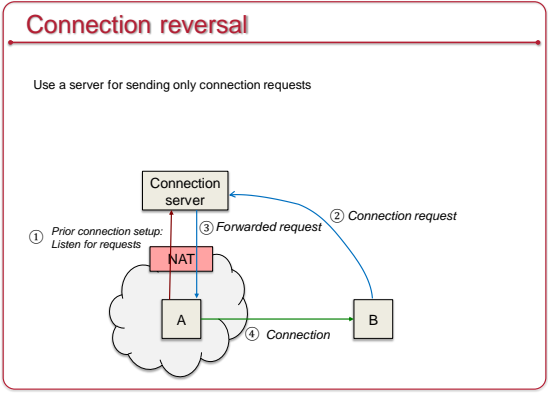
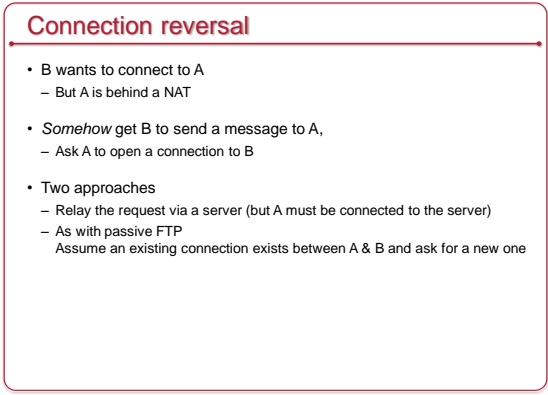
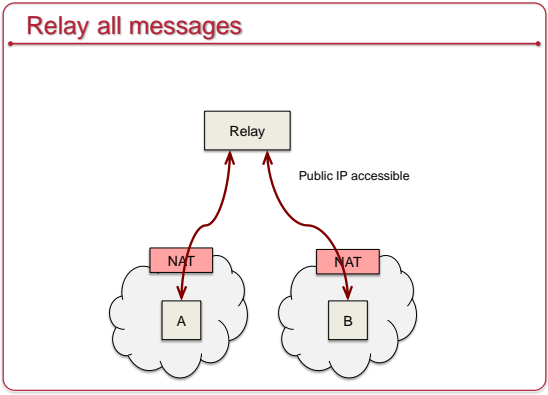
NAT Traversal Techniques

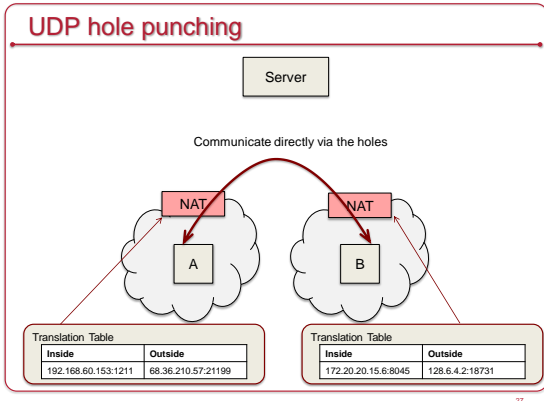
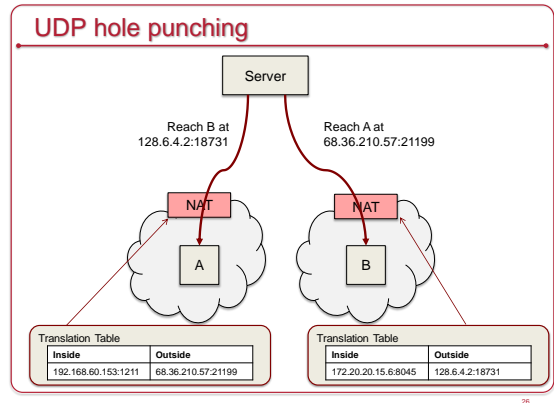
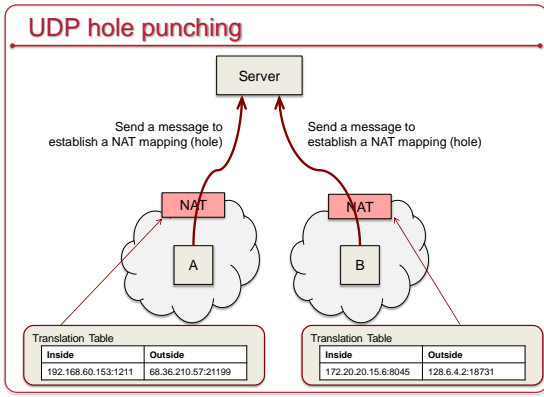
17

Relay all messages

- Hosts A & B want to communicate
- Have an Internet-accessible proxy, P
- A connects to P and waits for messages on the connection
- B talks to P; P relays messages to A
- Most reliable but not very efficient
 - Extra message relaying
 - Additional protocols needed (e.g., B needs to state what it wants)
 - Proxy can become a point of congestion (network links & CPU)

18






- ### TCP hole punching
- Same principle (tell other host of your address:port) – BUT
 - Use **TCP Simultaneous Open**
 - Both hosts will try to connect to each other
 - Each NAT creates a translation rule
 - At least one of the SYN messages during connection set up will go through the NAT translation on the other side
 - The remote side will send a SYN-ACK
 - Need to re-use the same port # that the remote side knows about
 - Socket option to reuse an address: `SO_REUSEADDR`
 - Not guaranteed to work with all NAT systems

NAT Traversal Protocols

- ### STUN
- **Session Traversal Utilities for NAT**; RFC 5389
 - Allows clients to discover whether they are in a NAT environment
 - Discover public IP address
 - Send a message to a **STUN server** on the Internet
 - STUN server returns the source IP address and port number
 - A client can share this external address/port
 - If both peers are behind NAT, they will need to find a way to share this information
- Hole punching

TURN

- Traversal Using Relays around NAT; RFC 5766
 - Protocol that uses a relay server

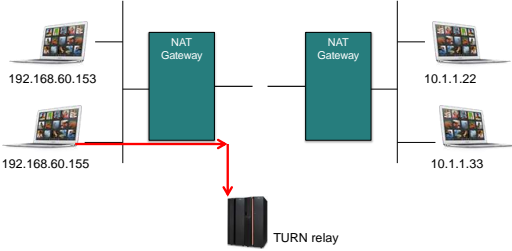


32

TURN

TURN server: Relay-based protocol

- .155 connects to a TURN server
- Informs the server which locations it should accept packets from
- Gets an IP address & port allocated by the TURN server to use as a relay

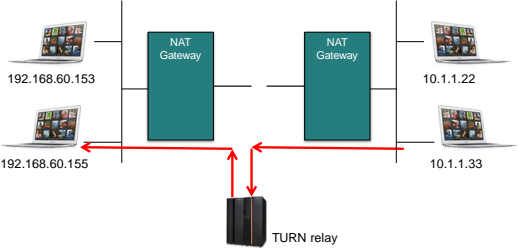


33

TURN

TURN server: STUN server with relay capabilities

- .33 contacts the TURN relay, which relays its external host:port to .155



34

ICE

- **Interactive Connectivity Establishment**; RFC 5245
 - Coordinates whether to use STUN or TURN
 - Protocol to negotiate NAT traversal
 - Discover presence of NAT on either side
 - Exchange information
 - Discover how to establish a connection
 - Choose STUN or TURN
 - Extension to SIP (but can be used by other protocols)

35

Zero Configuration Networking

36

Network Configuration

- Normally
 - DHCP server to get an IP address (and subnet mask, gateway)
 - DNS for looking up names
- What if we don't have these available?
 - Use **IP Link-Local Addresses**
 - Goal: each device gets an IP address that is unique in the LAN
 - These are non-routable (not valid on the outside Internet)

37

IPv4: Link-Local Addresses

- 169.254.0.0/16 block – reserved for link-local addresses
- Pick a random address in the 169.254.0.0/16 range
- Use ARP to see if someone else also has it
- If so, try again

IPv6 Stateless Address Autoconfiguration (SLAAC)

- **Link-local addresses**
 - Combination of address prefix & interface ID
 - Use `fe80::/64` block as an address prefix
 - Hosts generate a unique 64-bit interface ID from the MAC address
 - Run **Duplicate Address Detection** to ensure address is unique
 - Send a *Neighbor Solicitation* request (IPv6's version of ARP)
 - If someone else has it, fail: admin intervention required.
 - Unlike IPv4, every host should have a link-local address even if they have a routable address
- **Routable addresses**
 - Routers advertise prefixes that identify the link's subnet
 - Use this prefix instead of `fe80`
 - SLAAC can behave like a simplified DHCP
 - Good if just getting a unique, routable address is sufficient

Multicast DNS

- RFC 6762, used by Apple Bonjour and Android ≥ 4.1
- Translate between names and IP addresses without a DNS server
 - **Multicast DNS**: Use IP multicast for DNS queries
 - Each computer stores its own list of resource records
 - Sort of like ARP for DNS
 - Handles queries for the `.local` top-level domain (by default)
 - Runs its own mini DNS server: `mDNSResponder`

Also see Microsoft's [Link-local Multicast Name Resolution \(LLMNR\)](#), RFC 4795

Multicast DNS for service discovery

- Locate or advertise services without using a directory server
- Example, Apple DNS-based Service Discovery: DNS-SD (RFC 6763)
 - Use DNS services (DNS or multicast DNS)
 - Structured Instance Names
 - **SRV** record: query for *Instance.Service.Domain* gives target IP, port
 - **TXT** record with same name: extra info provided as key/value pairs
 - **PTR** record: service type to see all instances of the service
 - Also
 - Simple Service Discovery Protocol (SSDP; part of UPnP)
 - Service Location Protocol (SLP)

SRV record example

- Example DNS SRV record


```
myprinter._printer._tcp.local. 120 IN SRV 0 0 5432 myserver.local.
```
- DNS TXT record
 - May contain additional information
 - Example:
 - Different print queues for printer services on the same IP address
 - Information is application-specific
- PTR record


```
_printer._tcp.local. 28800 PTR myprinter._printer._tcp.local.
```

 - Allows one to query DNS for all services of type `_printer`.

Apple Bonjour initial steps

- New device starts up
 - **Is there a DHCP server?**
 - If yes, get IP address and routing info
 - If no, pick an address in the link-local (zeroconf) range: 169.254.0.0/16
 - Test the address and claim it if nobody responds
 - **Start up Multicast DNS responder**
 - Requests a chosen hostname
 - Multicasts query to see if it's taken
 - Claims it if not taken
 - **Start up service (get port)**
 - **Publish service** (friendly name, service name, address, port)
 - Create SRV record `friendly_name.service_name._tcp.local` that points to the hostname and port for the service
 - Create PTR record `service_name._tcp.local`

The end

44