15-441 15-641 Computer Networking

Lecture 6 – IP in Practice

Peter Steenkiste

Fall 2016

www.cs.cmu.edu/~prs/15-441-F16

Outline

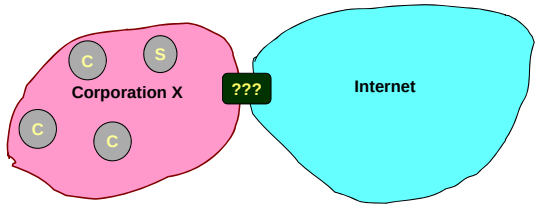
- Network address translation
- Address resolution protocols
- Tunnels

Altering the Addressing Model

- Original IP Model: Every host has unique IP address
- This has very attractive properties …
  - Any host can communicate with any other host
  - Any host can act as a server
    - Just need to know host ID and port number
- … but the system is open – complicates security
  - Any host can attack any other host
  - It is easy to forge packets
    - Use invalid source address
- … and it places pressure on the address space
  - Every host requires "public" IP address

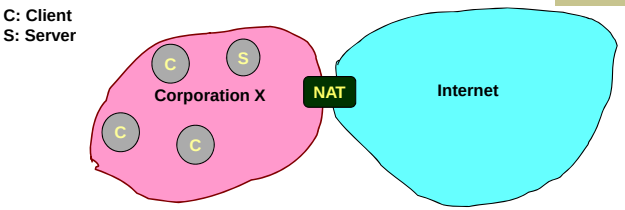
Challenges When Connecting to Public Internet

C: Client
S: Server



- Not enough IP addresses for every host in organization
  - Increasingly hard to get large address blocks
- Security
  - Don't want every machine in organization known to outside world
  - Want to control or monitor traffic in / out of organization

But not All Hosts are Equal!

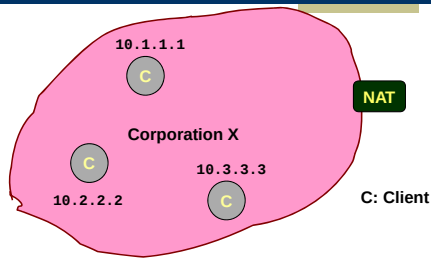


- Most machines within organization are used by individuals
  - For most applications, they act as clients
- Only a small number of machines act as servers for the entire organization
  - E.g., mail server, web, ..
  - All traffic to outside passes through firewall

(Most) machines within organization do not need public IP addresses!

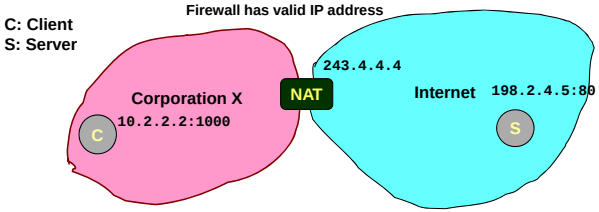
Reducing Address Use: Network Address Translation

- Within organization: assign each host a private IP address
  - IP addresses blocks 10/8 & 192.168/16 are set aside for this
  - Route within organization by IP protocol
  - Can do subnetting, ..



- The NAT translates between public and private IP addresses as packets travel to/from the public Internet
  - It does not let any packets from internal nodes “escape”
  - Outside world does not need to know about internal addresses

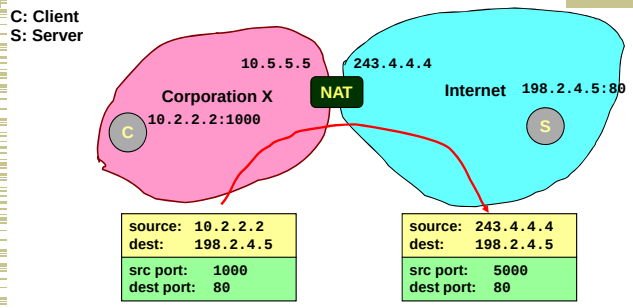
NAT: Opening Client Connection



- Client 10.2.2.2 wants to connect to server 198.2.4.5:80
  - OS assigns ephemeral port (1000)
- Connection request intercepted by firewall
  - Maps client to port of firewall (5000)
  - Creates NAT table entry

Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

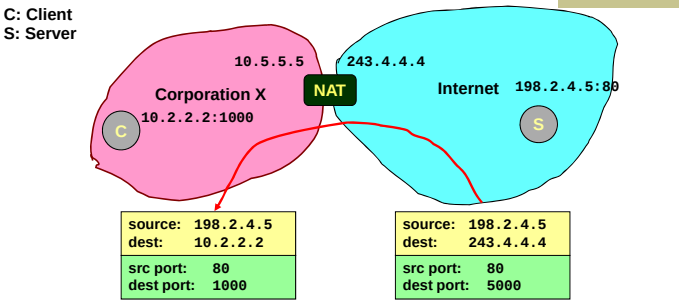
NAT: Client Request



- Firewall acts as proxy for client
  - Intercepts message from client and marks itself as sender

Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

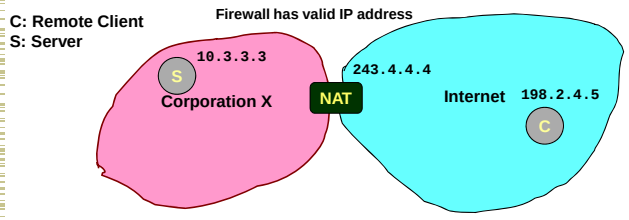
NAT: Server Response



Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

- Firewall acts as proxy for client
  - Acts as destination for server messages
  - Relabels destination to local addresses

NAT: Enabling Servers



- Use *port mapping* to make servers available

Int Addr	Int Port	NAT Port
10.3.3.3	80	80

  - Manually configure NAT table to include entry for well-known port
  - External users give address 243.4.4.4:80
  - Requests forwarded to server

Additional NAT Benefits

- They significantly reduce the need for public IP addresses
- NATs directly help with security
  - Hides IP addresses used in internal network
    - Easy to change ISP: only NAT box needs to have IP address
    - Fewer registered IP addresses required
  - Basic protection against remote attack
    - Does not expose internal structure to outside world
    - Can control what packets come in and out of system
    - Can reliably determine whether packet from inside or outside
- And NATs have many additional benefits
  - NAT boxes make home networking simple
  - Can be used to map between addresses from different address families, e.g, IPv4 and IPv6

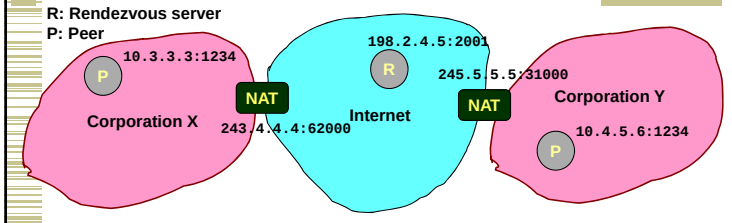
NAT Challenges

- NAT has to be consistent during a session.
  - Mapping (hard state) must be maintained during the session
    - Recall Goal 1 of Internet: Continue despite loss of networks or gateways
  - Recycle the mapping after the end of the session
    - May be hard to detect
- NAT only works for certain applications.
  - Some applications (e.g. ftp) pass IP information in payload - oops
  - Need application level gateways to do a matching translation
- NATs are a problem for peer-peer applications
  - File sharing, multi-player games, …
  - Who is server?
  - Need to “punch” hole through NAT

Principle: Fate Sharing

- "You can lose state information relevant to an entity's connections if and only if the entity itself is lost"
  - Example: OK to lose TCP state if either endpoint crashes
  - The TCP connection is no longer useful anyway!
- It is NOT okay to lose it if an unrelated entity goes down
  - Example: if an intermediate router reboots
- NATs violate this principle: if a NAT goes down, all communication session it supports are lost!
  - Unless you add redundancy and put state in persistent storage
- Bad news: many stateful "middleboxes" violate this rule
  - Firewalls, mobility services, … - more on this later
- Good news: today's hardware is very reliable

Many Options Exist for Peer-Peer



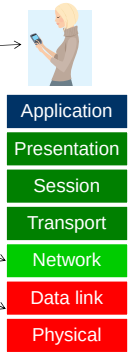
- NAT recognizes certain protocols and behaves as a application gateway
  - Used for standard protocols such as ftp
- Applications negotiate directly with NAT or firewall – need to be authorized
  - Multiple protocols dealing with different scenarios
- Punching holes in NAT: peers contact each other simultaneously using a known public (IP, port), e.g. used with rendezvous service
  - Use publicly accessible rendezvous service to exchange accessibility information
  - Assumes NATs do end-point independent mapping
- But remains painful!

Outline

- Translation: too many names and addresses!
- NATs
- ARP

How Do We Identify Hosts?

- Hosts have a
  - host name
  - IP address
  - MAC address
- There is a reason ..
  - Remember?
- But how do we translate?



IP to MAC Address Translation

- How does one find the Ethernet address of a IP host?
- Address Resolution Protocol - ARP
  - Broadcast search for IP address
    - E.g., “who-has 128.2.184.45 tell 128.2.206.138” sent to Ethernet broadcast (all FF address)
  - Destination responds (only to requester using unicast) with appropriate 48-bit Ethernet address
    - E.g, “reply 128.2.184.45 is-at 0:d0:bc:f2:18:58” sent to 0:c0:4f:d:ed:c6

Caching ARP Entries

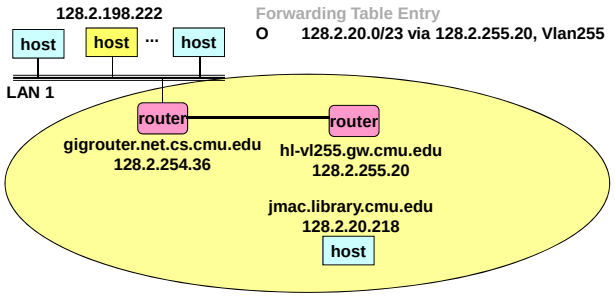
- Efficiency Concern
  - Would be very inefficient to use ARP request/reply every time need to send IP message to machine
- Each Host Maintains Cache of ARP Entries
  - Add entry to cache whenever get ARP response
  - “Soft state”: set timeout of ~20 minutes

ARP Cache Example

- Show using command “arp -a”

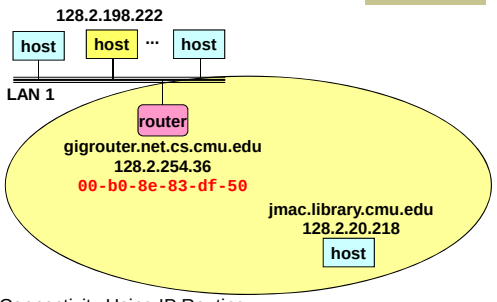
```
Interface: 128.2.222.198 on Interface 0x1000003
  Internet Address      Physical Address      Type
  128.2.20.218          00-b0-8e-83-df-50     dynamic
  128.2.102.129         00-b0-8e-83-df-50     dynamic
  128.2.194.66          00-02-b3-8a-35-bf     dynamic
  128.2.198.34          00-06-5b-f3-5f-42     dynamic
  128.2.203.3           00-90-27-3c-41-11     dynamic
  128.2.203.61          08-00-20-a6-ba-2b     dynamic
  128.2.205.192         00-60-08-1e-9b-fd     dynamic
  128.2.206.125         00-d0-b7-c5-b3-f3     dynamic
  128.2.206.139         00-a0-c9-98-2c-46     dynamic
  128.2.222.180         08-00-20-a6-ba-c3     dynamic
  128.2.242.182         08-00-20-a7-19-73     dynamic
  128.2.254.36          00-b0-8e-83-df-50     dynamic
```


CMU’s Internal Network Structure



- CMU Uses Routing Internally
  - Maintains forwarding tables using OSPF
  - Most CMU hosts cannot be reached at link layer

Proxy ARP



- Provides Link-Layer Connectivity Using IP Routing
  - Local router (gigrouter) sees ARP request
  - Uses IP addressing to locate host, i.e., which subnet
  - Replies with its own MAC address - becomes "Proxy" for remote host
    - Must then forward packets for that destination
  - Requestor thinks that it is communicating directly with remote host

Outline

- IP protocol
- IPv6
- Tunnels

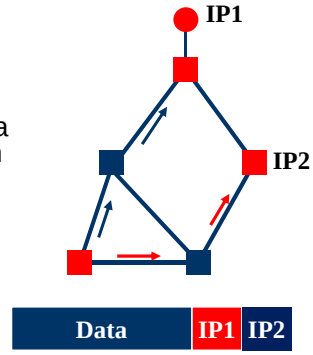
Motivation

There are many cases where not all routers have the same features or consistent state

- An experimental IP feature is only selectively deployed – how do we use this feature e-e?
  - E.g., IP multicast
- A few are using a protocol other than IPv4 – how can they communicate?
  - E.g., incremental deployment of IPv6
- I am traveling with a CMU laptop - how can I can I keep my CMU IP address?
  - E.g., must have CMU address to use services
- How do we make this happen?

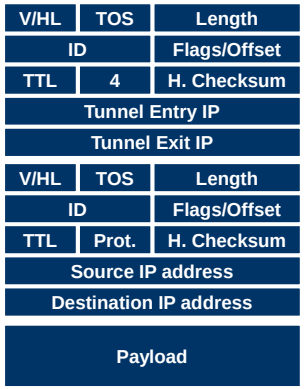
Tunneling

- Force a packet to go to a specific point in the network
  - Cannot rely on routers on regular path, e.g., an IPv6 packet
- Achieved by adding an extra IP header to the packet with a new destination address
  - Similar to putting a letter in another envelope
- Used to deal with new IP features
  - Mobile IP,..
  - Multicast, IPv6, research, ..



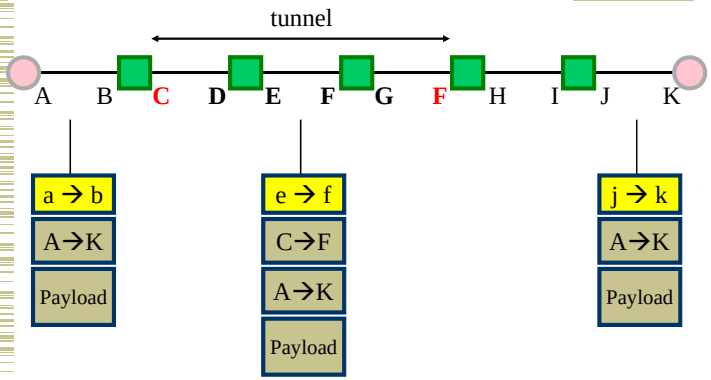
IP-in-IP Tunneling

- Described in RFC 1993.
- IP source and destination address identify tunnel endpoints.
- Protocol id = 4.
  - IP
- Several fields are copies of the inner-IP header.
  - TOS, some flags, ..
- Inner header is not modified, except for decrementing TTL.



25

Tunneling Example



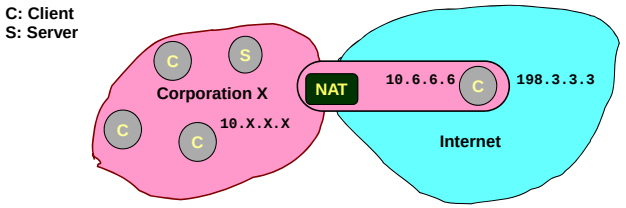
26

Tunneling Applications

- Virtual private networks.
  - Connect subnets of a corporation using IP tunnels
  - Often combined with IP Sec (later)
- Support for new or unusual protocols.
  - Routers that support the protocols use tunnels to "bypass" routers that do not support it
  - E.g. multicast, IPv6 (!)
- Force packets to follow non-standard routes.
  - Routing is based on outer-header
  - E.g. mobile IP (later)

27

Extending Private Network

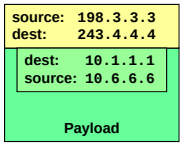


- Supporting Road Warrior
  - Employee working remotely with assigned IP address 198.3.3.3
  - Wants to appear to rest of corporation as if working internally
    - From address 10.6.6.6 – can also be a public address, e.g., 128.2.6.6
    - Gives access to internal services (e.g., ability to send mail)
- Virtual Private Network (VPN)
  - Overlays private network on top of regular Internet

28

Supporting VPN by Tunneling

- Idea: client sets up tunnel to company's firewall
- Example: client wants to send packet to internal node 10.1.1.1
- Entering Tunnel
  - Add extra IP header directed to firewall (243.4.4.4)
  - Original header becomes part of payload
  - Possible to encrypt it
- Exiting Tunnel
  - Firewall receives packet
  - Strips off header
  - Sends through internal network to destination



29