Outline

• NAT

• Tunneling and VPNs

• Name and address translation

Altering the Addressing Model

• Original IP Model: Every host has unique IP address

• Implications
  • Any host can communicate with any other host
  • Any host can act as a server
    • Just need to know host ID and port number
  • System is open – complicates security
  • Any host can attack any other host
  • Possible to forge packets
    • Use invalid source address
  • Places pressure on the address space
    • Every host requires “public” IP address

Reducing IP Addresses

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

(Most) machines within organization do not need public IP addresses!
Network Address Translation (NAT)

- Within Organization
  - Assign every host an unregistered IP address
  - IP addresses 10/8 & 192.168/16 unassigned
  - Route within organization by IP protocol, can do subnetting, ...
- Firewall
  - 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

NAT: Client Request

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

NAT: Server Response

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

<table>
<thead>
<tr>
<th>Int Addr</th>
<th>Int Port</th>
<th>NAT Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3.3.3</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

- Firewall has valid IP address
- Corporation X
- Internet
- Corporation Y

### NAT Considerations

- NAT has to be consistent during a session.
  - Set up mapping at the beginning of a session and maintain it during the session
  - Recall 2nd level goal 1 of Internet: Continue despite loss of networks or gateways
  - What happens if your NAT reboots?
  - Recycle the mapping at 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
  - Need application level gateways to do a matching translation
  - Peer-peer apps, multi-player games have problems – who is the server?
- NAT is loved and hated
  - Breaks some applications
  - Inhibits deployment of new applications like p2p (but so do firewalls!)
  - Little NAT boxes make home networking simple.
  - Saves addresses. Makes allocation simple.

### Properties of Firewalls with NAT

- **Advantages**
  - 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
  - Basic protection against remote attack
- **Disadvantages**
  - Contrary to the “open addressing” scheme envisioned for IP addressing
  - Hard to support peer-to-peer applications
  - Why do so many machines want to serve port 1214?

### Many Options Exist

- NAT recognizes certain protocols and behaves as a application gateway
- 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
Tunneling

- Force a packet to go to a specific point in the network.
  - Path taken is different from the regular routing
- Achieved by adding an extra IP header to the packet with a new destination address.
  - Similar to putting a letter in another envelope
  - Preferable to using IP source routing option
- Used increasingly to deal with special routing requirements or new 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.

Tunneling Example
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)

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
    - Gives access to internal services (e.g., ability to send mail)
- Virtual Private Network (VPN)
  - Overlays private network on top of regular Internet

Supporting VPN by Tunneling

- Concept
  - Appears as if two hosts connected directly
- Usage in VPN
  - Create tunnel between client & firewall
  - Remote client appears to have direct connection to internal network
CMU CS VPN Example

• CS has server to provide VPN services
• Operation
  • Running echo server on CMU machine 128.2.198.135
  • Run echo client on laptop connected through DSL from non-CMU ISP
  • With VPN
    • server connected to 128.2.216.18 - VPN-18.NET.CS.CMU.EDU
  • Without VPN
    • server connected to 205.201.7.7 - dhcp-7-7.dsl.telerama.com
• Effect
  • For CMU hosts, packets appear to originate from within CMU

Overlay Networks

• A network “on top of the network”.
  • E.g., initial Internet deployment
    • Internet routers connected via phone lines
      • An overlay on the phone network
    • Tunnels between nodes on a current network
  • Examples: IPv6 “6bone”, multicast “Mbone”.
  • But not limited to IP-layer protocols…
    • Peer-to-peer networks, anonymising overlays
    • Application layer multicast
    • Improve routing, e.g. work around route failures

Important Concepts

• Changes to Addressing Model
  • Have moved away from “everyone knows everybody” model of original Internet
  • Firewalls + NAT hide internal networks
  • VPN / tunneling build private networks on top of commodity network
• IPv6
  • Cleanup of various v4 flaws
  • Larger addresses

Outline

• NAT
• Tunneling and VPNs
• Name and address translation
  • ARP
  • DNS
Too Much of a Good Thing?

- 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?
- 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
  - Set timeout of ~20 minutes

ARP Cache Example

- Show using command “arp -a”

<table>
<thead>
<tr>
<th>Internet Address</th>
<th>Physical Address</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.2.20.218</td>
<td>00-b0-8e-83-df-50</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.102.129</td>
<td>00-b0-8e-83-df-50</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.194.66</td>
<td>00-02-b3-8a-35-bf</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.198.34</td>
<td>00-06-5b-f3-5f-42</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.203.3</td>
<td>00-90-27-3c-41-11</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.203.61</td>
<td>08-00-20-a6-ba-2b</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.205.192</td>
<td>00-60-08-1e-9b-fd</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.206.125</td>
<td>00-d0-b7-c5-b3-f3</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.206.139</td>
<td>00-a0-c9-98-2c-46</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.222.180</td>
<td>08-00-20-a6-ba-c3</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.242.182</td>
<td>08-00-20-a7-19-73</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.2.254.36</td>
<td>00-b0-8e-83-df-50</td>
<td>dynamic</td>
</tr>
</tbody>
</table>
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
  - Becomes “Proxy” for remote host
    - Using its own MAC address
    - Requestor thinks that it is communicating directly with remote host

Naming

- How do we efficiently locate resources?
  - DNS: name \( \rightarrow \) IP address
- Challenge
  - How do we scale this to the wide area?
Obvious Solutions (1)

Why not centralize DNS?
- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update
- Doesn’t scale!

Obvious Solutions (2)

Why not use /etc/hosts?
- Original Name to Address Mapping
- Flat namespace
- /etc/hosts
- SRI kept main copy
- Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
- Many more downloads
- Many more updates

Domain Name System Goals

- Basically a wide-area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
  - Names mean the same thing everywhere
- Don’t need
  - Atomicity
  - Strong consistency

Programmer’s View of DNS

- Conceptually, programmers can view the DNS database as a collection of millions of host entry structures:

```c
/* DNS host entry structure */
struct addrinfo {
    int   ai_family; /* host address type (AF_INET) */
    size_t ai_addrlen; /* length of an address, in bytes */
    struct sockaddr *ai_addr; /* address */
    char    *ai_canonname; /* official domain name of host */
    struct addrinfo *ai_next; /* other entries for host */
};
```

- Functions for retrieving host entries from DNS:
  - `getaddrinfo`: query key is a DNS host name.
  - `getnameinfo`: query key is an IP address.
DNS Message Format

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Questions</td>
<td>No. of Answer RRs</td>
</tr>
<tr>
<td>No. of Authority RRs</td>
<td>No. of Additional RRs</td>
</tr>
</tbody>
</table>

Name, type fields for a query

RRs in response to query

Records for authoritative servers

Additional "helpful info that may be used"

DNS Header Fields

- **Identification**
  - Used to match up request/response

- **Flags**
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution

DNS Records

RR format: (class, name, value, type, ttl)

- DB contains tuples called resource records (RRs)
- Classes = Internet (IN), Chaosnet (CH), etc.
- Each class defines value associated with type

**FOR IN class:**

- **Type=A**
  - name is hostname
  - value is IP address
- **Type=NS**
  - name is domain (e.g. foo.com)
  - value is name of authoritative name server for this domain
- **Type=CNAME**
  - name is an alias name for some "canonical" (the real) name
  - value is canonical name
- **Type=MX**
  - value is hostname of mailserver associated with name

Properties of DNS Host Entries

- Different kinds of mappings are possible:
  - Simple case: 1-1 mapping between domain name and IP addr:
    - kittyhawk.cmcl.cs.cmu.edu maps to 128.2.194.242
  - Multiple domain names maps to the same IP address:
    - eecs.mit.edu and cs.mit.edu both map to 18.62.1.6
  - Single domain name maps to multiple IP addresses:
    - aol.com and www.aol.com map to multiple IP addrs.
  - Some valid domain names don't map to any IP address:
    - for example: cmcl.cs.cmu.edu
DNS Design: Zone Definitions

- Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree
  - A zone has an associated set of name servers
  - Must store list of names and tree links

- Subtree
- Single node
- Complete Tree

DNS Design: Management

- Zones are created by convincing owner node to create/delegate a subzone
  - Records within zone stored multiple redundant name servers
  - Primary/master name server updated manually
  - Secondary/redundant servers updated by zone transfer of name space
  - Zone transfer is a bulk transfer of the “configuration” of a DNS server – uses TCP to ensure reliability

- Example:
  - CS.CMU.EDU created by CMU.EDU administrators
  - Who creates CMU.EDU or .EDU?

DNS: Root Name Servers

- Responsible for “root” zone
- Approx. 13 root name servers worldwide
  - Currently (a-m).root-servers.net
  - Very well protected
  - Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers
  - Newer picture ➔ www.root-servers.org

Root Zone

- Generic Top Level Domains (gTLD) = .com, .net, .org, etc…
- Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc…
- Root server ((a-m).root-servers.net) also used to cover gTLD domains
  - Load on root servers was growing quickly!
  - Moving .com, .net, .org off root servers was clearly necessary to reduce load ➔ done Aug 2000
Servers/Resolvers

- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone

Typical Resolution

- Steps for resolving www.cmu.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server (S1)
  - S1 queries root server (S2) for (www.cmu.edu)
  - S2 returns NS record for cmu.edu (S3)
  - What about A record for S3?
    - This is what the additional information section is for (PREFETCHING)
  - S1 queries S3 for www.cmu.edu
  - S3 returns A record for www.cmu.edu

Lookup Methods

Recursive query:
- Server goes out and searches for more info (recursive)
- Only returns final answer or “not found”

Iterative query:
- Server responds with as much as it knows (iterative)
- “I don’t know this name, but ask this server”

Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative
**Workload and Caching**

- Are all servers/names likely to be equally popular?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
- DNS negative queries are cached
  - Don’t have to repeat past mistakes, e.g., misspellings
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
- Responses can include additional information
  - Often used for prefetching, e.g., CNAME/MX/NS records

**Typical Resolution**

![Diagram of DNS resolution](image)

**Subsequent Lookup Example**

![Diagram of subsequent DNS lookup](image)

**Reliability**

- DNS servers are replicated
  - Name service available if ≥ one replica is up
  - Queries can be load balanced between replicas
  - Queries return multiple A records
- UDP used for queries
  - Need reliability → must implement this on top of UDP!
  - Why not just use TCP?
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
  - Same identifier for all queries
  - Don’t care which server responds
Mail Addresses

- MX records point to mail exchanger for a name
  - E.g. mail.acm.org is MX for acm.org
- Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery?
  - Needed critical mass of such mailers

Tracing Hierarchy (1)

- Dig Program
  - Allows querying of DNS system
  - Use flags to find name server (NS)
  - Disable recursion so that operates one step at a time

```unix
$ dig +norecurse @a.root-servers.net NS kittyhawk.cmcl.cs.cmu.edu
;; AUTHORITY SECTION:
edu.                   172800  IN      NS      L3.NSTLD.COM.
edu.                   172800  IN      NS      D3.NSTLD.COM.
edu.                   172800  IN      NS      A3.NSTLD.COM.
edu.                   172800  IN      NS      E3.NSTLD.COM.
edu.                   172800  IN      NS      C3.NSTLD.COM.
edu.                   172800  IN      NS      F3.NSTLD.COM.
edu.                   172800  IN      NS      G3.NSTLD.COM.
edu.                   172800  IN      NS      B3.NSTLD.COM.
edu.                   172800  IN      NS      M3.NSTLD.COM.
```

DNS (Summary)

- Motivations → large distributed database
  - Scalability
  - Independent update
  - Robustness
- Hierarchical database structure
  - Zones
  - How is a lookup done
- Caching/prefetching and TTLs
- Reverse name lookup
- What are the steps to creating your own domain?

Next Lecture

- How do forwarding tables get built?
- Routing protocols
  - Distance vector routing
  - Link state routing