15-441 15-641 Computer Networking

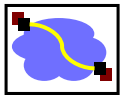
Lecture 9 – Translations

Peter Steenkiste

Fall 2015

www.cs.cmu.edu/~prs/15-441-F15

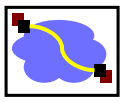
Outline



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

2

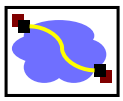
Altering the Addressing Model



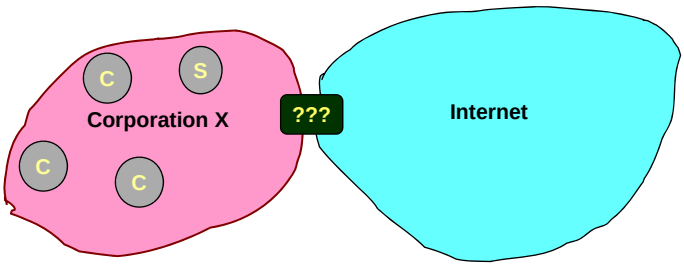
- Original IP Model: Every host has unique IP address
- Implications
  - Any host can communicate with any other host
  - Any host can act as a server
    - Just need to know host ID and port number
- System is open – complicates security
  - Any host can attack any other host
  - Possible to forge packets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

3

Challenges When Connecting to Public Internet



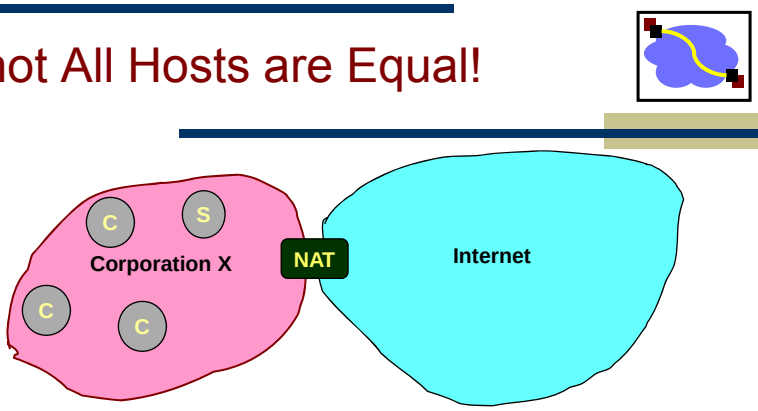
C: Client
S: Server



- 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

4

But not All Hosts are Equal!

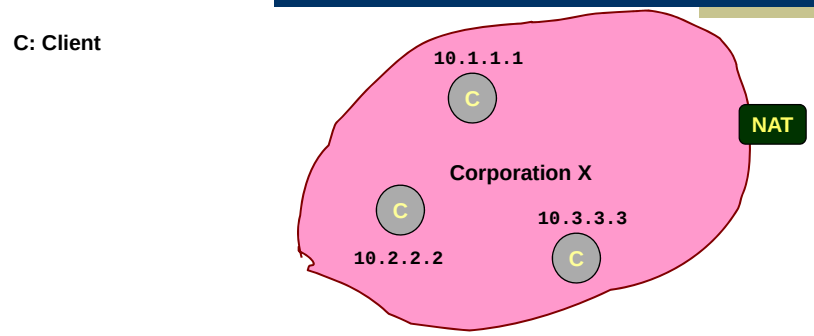


- Most machines within organization are used by individuals
  - For most applications, they act as clients
- Small number of machines act as servers for entire organization
  - E.g., mail server, web, ..
  - All traffic to outside passes through firewall

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

5

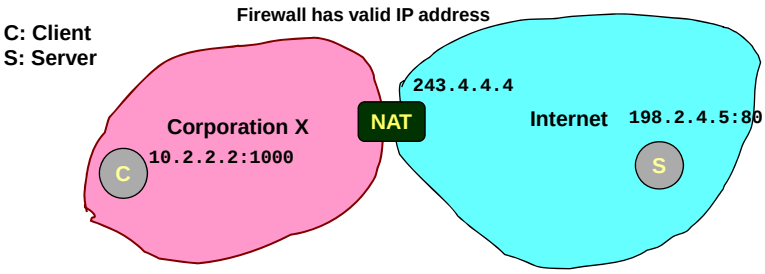
Reducing Address Use: Network Address Translation



- Within Organization: assign every host a private IP address
  - IP addresses 10/8 & 192.168/16 set aside for this
  - Route within organization by IP protocol, can do subnetting, …
- NAT translates between public and private IP addresses
  - Does not let any packets from internal nodes escape
  - Outside world does not need to know about internal addresses

6

NAT: Opening Client Connection

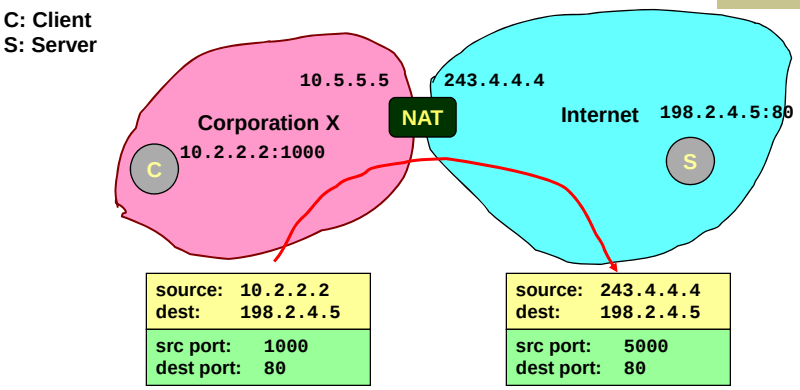


- Client 10.2.2.2 wants to connect to server 198.2.4.5:80
  - OS assigns ephemeral port (1000)
- Connection request intercepted by firewall
  - Maps client to port of firewall (5000)
  - Creates NAT table entry

Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

7

NAT: Client Request



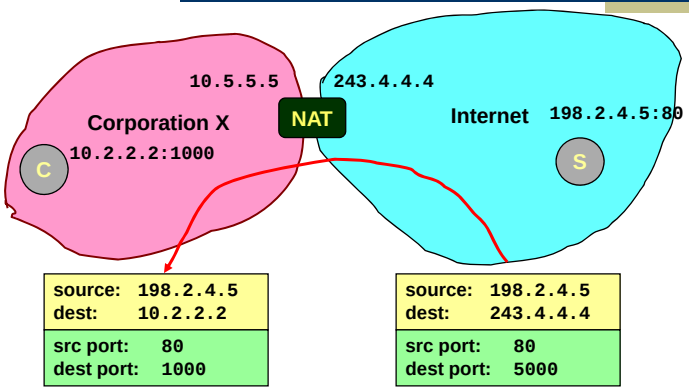
Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

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

8

NAT: Server Response

C: Client
S: Server

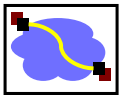


Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

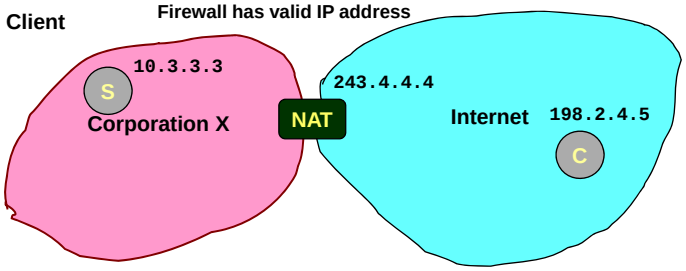
- Firewall acts as proxy for client
  - Acts as destination for server messages
  - Relabels destination to local addresses

9

NAT: Enabling Servers



C: Remote Client
S: Server



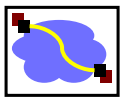
- Use *port mapping* to make servers available

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

10

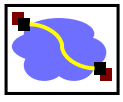
Additional NAT Benefits



- NATs already 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
- 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

11

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

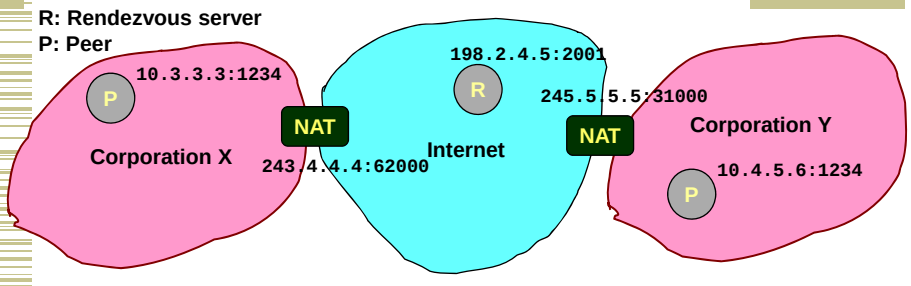
12

Principle: Fate Sharing

- Lose state information relevant to an entity's connections if and only if the entity itself is lost.
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose it if an intermediate router reboots
  - Is this still true in today's network?
    - NATs and firewalls
- Tradeoffs
  - Survivability: Heterogeneous network → less information available to end hosts and Internet level recovery mechanisms
  - Trust: must trust endpoints more

13

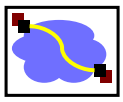
Many Options Exist for Peer-Peer



- 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!

14

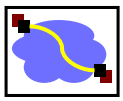
Outline



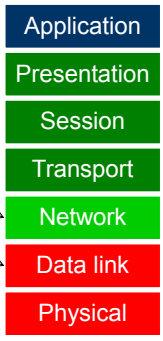
- Translation: too many names and addresses!
- NATs
- ARP
- DNS

15

Too Much of a Good Thing?

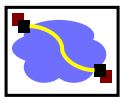


- Hosts have a
  - host name
  - IP address
  - MAC address
- There is a reason ..
  - Remember?
- But how do we translate?



16

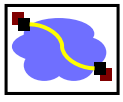
IP to MAC Address Translation



- 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

17

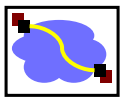
Caching ARP Entries



- 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

18

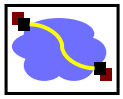
ARP Cache Example

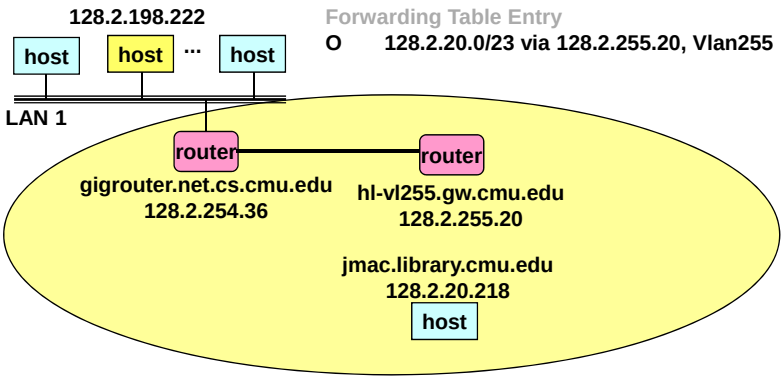


- Show using command “arp -a”

```
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



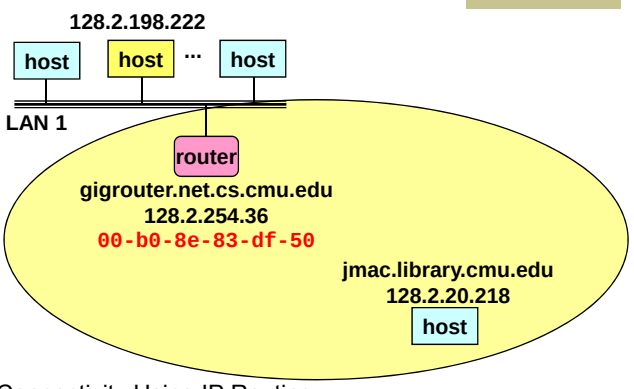


Forwarding Table Entry

O 128.2.20.0/23 via 128.2.255.20, Vlan255

- 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

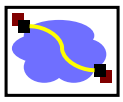
21

Outline

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

22

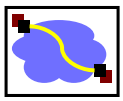
Naming



- How do we efficiently locate resources?
  - DNS: name → IP address
- Challenge
  - How do we scale this to the wide area?

23

Obvious Solutions (1)



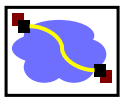
Why not centralize DNS?

- Distant centralized database
  - Traffic volume
- Single point of failure
- Single point of update
- Single point of control

- Doesn't *scale!*

24

Obvious Solutions (2)

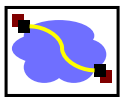


Why not use /etc/hosts?

- Original Name to Address Mapping
  - Flat namespace
  - /etc/hosts keeps track of the mappings
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Many more updates

25

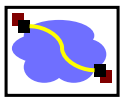
Domain Name System Goals



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

26

Programmer's View of DNS



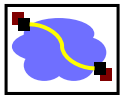
- Conceptually, programmers can view the DNS database as a collection of millions of *host entry structures*:

```
/* 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.

27

DNS Records



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

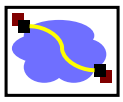
- DB contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

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**

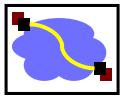
28

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 Message Format



12 bytes

Identification	Flags
No. of Questions	No. of Answer RRs
No. of Authority RRs	No. of Additional RRs

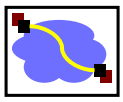
Name, type fields for a query → Questions (variable number of answers)

RRs in response to query → Answers (variable number of resource records)

Records for authoritative servers → Authority (variable number of resource records)

Additional "helpful info that may be used → Additional Info (variable number of resource records)

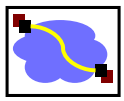
DNS Header Fields

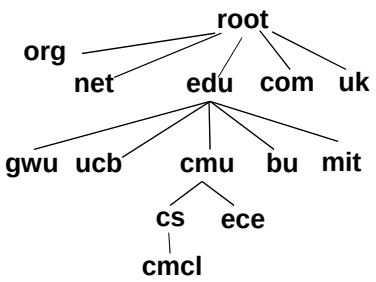


- Identification
  - Used to match up request/response
- Flags
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution

31

DNS Design: Hierarchy Definitions

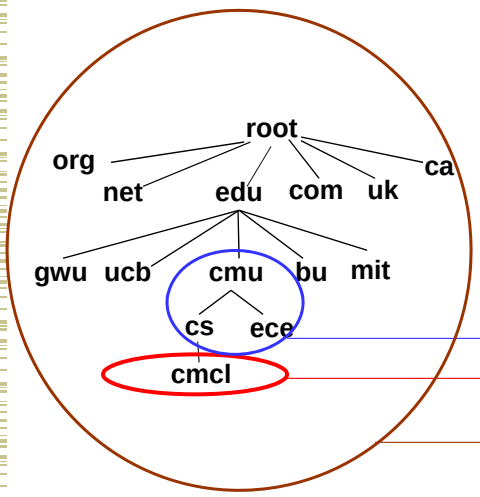




- Each node in hierarchy stores a list of names that end with same suffix
  - Suffix = path up tree
- E.g., given this tree, where would following be stored:
  - Fred.com
  - Fred.edu
  - Fred.cmu.edu
  - Fred.cmcl.cs.cmu.edu
  - Fred.cs.mit.edu

32

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

33

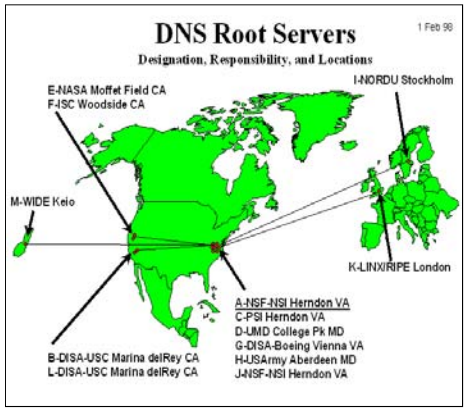
DNS Design: Management

- Zones are created by convincing owner node (parent) 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?

34

DNS: Root Name Servers

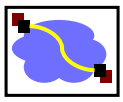
- Responsible for “root” zone
- Approx. 13 root name servers worldwide
  - Currently {a-m}.root-servers.net
  - Very well protected
- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers
  - Newer picture → www.root-servers.org



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

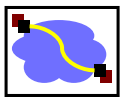
Servers/Resolvers

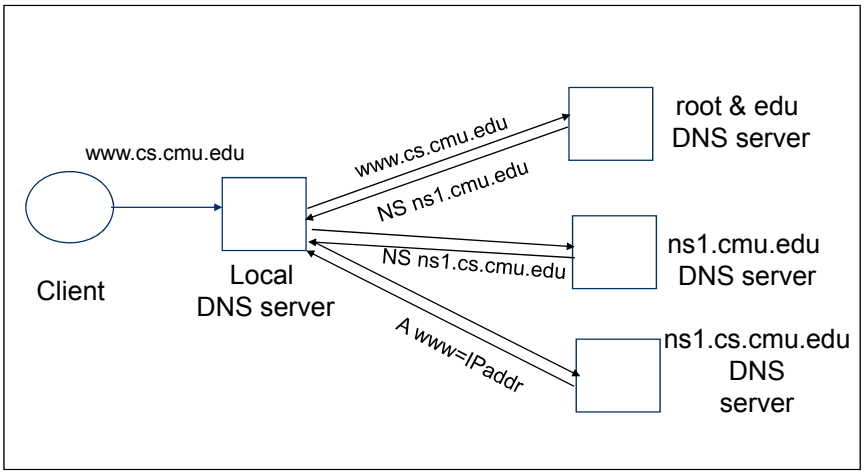


- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries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

37

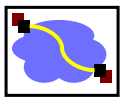
Typical Resolution





38

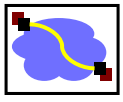
Typical Resolution: Steps



- Steps for resolving www.cmu.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server (S_1)
  - S_1 queries root server (S_2) for (www.cmu.edu)
  - S_2 returns NS record for cmu.edu (S_3)
  - What about A record for S_3?
    - This is what the additional information section is for (PREFETCHING)
  - S_1 queries S_3 for www.cmu.edu
  - S_3 returns A record for www.cmu.edu

39

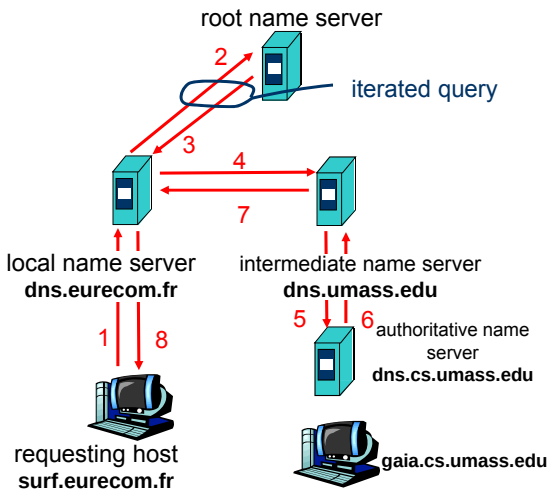
Lookup Methods



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

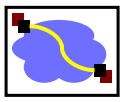
Iterative query:
- Server responds with as much as it knows (iterative)
- "I don't know this name, but ask this server"

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



40

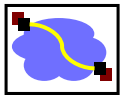
Workload and Caching

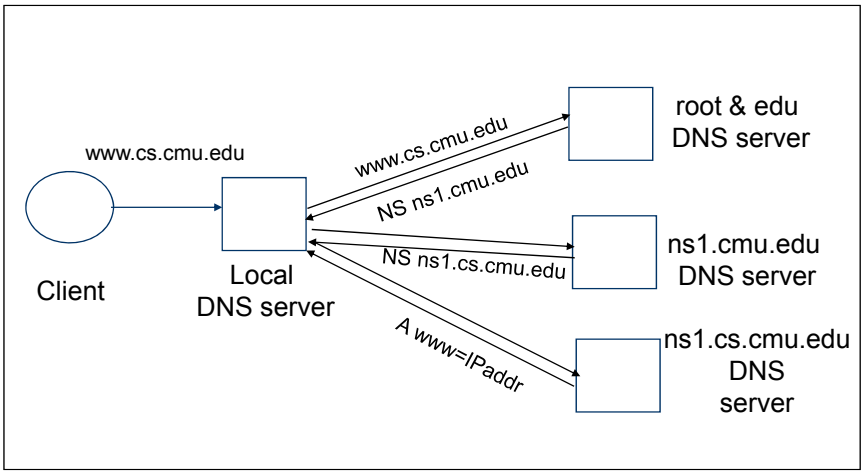


- Are all servers/names likely to be equally popular?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
- DNS negative queries are cached
  - Don't have to repeat past 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

41

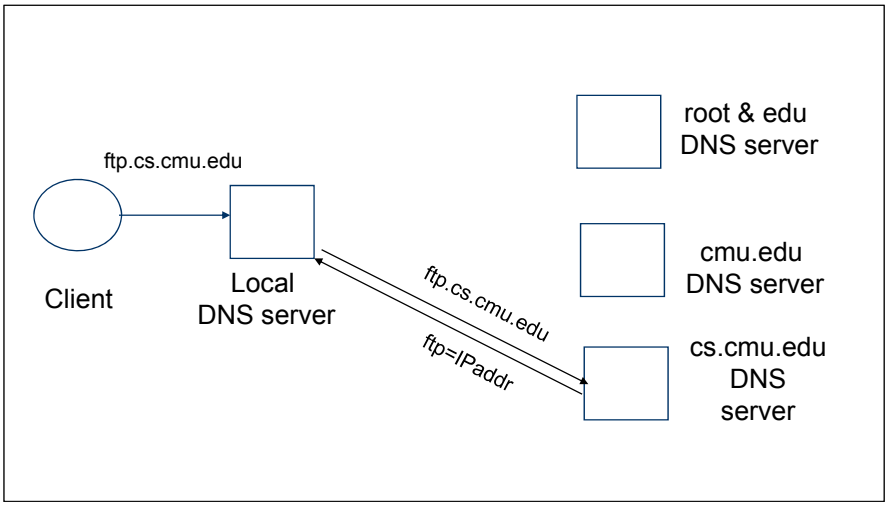
Typical Resolution





42

Subsequent Lookup Example



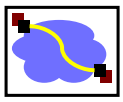
43

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
  - Client does not care which server responds

44

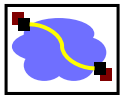
Mail Addresses



- 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

45

Tracing Hierarchy (1)



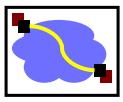
- Dig Program
  - Allows querying of DNS system
  - Use flags to find name server (NS)
  - Disable recursion so that operates one step at a time

```
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.
```

46

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?

47