15-441 15-641 Computer Networking

Lecture 8 – DNS

Peter Steenkiste

Fall 2016

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

Outline

- The IP protocol
  - IPv4
  - IPv6
- IP in practice
  - Network address translation
  - Address resolution protocol
  - Tunnels

How Do We Identify Hosts?

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

Naming

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

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!*

5

Obvious Solutions (2)

Why not use /etc/hosts?

- Original Name to Address Mapping
  - Flat namespace
  - /etc/hosts keeps track of the mappings
  - SRI kept a master copy
  - All computers periodically download the master
- Number of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Updates are larger
  - Many more updates

6

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

7

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.

8

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`

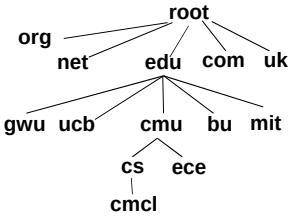
DNS Message Format

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

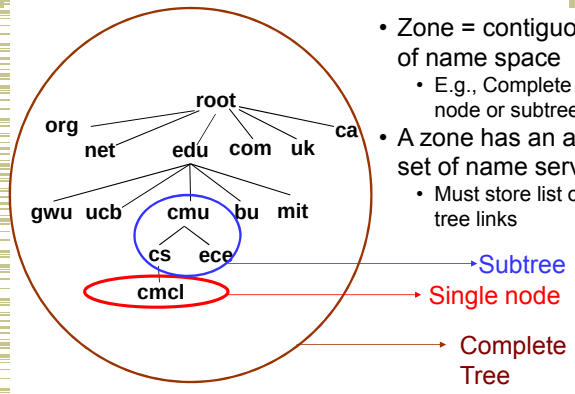
- Identification
  - Used to match up request/response
- Flags
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution

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

DNS Design: Zone Definitions



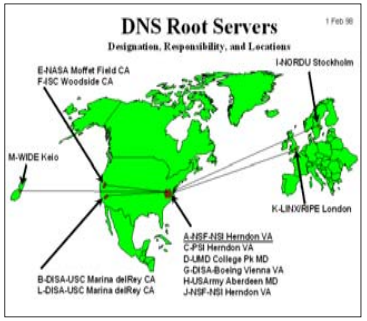
- Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree
- A zone has an associated set of name servers
  - Must store list of names and tree links

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?

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

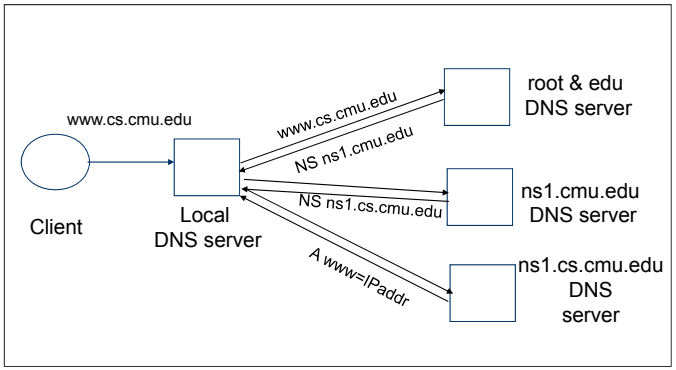
17

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

18

Typical Resolution



19

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

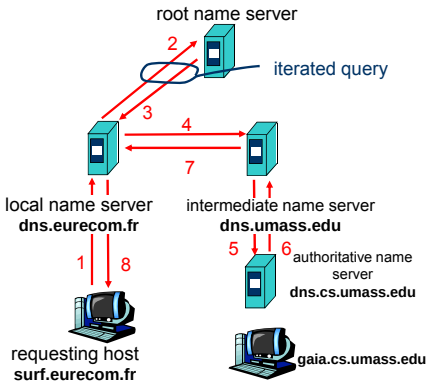
20

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"

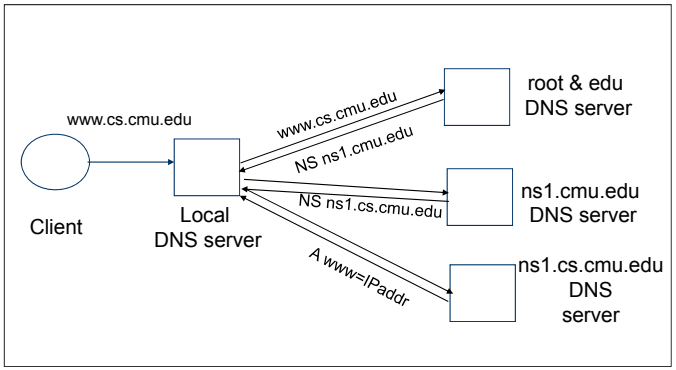
Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative



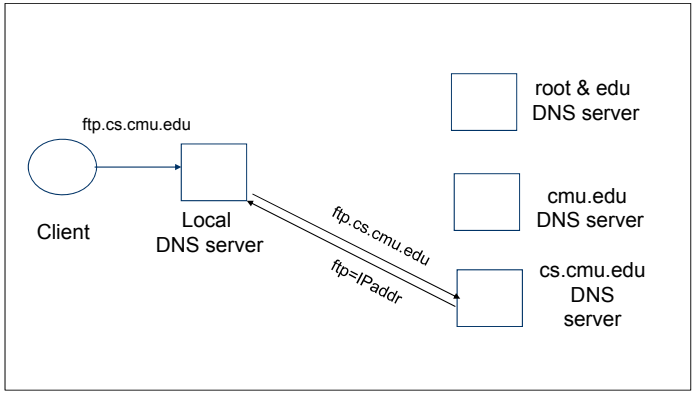
Workload and Caching

- 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



Subsequent Lookup Example



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

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?

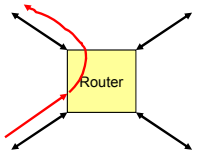
Outline

- Routing intro
- Distance Vector
- Link State

29

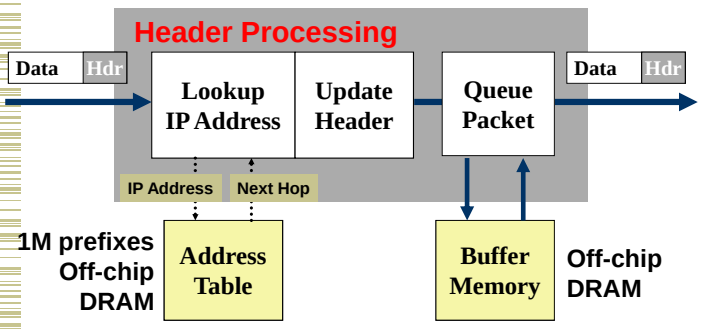
IP Forwarding

- The Story So Far…
  - IP addresses are structured to reflect Internet structure
  - IP packet headers carry these addresses
  - When Packet Arrives at Router
    - Examine header to determine intended destination
    - *Look up in table to determine next hop in path – longest prefix match*
    - Send packet out appropriate port
- This/next lecture
  - How to generate the forwarding table



30

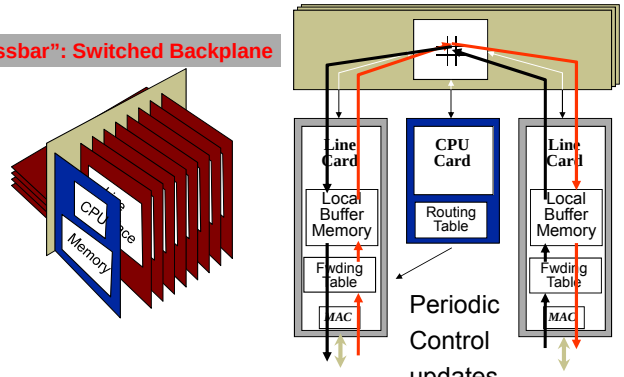
Generic Router Architecture



31

Third Generation Routers

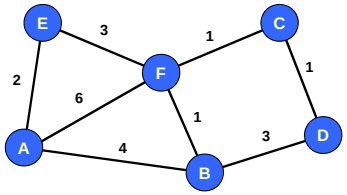
"Crossbar": Switched Backplane



Periodic Control updates

Graph Model

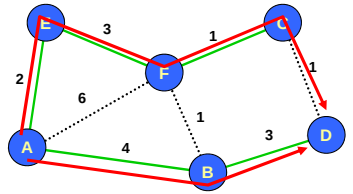
- Represent each router as node
- Direct link between routers represented by edge
  - Symmetric links ⇒ undirected graph
- Edge "cost" c(x,y) denotes measure of difficulty of using link
  - delay, $ cost, or congestion level
- Task
  - Determine least cost path from every node to every other node
    - Path cost d(x,y) = sum of link costs



Routes from Node A

Forwarding Table for A

Dest	Cost	Next Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

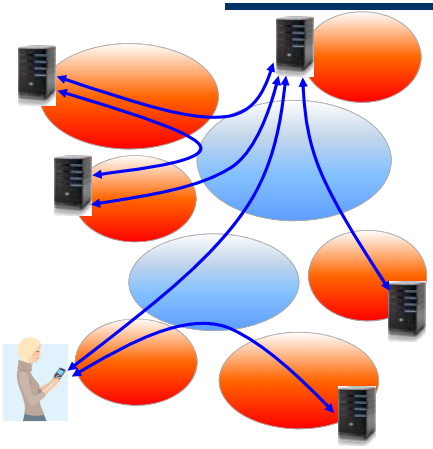


- Set of shortest paths forms tree
  - Shortest path spanning tree
- Solution is not unique
  - E.g., A-E-F-C-D also has cost 7

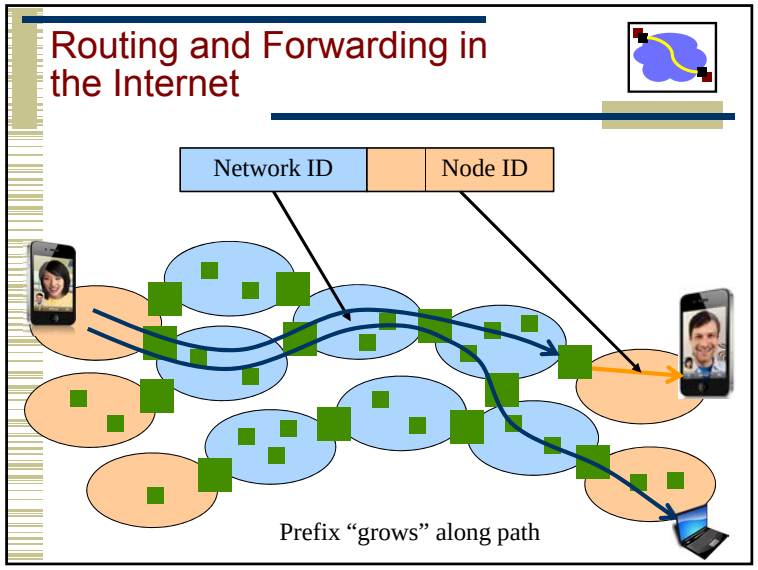
Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Link-state
  - Every node collects complete graph structure
  - Each computes shortest paths from it
  - Each generates its own routing table
- Distance-vector
  - No one has copy of graph
  - Nodes construct their own tables iteratively
  - Each sends information about its table to neighbors

Routing Hierarchy



- IP packets must travel across domains – inter-domain routing
  - Primary role of IP
  - Based on CIDR prefix
- Must also travel through domains – intro-domain routing
  - Across subnets
  - Based on subnet ID or longer prefix


Routing and Forwarding in the Internet
Network ID
Node ID
Prefix “grows” along path