Lecture 8: Application Layer Domain Name System

COMP 332, Fall 2018

Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, slides by Daniel Zappala at Brigham and Young University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

Announcements

- hwk 3 due today at 11:59p, hwk 4 posted
  - I will post hw3 code in a few days, once all hwk3 submitted
- Tu night help session conflicts with Algorithms help session
  - Q: is there a better day/time for the Tu help session?
  - Q: is it better for me to go to the Mo or Tu night help session?

Domain names

- overview

Domain Name System

- name resolution
- protocol
- dig and wireshark
- attacks

Domain Names
OVERVIEW

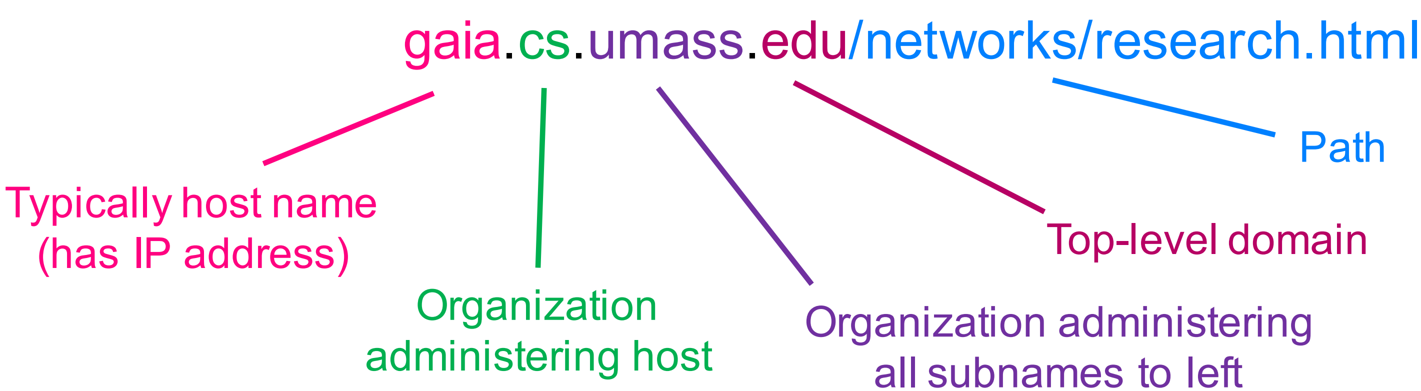
Hostname vs. domain

Hostname

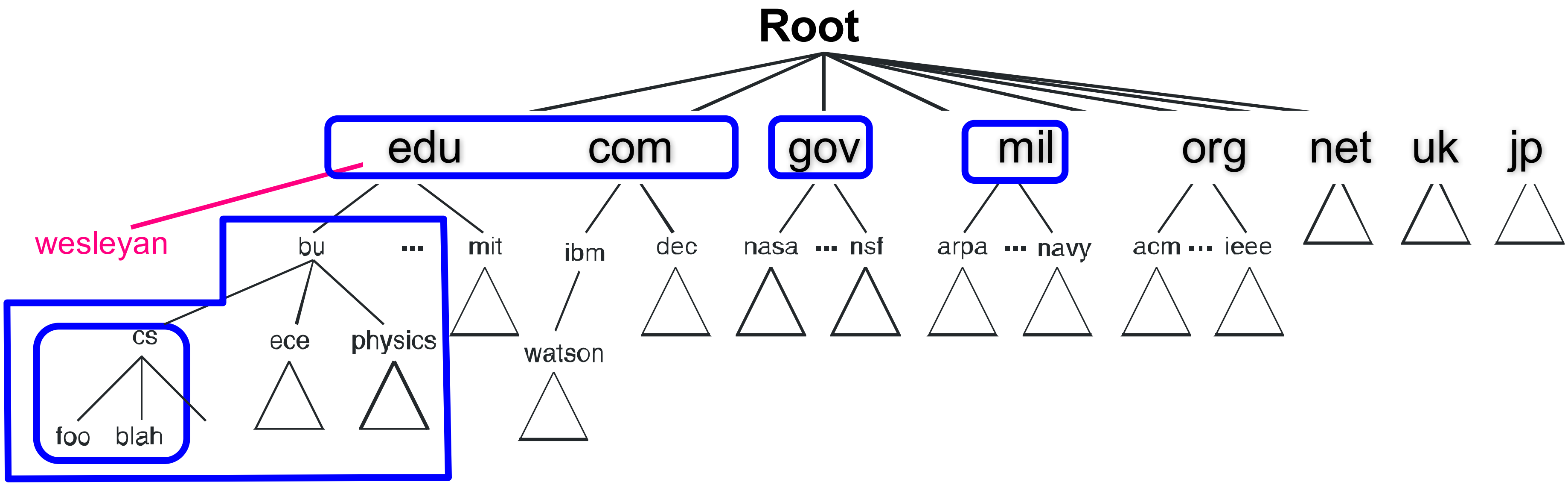
- comprises host + domain
- may be domain if properly organized into domain name system

Domain

- may be hostname if it assigned to Internet host and associated with host's IP address



Administration zones based on name hierarchy



Tree is split into zones

- each zone owned by organization responsible for their part of hierarchy
- delegation can happen along administrative boundaries
  - Boston university controls *.bu.edu
  - cs controls *.cs.bu.edu
- Wesleyan's zone isn't so interesting…

It wasn't always this way...

Before DNS, when the Internet was tiny

- all hostname to IP address mapping was in flat file: hosts.txt

Centralized and manual system

- changes sent via email to SRI (Menlo Park, CA)
- machines across Internet periodically FTPd latest copy of hosts.txt
- hostnames had no enforceable convention
  - Compsci_server1_at_uc

Q: Eventually fell apart for several reasons. Why?

- not scalable
  - SRI unable to handle load from Internet growth. No single org could
- hard to enforce name uniqueness.
  - is 'uc' U of California? U of Cambridge? University of Caledonia? …
- inaccurate copies of hosts.txt persisted across Internet

Domain Name System

OVERVIEW

DNS comprises 2 components

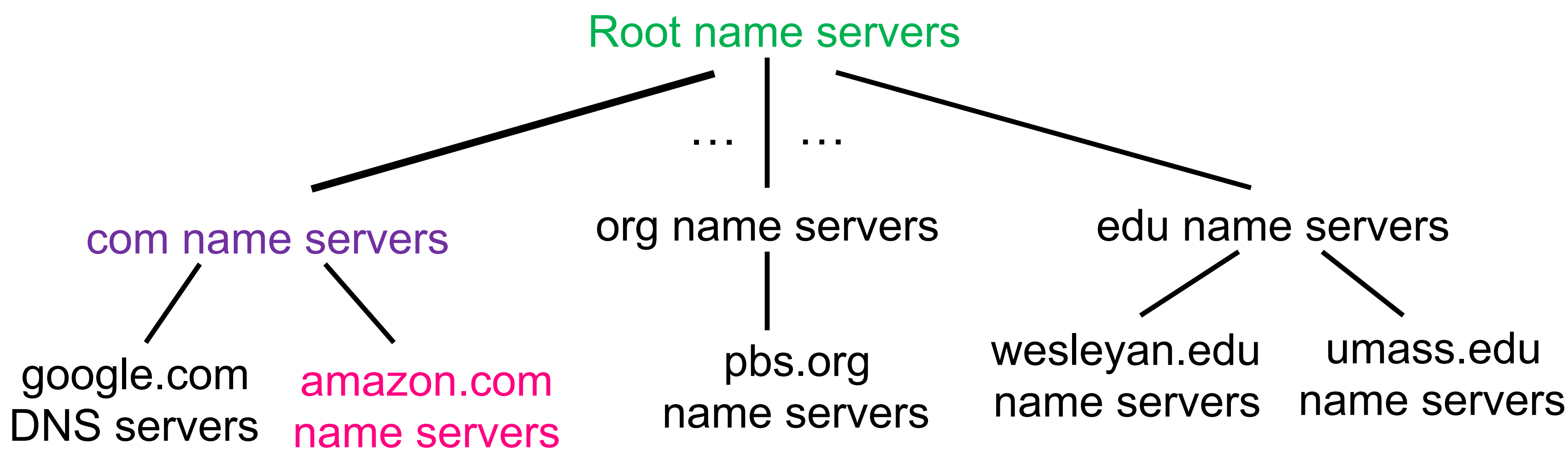
1. App layer protocol
   - translates between identifiers
     - hostnames ⇔IP addresses
   - used by other app-layer protocols
     - HTTP, SMTP, …
   - runs primarily over UDP (port 53)

Core Internet function
implemented as app-layer protocol, complexity at network edge

2. Distributed hierarchical database
   - organized to follow name hierarchy
     - fixes namespace issues and ownership
   - distributed across many name servers
     - for scalability
   - no name server has all mappings (resource records)
     - for scalability, updatability, freshness

Q: why not centralize DNS?
single point of failure, handles less traffic volume, need to go to distant centralized database, harder to maintain

A distributed, hierarchical database



Client wants IP for www.amazon.com

- client queries root server to find com name server
- client queries com name server to get amazon.com name server
- client queries amazon.com name server to get IP address for www.amazon.com

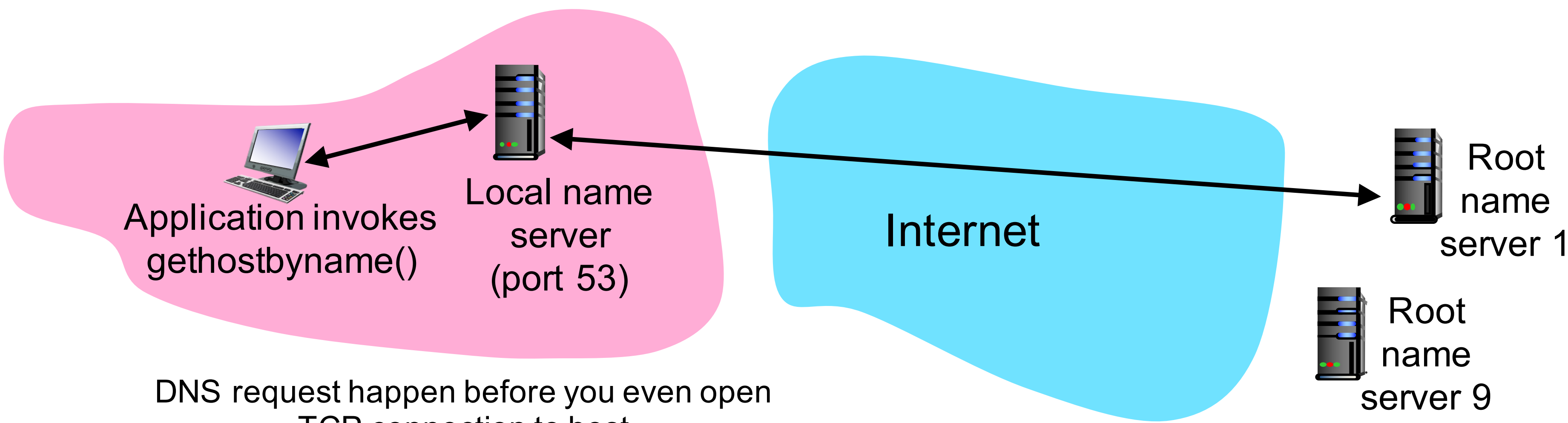
Local name server

When host makes DNS query, sent 1st to its local name server

- has local cache of recent name-to-address translation pairs
  - but may be out of date!
- acts as proxy, forwards query into hierarchy if it cannot resolve

Each ISP (residential, company, university, …) has one

- hosts get IP address of local name server from DHCP or manual config



DNS request happen before you even open TCP connection to host

Python3

socket.gethostbyname(*hostname*)

- Translate a host name to IPv4 address format. The IPv4 address is returned as a string, such as'100.50.200.5'. If the host name is an IPv4 address itself it is returned unchanged. See gethostbyname_ex() for a more complete interface. gethostbyname() does not support IPv6 name resolution, and getaddrinfo() should be used instead for IPv4/v6 dual stack support.

socket.gethostname()

- Return a string containing the hostname of the machine where the Python interpreter is currently executing.

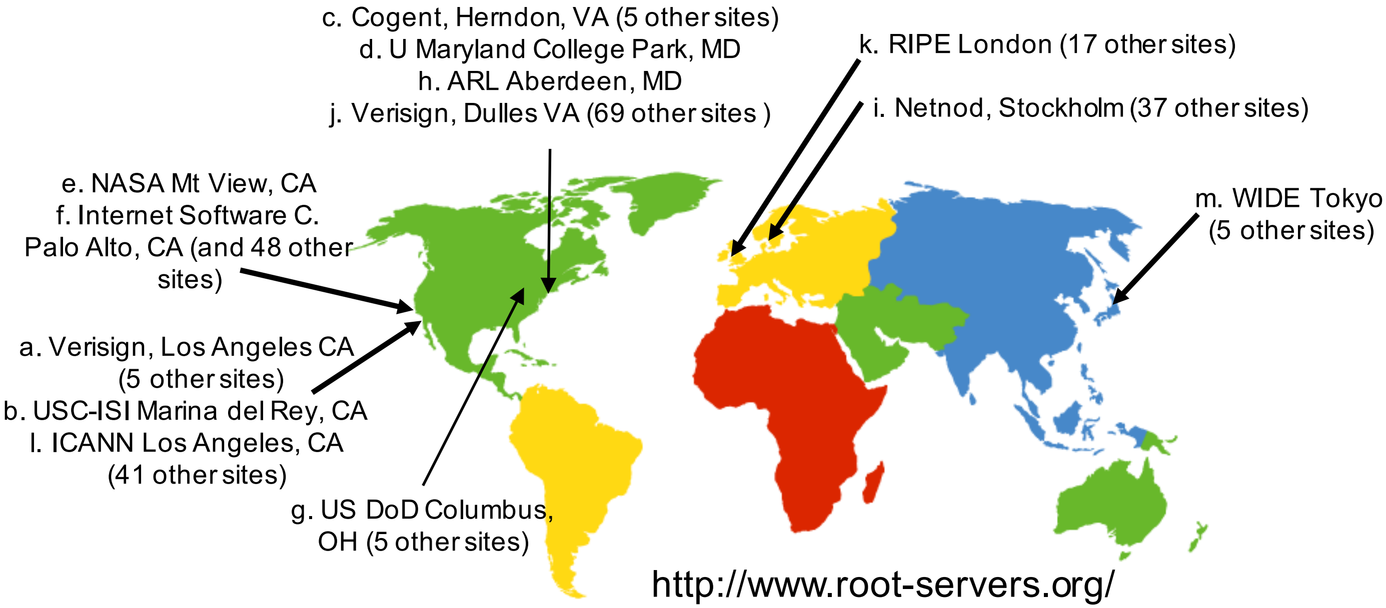
AF_INET address family

- *host* is a string representing either a hostname in Internet domain notation like 'daring.cwi.nl' or an IPv4 address like '100.50.200.5',

Root name servers

13 logical name servers, [a-m].root-servers.net

- know all Top Level Domain (TLD) name servers and their IP addr
- each root server replicated many times (933 currently)
- contact authoritative name server if name mapping not known



Robust distributed infrastructure

dig

```
> dig

; <<>> DiG 9.8.3-P1 <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27061
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       35480   IN      NS      d.root-servers.net.
.                       35480   IN      NS      k.root-servers.net.
.                       35480   IN      NS      m.root-servers.net.
.                       35480   IN      NS      h.root-servers.net.
.                       35480   IN      NS      i.root-servers.net.
.                       35480   IN      NS      a.root-servers.net.
.                       35480   IN      NS      e.root-servers.net.
.                       35480   IN      NS      l.root-servers.net.
.                       35480   IN      NS      c.root-servers.net.
.                       35480   IN      NS      j.root-servers.net.
.                       35480   IN      NS      g.root-servers.net.
.                       35480   IN      NS      f.root-servers.net.
.                       35480   IN      NS      b.root-servers.net.

;; Query time: 28 msec
```

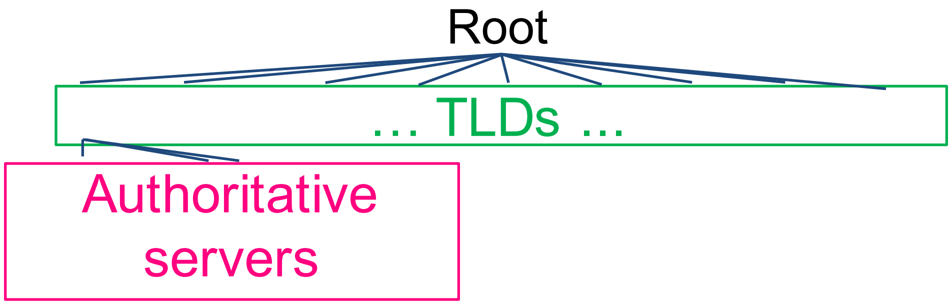
Top-level domain and authoritative servers

Top-level domain (TLD) servers

- know authoritative name servers and their ip addresses for (sub)-domains in their zone
- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g., uk, fr, ca, jp
- Verisign maintains servers for .com, .edu TLDs, among others

Authoritative servers

- know ip addresses for all hosts in organization's domain
- organization's own name server(s)
- provides authoritative hostname to IP mappings for org's named hosts
- maintained by organization or service provider



Domain Name System

NAME RESOLUTION

Resolving non-local names

No single name server has complete information

If local name server can't resolve address

- contacts root name server

13 root name servers world-wide

- each has addresses of name servers for all TLD name servers
  - e.g., wesleyan.edu, ibm.com

What happens?

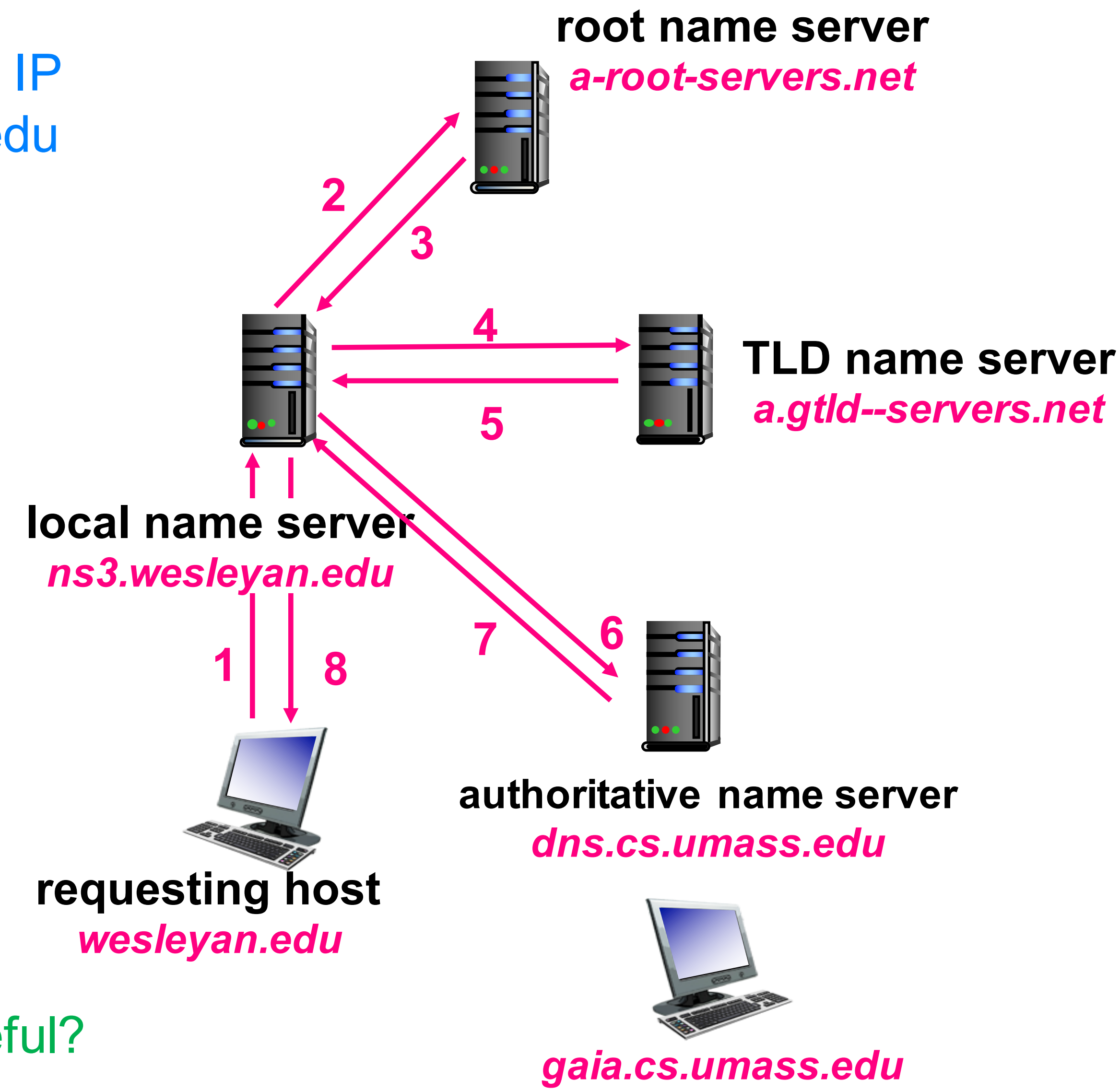
- contact root server
  - returns IP address of name server which should be contacted next
- contact TLD name server
  - may itself return a pointer to another name server
- iterative process of following name server pointers

Iterative name resolution

Host at wesleyan.edu wants IP address for gaia.cs.umass.edu

Iterated query

- requesting server responsible for name resolution
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Q: why is iterated query useful?

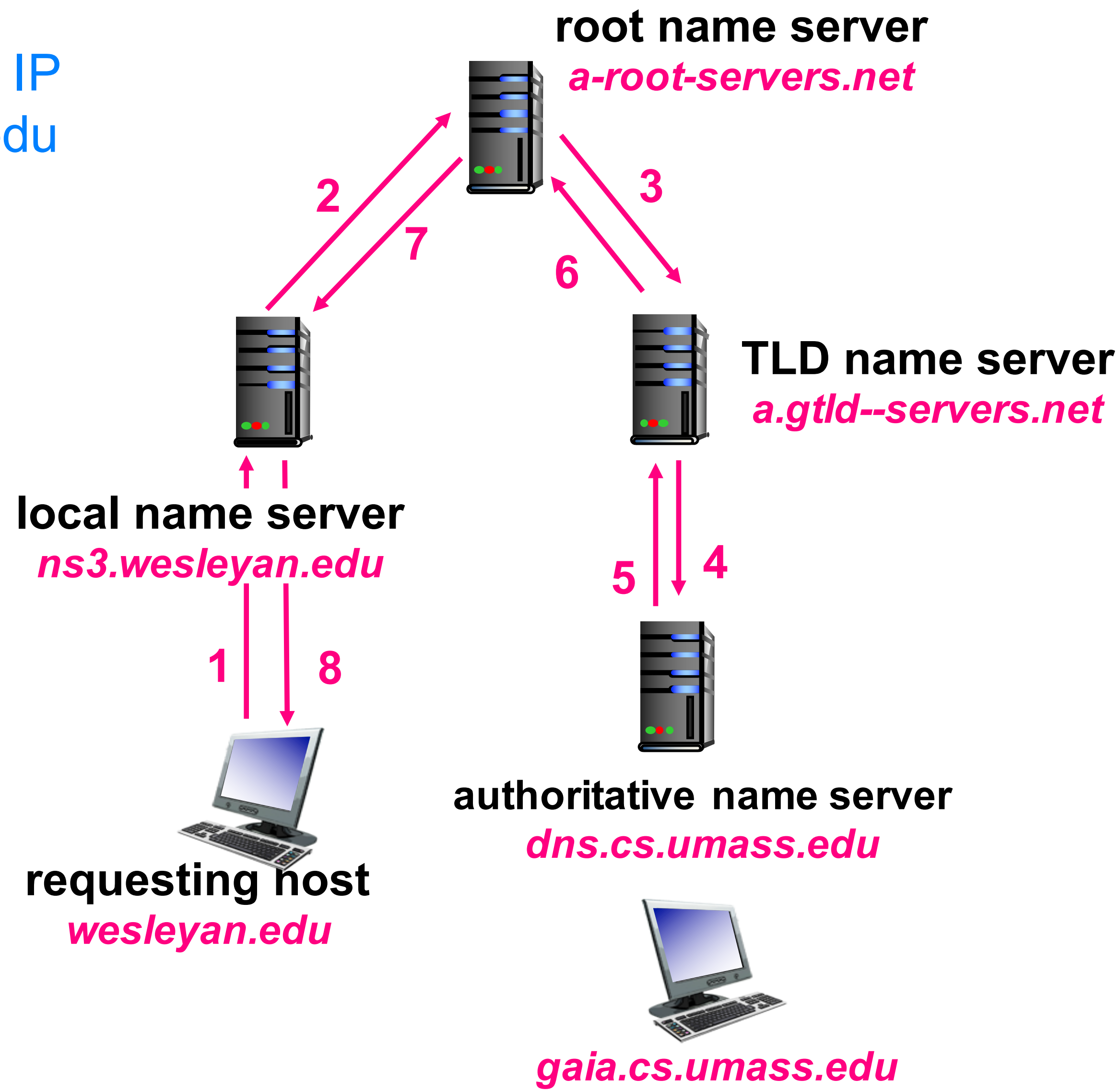
Reduces load on other servers

Recursive name resolution

Host at wesleyan.edu wants IP address for gaia.cs.umass.edu

Recursive query

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy
- "Let me ask about that name for you"



Q: why is this useful?

Reduces load on requesting server

Domain Name System

PROTOCOL

DNS resource records (RR)

DNS is distributed database

- returns RRs in response to queries

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

What you pass to index in on — What is returned

type=A/AAAA

- name is hostname
- value is IPv4 address (IPv6 for AAAA)

type=NS

- name is domain (e.g., foo.com)
- value is hostname of authoritative name server for this domain

type=CNAME

- name is alias name for some “canonical” (the real) name
- value is canonical name
- www.ibm.com is really servereast.backup2.ibm.com or ibm.com is really www.ibm.com

type=MX

- value is name of mailserver associated with name

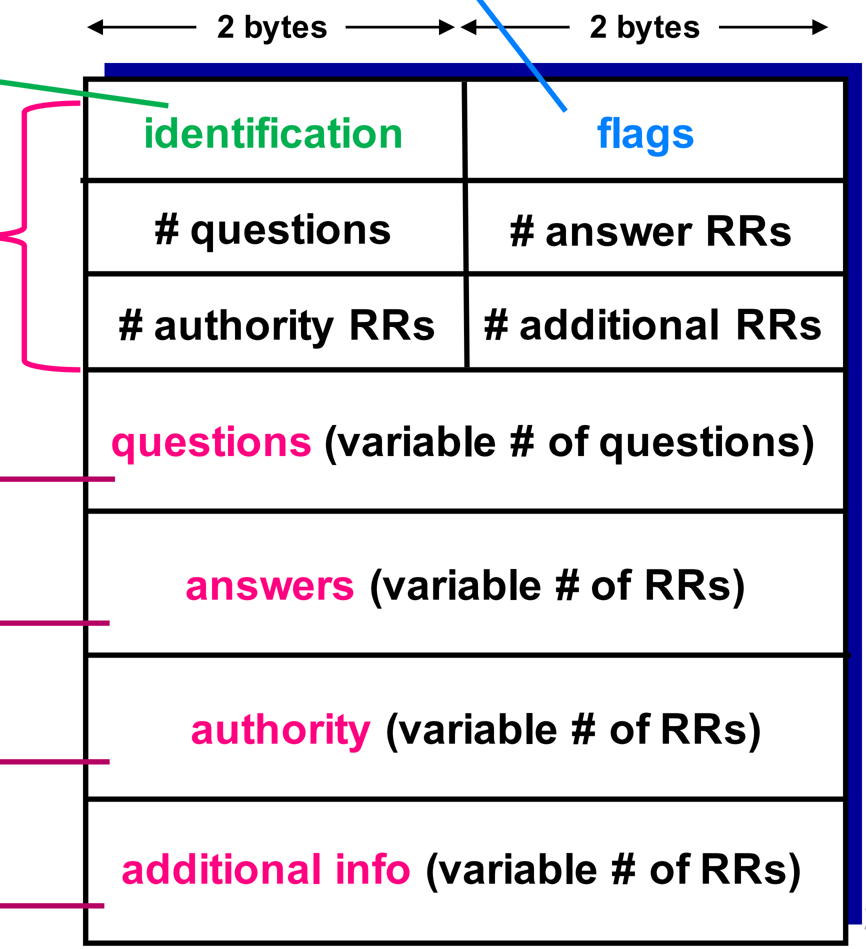
DNS protocol message format

Format of messages exchanged with DNS servers

flags
- query or reply
- recursion desired
- recursion available
- reply is authoritative

identification: 16 bit # for query, reply to query uses same #

message header



name, type fields for a query

RRs in response to query

records for authoritative servers

info that may be helpful (e.g., IP address of mail server)

Caching and updating records

Once (any) name server learns mapping, it caches mapping

- cache entries timeout (disappear) after some time (TTL)
- marked as "non-authoritative" mapping with address of authoritative server
- TLD servers typically cached in local name servers
  - thus root name servers not often visited

Cached entries may be out-of-date (best effort)

- if host changes IP address
  - may not be known Internet-wide until all TTLs expire

Inserting records into DNS

Example

- new startup "Network Utopia"

Register name networkuptopia.com at DNS registrar

- e.g., Network Solutions, delegated by ICANN
- need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
- registrar inserts two RRs into .com TLD (top-level) server

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

Create

- authoritative server Type A record for www.networkuptopia.com
- type MX record for networkutopia.com

DNS Propagation

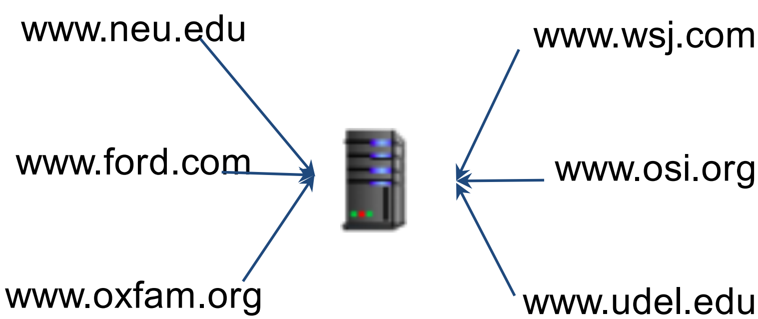
New domain names can take up to 72 hours to be accessible.

Why?

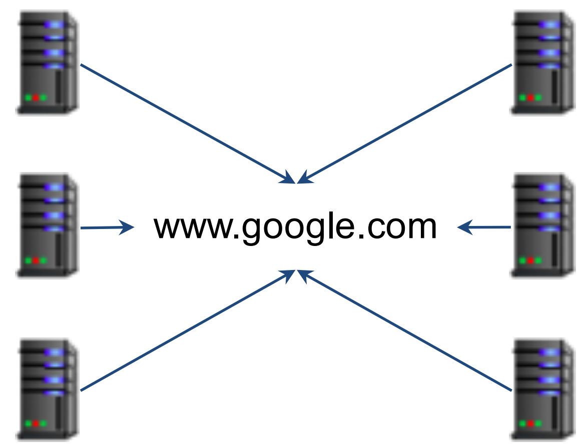
- new domain names require
  - authoritative server for domain and TLD name server to be updated
- name servers in hierarchy cache Root and TLD information
  - TLD servers have 2-3 day TTLs to prevent overuse
  - if old TLD info is cached at any level, “Does not exist” returned

DNS Aliasing and Load Balancing

One machine (IP address) can have multiple domain names



One domain name can point to multiple hosts (IP addresses)



CDNs use these properties to deliver content at scale while offering geographic, ISP, end system , … differentiation.

Domain Name System

DIG AND WIRESHARK

dig wesleyan.edu

```
> dig wesleyan.edu

; <<>> DiG 9.8.3-P1 <<>> wesleyan.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65061
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;wesleyan.edu.                  IN      A

;; ANSWER SECTION:
wesleyan.edu.           21493   IN      A       129.133.7.68

;; Query time: 2 msec
;; SERVER: 129.133.52.12#53(129.133.52.12)
;; WHEN: Sun Feb 18 08:25:59 2018
;; MSG SIZE  rcvd: 46
```

dig inria.fr

```
> dig inria.fr

; <<>> DiG 9.8.3-P1 <<>> inria.fr
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11551
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;inria.fr.                      IN      A

;; ANSWER SECTION:
inria.fr.               7169    IN      A       128.93.162.84

;; Query time: 8 msec
;; SERVER: 129.133.52.12#53(129.133.52.12)
;; WHEN: Sun Feb 18 08:22:23 2018
;; MSG SIZE  rcvd: 42
```

Wireshark for dig inria.fr query

```
▶ Frame 27: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▶ Ethernet II, Src: 78:4f:43:73:43:26 (78:4f:43:73:43:26), Dst: 129.133.176.1 (3c:8a:b0:1e:18:01)
▶ Internet Protocol Version 4, Src: 129.133.187.174, Dst: 129.133.52.12
▶ User Datagram Protocol, Src Port: 51519 (51519), Dst Port: 53 (53)
▼ Domain Name System (query)
     [Response In: 28]
     Transaction ID: 0x2d1f
   ▶ Flags: 0x0100 Standard query
     Questions: 1
     Answer RRs: 0
     Authority RRs: 0
     Additional RRs: 0
   ▼ Queries
      ▼ inria.fr: type A, class IN
           Name: inria.fr
           [Name Length: 8]
           [Label Count: 2]
           Type: A (Host Address) (1)
           Class: IN (0x0001)
```

Wireshark for dig inria.fr response

```
▶ Frame 28: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
▶ Ethernet II, Src: 129.133.176.1 (3c:8a:b0:1e:18:01), Dst: 78:4f:43:73:43:26 (78:4f:43:73:43:26)
▶ Internet Protocol Version 4, Src: 129.133.52.12, Dst: 129.133.187.174
▶ User Datagram Protocol, Src Port: 53 (53), Dst Port: 51519 (51519)
▼ Domain Name System (response)
    [Request In: 27]
    [Time: 0.007877000 seconds]
    Transaction ID: 0x2d1f
  ▶ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    ▼ inria.fr: type A, class IN
        Name: inria.fr
        [Name Length: 8]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▼ Answers
    ▼ inria.fr: type A, class IN, addr 128.93.162.84
        Name: inria.fr
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 7169
        Data length: 4
        Address: 128.93.162.84
```

What really happens when you type URL?

1. DNS query
   - sent to get ip address for hostname over UDP

2. TCP socket opened to ip address

3. HTTP msgs sent over TCP socket

4. TCP socket shutdown

Load inria.fr webpage

129.133.188.34	129.133.52.11	DNS	68	Standard query 0xe8ca A inria.fr
129.133.52.11	129.133.188.34	DNS	84	Standard query response 0xe8ca A 128.93.162.84
129.133.188.34	129.133.52.11	DNS	68	Standard query 0x67ba AAAA inria.fr
JuniperN_1e:18:01	Broadcast	ARP	64	Gratuitous ARP for 129.133.176.1 (Request) [ETHERN
129.133.52.11	129.133.188.34	DNS	119	Standard query response 0x67ba
129.133.188.34	128.93.162.84	TCP	74	33302 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
128.93.162.84	129.133.188.34	TCP	74	http > 33302 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=
129.133.188.34	128.93.162.84	TCP	66	33302 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSv
129.133.188.34	128.93.162.84	HTTP	382	GET / HTTP/1.1
128.93.162.84	129.133.188.34	TCP	66	http > 33302 [ACK] Seq=1 Ack=317 Win=15616 Len=0 T
128.93.162.84	129.133.188.34	HTTP	558	HTTP/1.1 301 Moved Permanently (text/html)
129.133.188.34	128.93.162.84	TCP	66	33302 > http [ACK] Seq=317 Ack=493 Win=30336 Len=0
129.133.188.34	128.93.162.84	TCP	66	33302 > http [FIN, ACK] Seq=317 Ack=493 Win=30336
128.93.162.84	129.133.188.34	TCP	66	http > 33302 [FIN, ACK] Seq=493 Ack=317 Win=15616
129.133.188.34	128.93.162.84	TCP	66	33302 > http [ACK] Seq=318 Ack=494 Win=30336 Len=0
129.133.188.34	129.133.52.11	DNS	72	Standard query 0x52a5 A www.inria.fr
129.133.188.34	129.133.52.11	DNS	72	Standard query 0x1f32 AAAA www.inria.fr
128.93.162.84	129.133.188.34	TCP	66	http > 33302 [ACK] Seq=494 Ack=318 Win=15616 Len=0
129.133.52.11	129.133.188.34	DNS	142	Standard query response 0x1f32 CNAME ezp3.inria.f
129.133.52.11	129.133.188.34	DNS	107	Standard query response 0x52a5 CNAME ezp3.inria.f
129.133.188.34	128.93.162.84	TCP	74	36018 > https [SYN] Seq=0 Win=29200 Len=0 MSS=1460
128.93.162.84	129.133.188.34	TCP	74	https > 36018 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len
129.133.188.34	128.93.162.84	TCP	66	36018 > https [ACK] Seq=1 Ack=1 Win=29312 Len=0 TS
129.133.188.34	128.93.162.84	TLSv1.2	255	Client Hello
128.93.162.84	129.133.188.34	TCP	66	https > 36018 [ACK] Seq=1 Ack=190 Win=15616 Len=0

Domain Name System

VULNERABILITIES

DNS trust

Without DNS, how would you get to any websites?

- your email…banks...government/municipal services…
- when you enter bankofamerica.com you expect to go to BoA site

DNS is the root of trust to the web.

- what can happen if DNS is compromised?
- how can DNS be compromised?

Attacking DNS

Distributed Denial-of-Service (DDoS) attacks

- bombard root servers with traffic
  - not successful to date, e.g., due to traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers: potentially more dangerous

Redirect attacks

- man-in-middle
  - intercept queries, malware on host / subvert DHCP server (home routers) to issue bogus DNS server
- DNS poisoning
  - send bogus replies to DNS server, which caches

Exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification

Turkey hijacks DNS to enable censorship

BGPMON Now part of OpenDNS

HOME BLOG ABOUT US PRODUCTS AND SERVICES CLIENT PORTAL

Turkey Hijacking IP addresses for popular Global DNS providers

Posted by Andree Toonk - March 29, 2014 - Hijack, News and Updates - 26 Comments

At BGPmon we see numerous BGP hijacks every single day, some are interesting because of the size and scale of the hijack or as we've seen today because of the targeted hijacked prefixes. It all started last weekend when the Turkish president ordered the censorship of twitter.com. This started with a block of twitter by returning false twitter IP addresses by Turk Telekom DNS servers. Soon users in Turkey discovered that changing DNS providers to Google DNS or OpenDNS was a good method of bypassing the censorship. But as of around 9am UTC today (Saturday March 29) this changed when Turk Telekom started to hijack the IP address for popular free and open DNS providers such as Google's 8.8.8.8, OpenDNS' 208.67.222.222 and Level3's 4.2.2.2. **BGP hijack** Using the Turk Telekom looking glass we can see that AS9121 (Turk Telekom) has specific /32 routes for these IP addresses. Since this is the most specific route possible for an IPv4 address, this route will always be selected and the result is that traffic for this IP address is sent to this new bogus route.

```
show router bgp routes 8.8.8.8
```

DNSSEC

Cryptographically sign critical Resource Records (RRs)

– resolvers can verify signature

Two new RRs

1. DNSKEY
   – name: zone domain name
   – value: public key for the zone
   – supports chain of trust to Root

2. RRSIG
   – name: the query -- [type, name]
   – value: signature of results to query
   – prevents spoofing