Networking 2

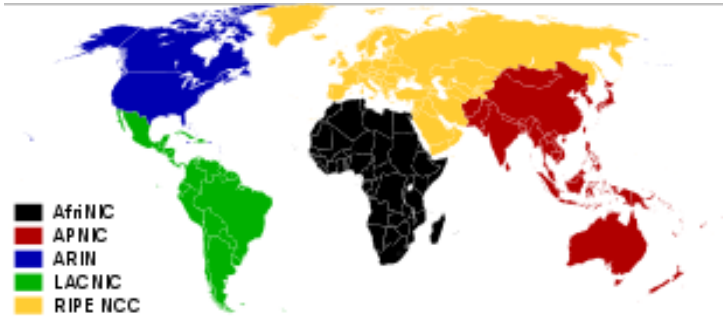
Portions courtesy Ellen Liu

Outline

- IP address allocation
- NAT (Network address translation)
- Routing configuration
- DHCP (Dynamic host configuration protocol)
- DNS (Domain name system)

IP Address Allocation

- A site can subdivide the address space assigned into subnets in any manner the site likes
- ICANN (Internet Corp. for Assigned Names & Numbers) delegates address blocks to 5 regional Internet registries
  - ARIN: north America
  - APNIC: Asia Pacific, Australia, New Zealand
  - AfriNIC: Africa
  - LACNIC: Central / south America
  - RIPE NCC: Europe
- Then national / regional ISPs,



Background: Private networks

Class A: 10.0.0.0 to 10.255.255.255 == 10.0.0.0/8

Class B: 172.16.0.0 to 172.31.255.255 == 172.16.0.0/12

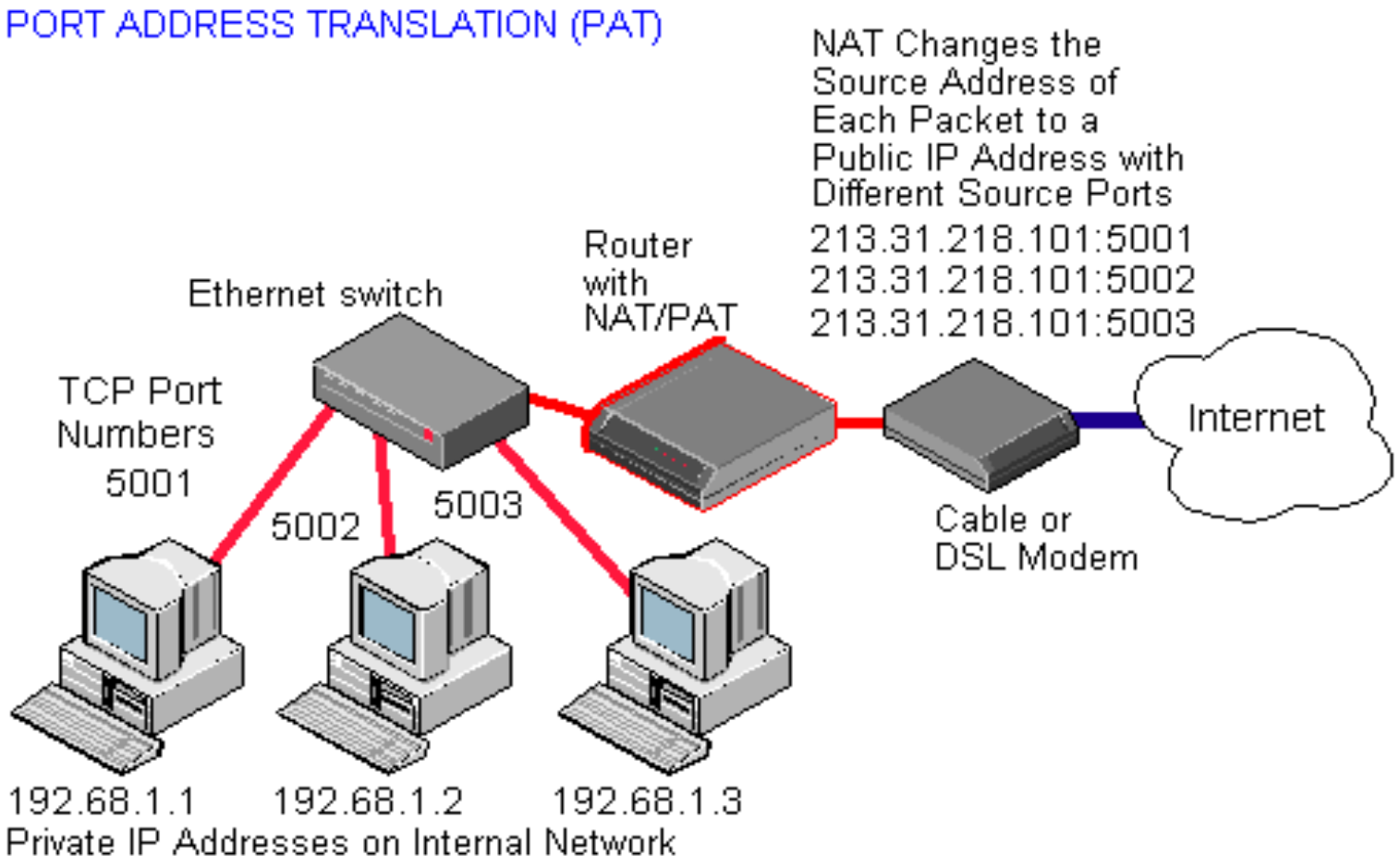
Class C: 192.168.0.0 to 192.168.255.255 == 192.168.0.0/16

- No one owns these networks
- These addresses will not be routed on the internet
- Good choice to use for a disconnected/private network

Network address translation (**NAT**)

- Local network uses just one IP address as seen from outside
- Router translates all IP addresses on incoming and outgoing packets to internal private addresses
- Hosts inside local network not explicitly addressable, visible by outside world
  - Mitigates the IP address shortage problem
  - Most residential ISPs only give customers one IP

NAT (Cont'd)



Why not NAT?

- If application encodes its IP address in application-level payload
  - Arguably poor design, but the customer is always right
- I want a service visible on the internet?
  - Example: Run a web server from home
  - Most NAT systems allow static routes
    - I can map port 80 from my router to my web box

How to configure routing/NAT?

- *Any* system with 2 network interfaces can serve as a router
  - This is basically what wireless tethering does
- Here we discuss the basics of doing this on Linux
- Dedicated boxes tend to have higher performance, energy efficiency (more specialized hardware), and easier UI
  - Even if they use Linux internally

Network Code

- Most lower-layer networking code is in the kernel, not in any application.
- Why?
  - Mostly performance: handle packet after an interrupt without a context switch
- Alternatives:
  - TCP/IP Offload: push some of the networking code into specialized hardware device
  - User-level drivers: historically inefficient, newer virtualization HW may improve this

Network configuration

- Linux provides a number of utilities that configure the in-kernel networking code
- ifconfig: bring up a network device, assign an IP address, netmask, etc.
- route: configure routing tables on the system
- iptables: configure firewall rules, forwarding between interfaces, NAT, etc.

Examples

- Suppose I want to configure a single network card to use IP 192.168.0.2/24

```
ifconfig eth0 192.168.0.2 netmask 255.255.255.0
```

- Linux generally names network interfaces eth0, eth1, etc.

Examples adapted from:
http://how-to.wikia.com/wiki/How_to_set_up_a_NAT_router_on_a_Linux-based_computer

Example (cont)

- Now I want to set up a router
  - One network card listening on my private network: 192.168.0.0/24
  - Another network card on the public network, provided IP address 130.245.153.3

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0
ifconfig eth1 130.245.153.3 netmask 255.255.255.0
```

Router, cont

```
# route
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.0.0 * 255.255.255.0 U 0 0 0 eth0
130.245.153.0 * 255.255.255.0 U 0 0 0 eth1
default 192.168.1.1 0.0.0.0 UG 0 0 0 eth0
```

- Packets originating from the router will (mostly) be delivered to the right interface
  - To 192.168.0.* goes to eth0
  - To 130.245.153.* goes to eth1
  - Everything else goes to eth0 (the private network)
- Problems?
  - Router won't send internet traffic to eth1
  - Router won't forward traffic from eth0 to eth1 (or do translation)

Default Route

- If I want to change the default route on the router box:

```
route add default gw 130.245.153.0
```

- "gw" == gateway (== router)
- Now packets go to eth1 if they aren't going to either local network

Set up NAT

```
modprobe iptable_nat
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth1 -j ACCEPT
```

- Pseudo-files in /proc configure the Linux kernel
- Iptables arguments:

-t nat	Operate on the nat table
-A POSTROUTING	Append to rule list (chain) named POSTROUTING
-o eth0	The packet is going to eth0
-i eth1	The packet came from eth1
-j MASQUERADE	If a packet matches this rule, jump to chain MASQ

Toward simpler network management

- In the previous examples, we manually assigned IP addresses
- What if a machine is powered off?
- Leaves the network? (laptop)
- Or just doesn't need to be running any world-visible services?

- Automation would be nice…

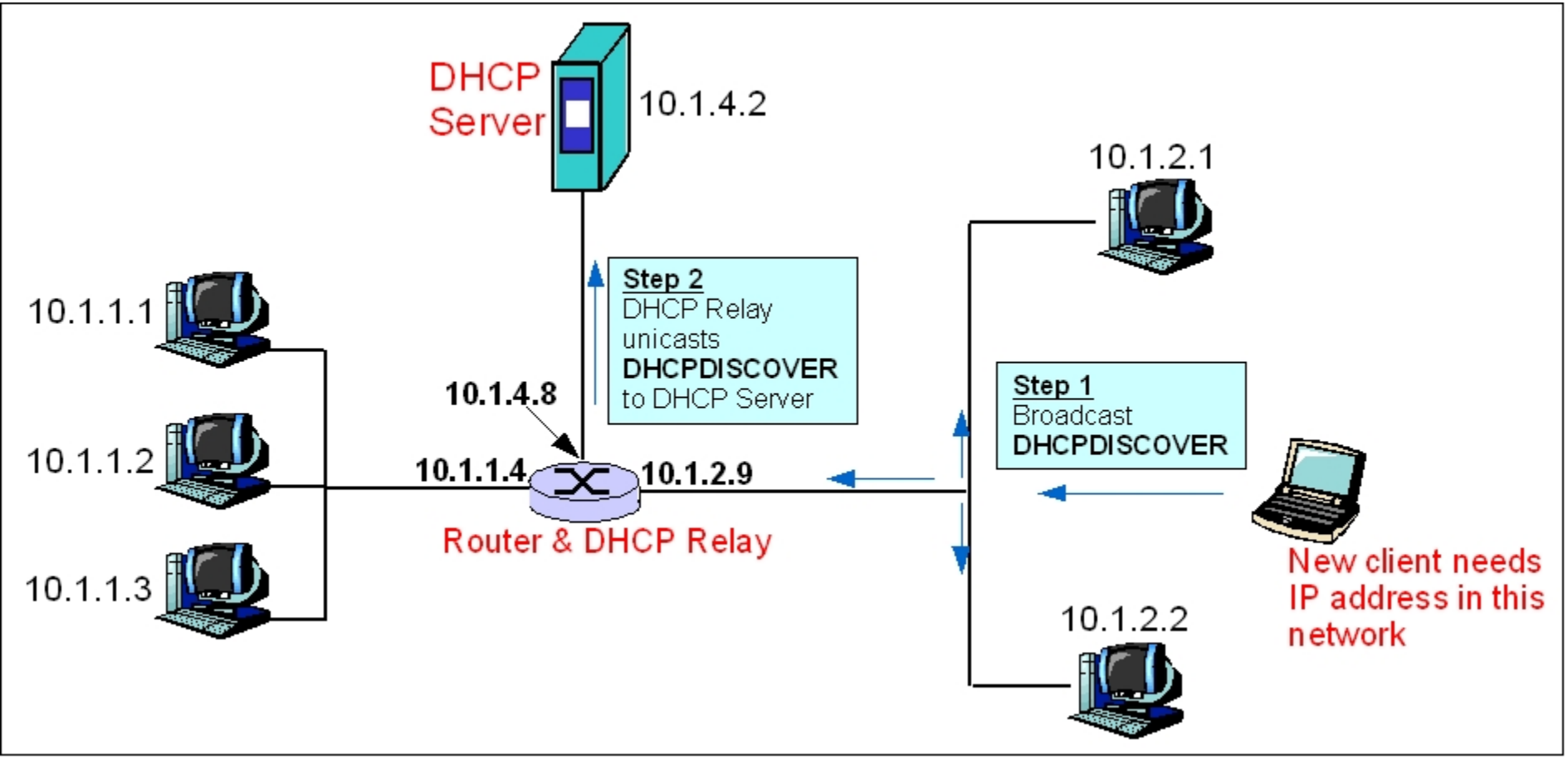
DHCP

- Dynamic host configuration protocol
  - Link layer protocol. Why?
    - No IP address yet…
- Server keeps a pool of available addresses
- When adding a new computer on a network, that computer can lease an IP address from server
  - Lease must be renewed periodically (generally daily)
  - When a lease expires, IP address goes back in pool, can be given to another computer

DHCP

- Leasable parameters include
  - IP addresses and netmasks
  - Default gateway (router)
  - DNS name servers
  - Syslog hosts, NTP servers, proxy servers, etc.
- Also handy for pushing other network configuration to clients
  - PXE implemented using DHCP
- DHCP server can also assign specific IP addresses to MAC addresses

DHCP (Cont'd)



Many wireless routers include DHCP server software.
There are open source DHCP servers available also.

Linux DHCP server

- Creatively named dhcp3
  - Ubuntu has a new udhcpd
- Generally configured using a file named /etc/dhcpd.conf
- Places leases in a file called /etc/dhcpd.leases

Configuration examples

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.254;
  option subnet-mask 255.255.255.0;
  option domain-search "example.com";
  option domain-name-servers 192.168.1.1;
  option time-offset -18000; # Eastern Standard Time
  range 192.168.1.10 192.168.1.100;
}
```

- Allocates IP addresses between .10 and .100
- Pushes the other configuration options to client

Configuration example 2

- Suppose I want to always assign a given IP and hostname to a machine
  - Why?

```
host apex {
  option host-name "apex.example.com";
  hardware ethernet 00:A0:78:8E:9E:AA;
  fixed-address 192.168.1.4;
}
```

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- **IP address** (32 bit) - used for addressing datagrams
- **"name",** e.g., www.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol.*

  Host, routers, name servers communicate to *resolve* names (address/name translation)

- >100 RFCs, popular implementations: BIND, MS DNS, NSD, Unbound

DNS

DNS services

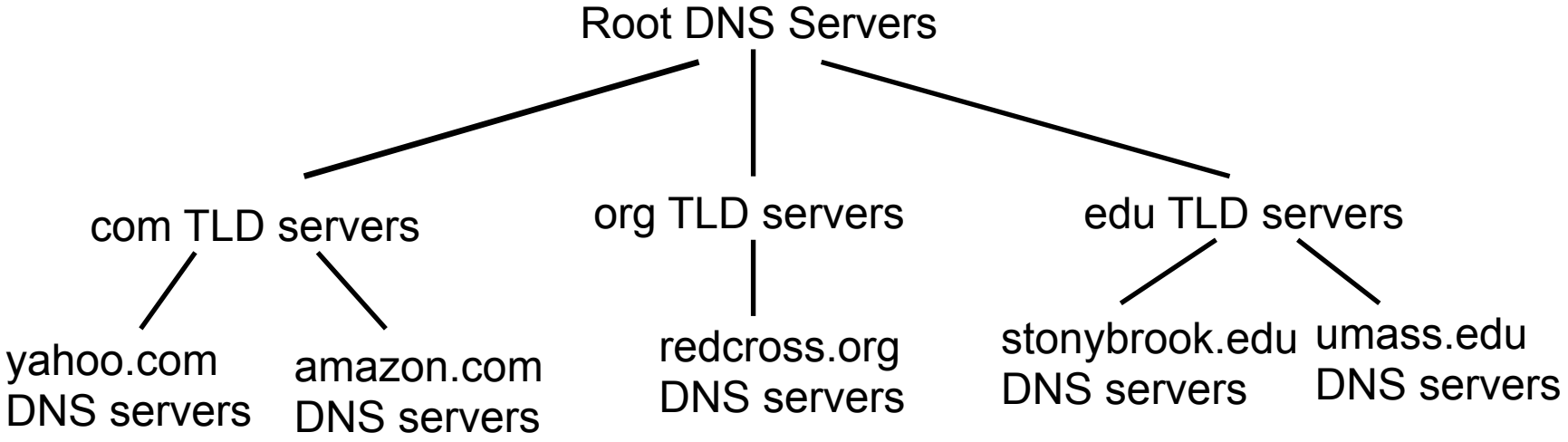
- hostname to IP address translation
- host aliasing
  - Canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

Distributed, Hierarchical Database

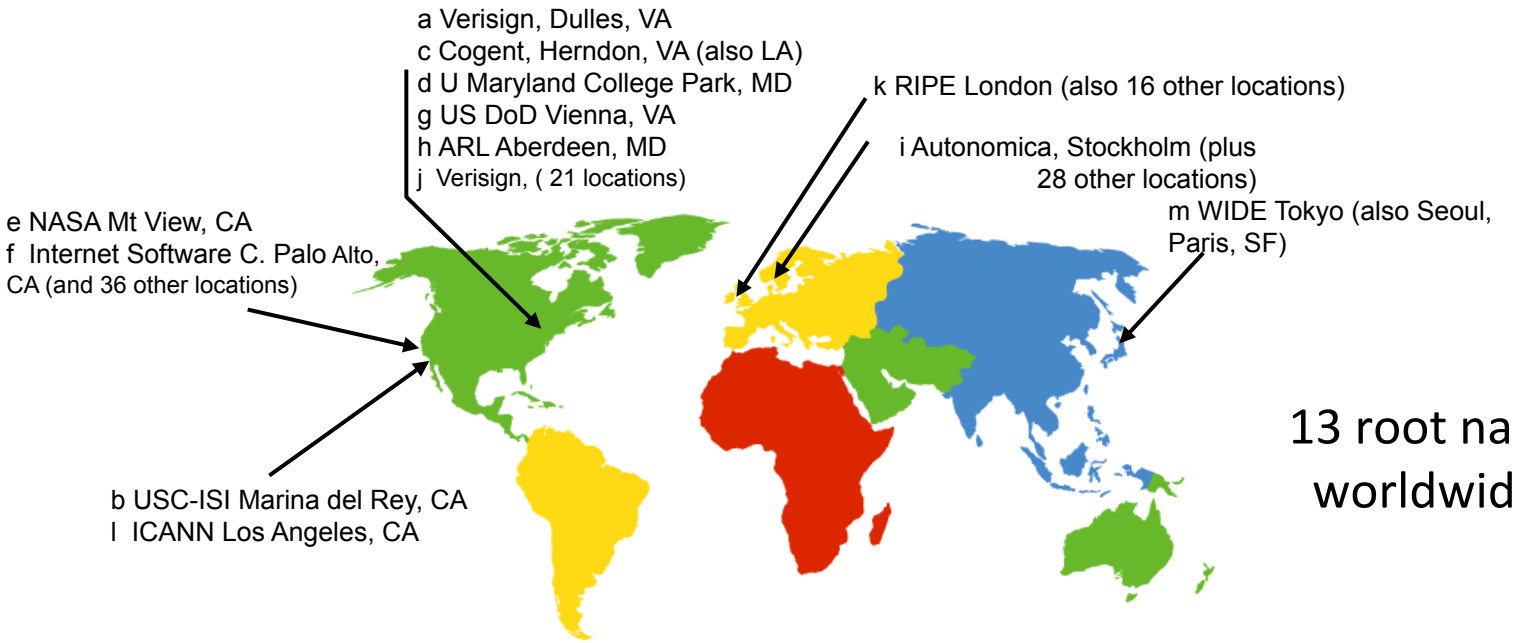


Client wants IP for www.amazon.com; 1st approximation:

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

DNS: Root name servers

- contacted by local name servers that can not resolve a name
- root name server:
  - contacts authoritative name server if name not known
  - gets mapping
  - returns mapping to local name server



13 root name servers worldwide

TLD and Authoritative Servers

- Top-level domain (TLD) servers: ~20 of them
  - Responsible for generic TLDs (gTLDs): com, org, net, edu, gov, mil, etc
    - VeriSign administers the com TLD, Educause for edu TLD
  - And country code domains (ccTLDs) :~250 of them
    - all top-level country domains uk, fr, ca, jp…

- Authoritative DNS servers: organization's DNS servers, providing authoritative hostname to IP mappings (called resource records) for organization's servers (e.g., Web, mail).
  - can be maintained by organization or service provider

Local Name Server

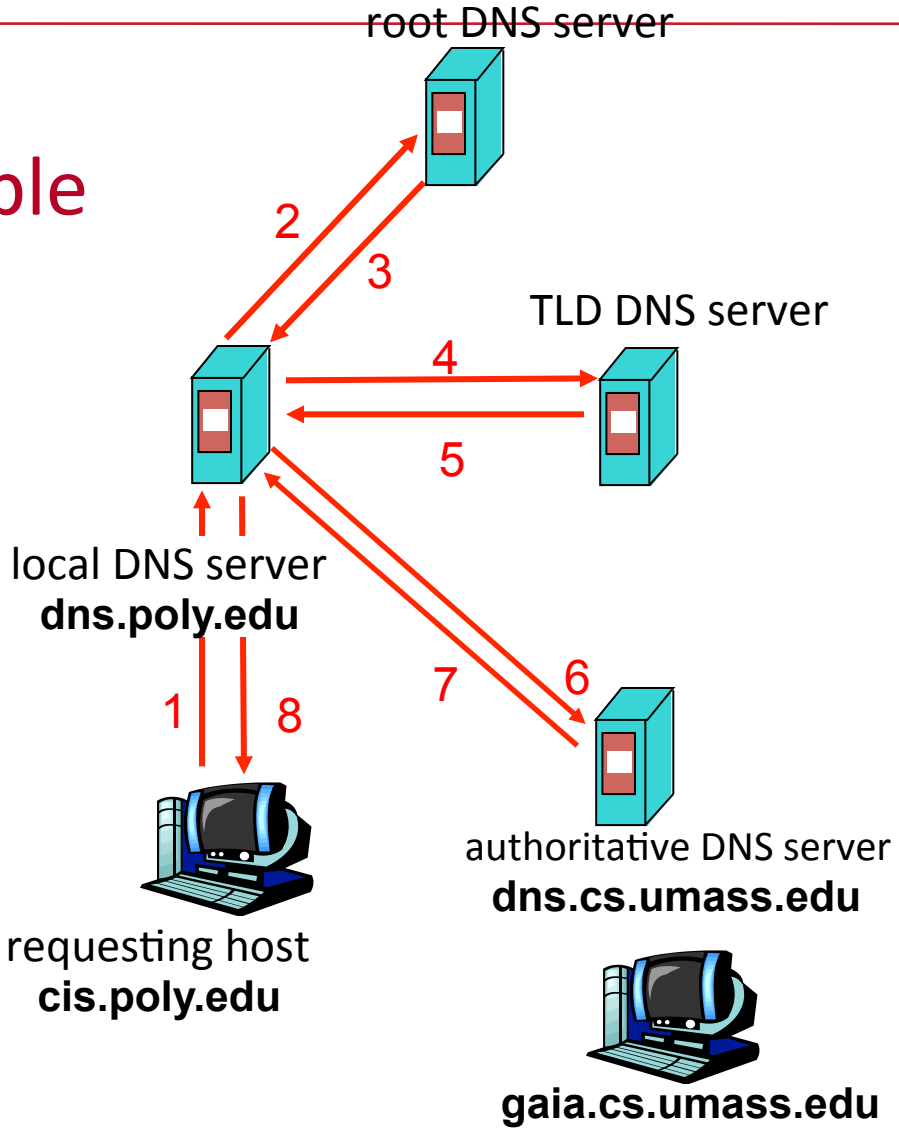
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one.
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

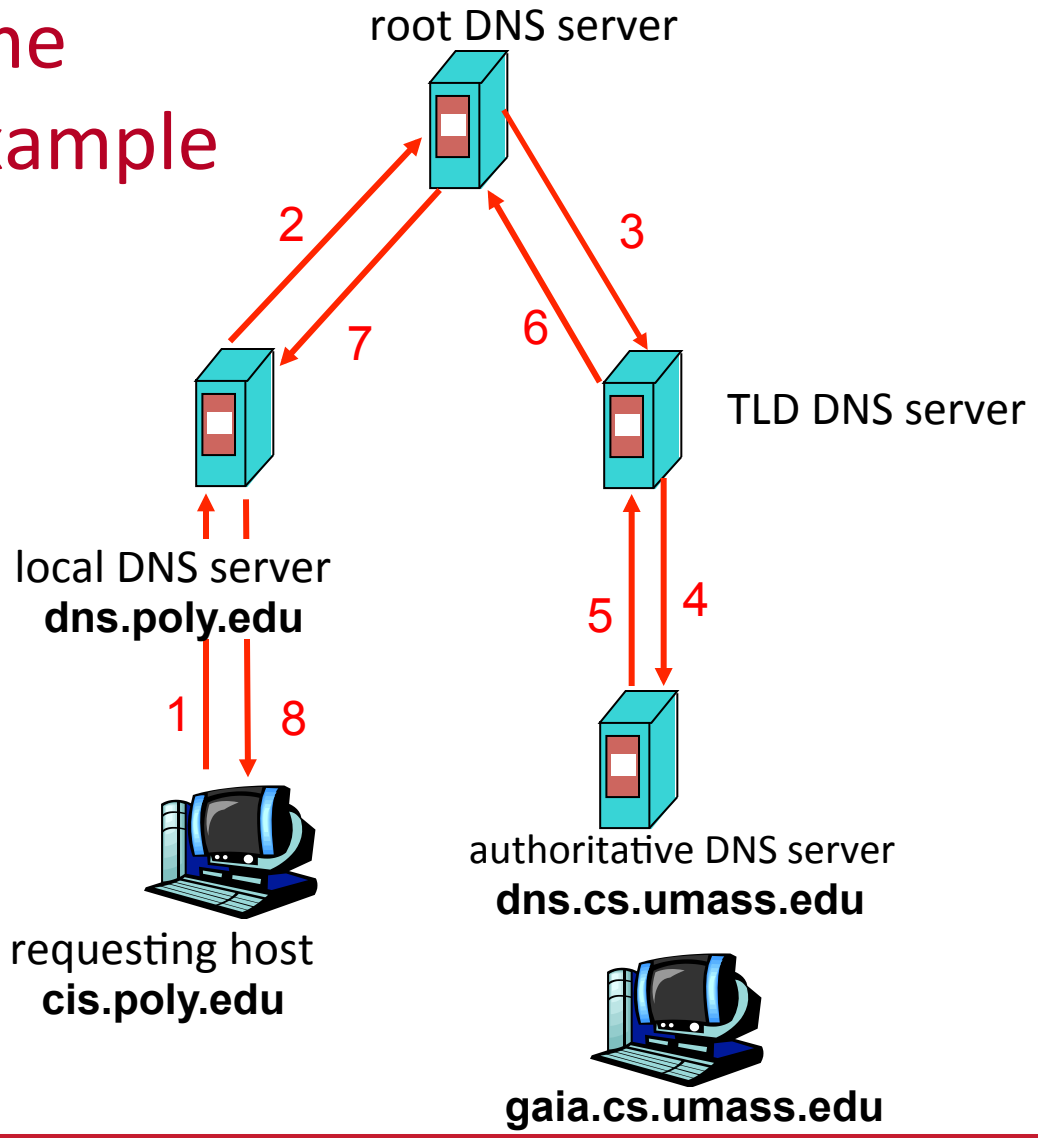
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?



Local vs. Public

- A local DNS server (i.e., a caching server for your internal network), must support recursive queries
  - Each system's resolver won't do this
- I wouldn't allow public access to a caching server. Why not?
  - Mostly to prevent denial of service

Advice for a public DNS server

- Configure it to not service recursive queries
  - I answer for my domain, and my domain only
- Again, reduce denial-of-service risk
- Caching servers can make their own recursive requests
- Point: you probably want 2 different servers (internal vs. external)

Resource Records

- Information about one host in a standardized format
  - Ensures interoperability across implementations
- Example: map hostname “ns” to IP 192.168.1.10

```
ns IN A 192.168.1.10
```

Resource Record format

[name] [ttl] [class] type data

```
ns IN A 192.168.1.10
```

ns is the host name

class IN == internet

type A == address (name->addr. translation)

Other record types

- Start of Authority (SOA) – declare a zone, assert ownership of it
- Name Server (NS) – identify authoritative name servers
- Address (A) – name to address
- Pointer (PTR) – address to name
- Mail Exchanger (MX) – mail server
- CNAME – aliases for a host

FQDN

- What is it?
  - Fully qualified domain name
  - Eg., mail.cs.stonybrook.edu (vs mail)

DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited
- update/notify mechanisms
  - RFC 2136
  - http://datatracker.ietf.org/wg/dnsind/charter/
- Current IETF DNS working group:
  - http://datatracker.ietf.org/wg/dnsext/charter/
  - Many new areas: authentication and DNSSEC, internationalization, IPv6

DNS records

DNS: distributed db storing resource records (RR)

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

- Type=A
  - **name** is hostname
  - **value** is IP address
- 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 `www.ibm.com` is really `servereast.backup2.ibm.com`
  - **value** is canonical name
- Type=MX
  - **value** is name of mailserver associated with **name**

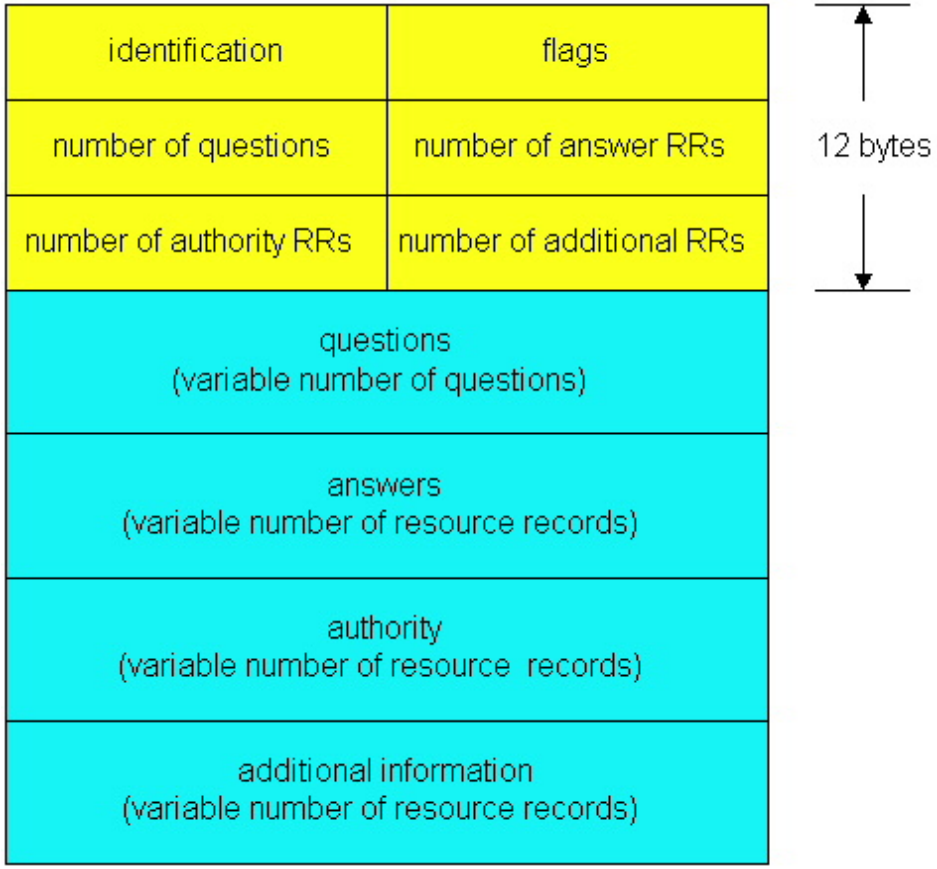
DNS protocol, messages

DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query. Reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

- **Non-authoritative** means the answer of a query is from cache, doesn't know if the data is still valid



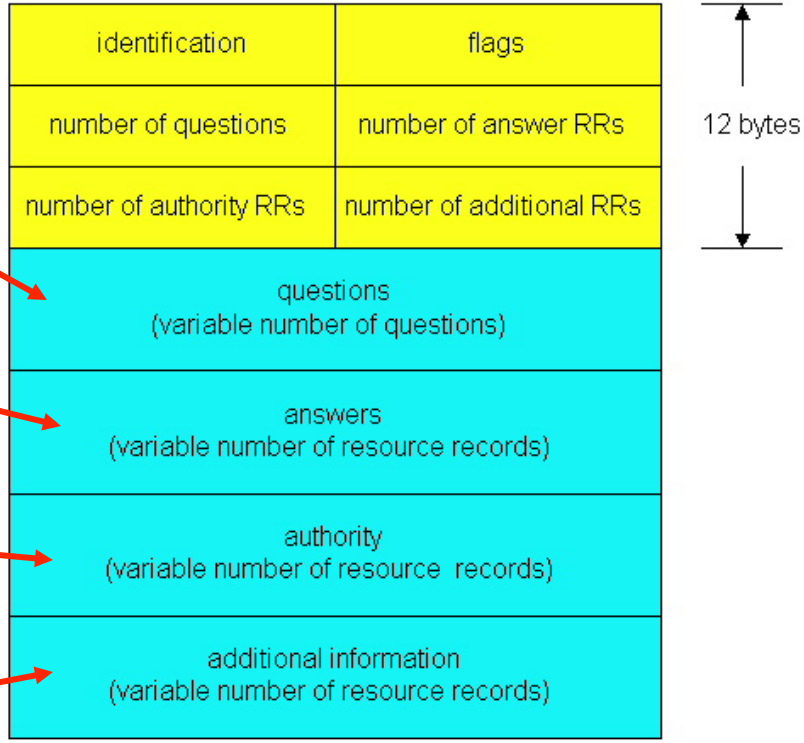
DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



Time to live (TTL)

- Long at the roots (~42 days)
- .edu, stonybrook.edu (~2 days)
- www.stonybrook.edu (~1 day)

- Of course, an administrator can set whatever she likes, but these are reasonable defaults

Unsolicited Advice

- Get your "upstream" DNS from google
  - 8.8.8.8 and 8.8.4.4
- It is probably faster and more reliable than the caching service your ISP offers

Linux DNS server: Bind

- Can run in several modes:
  - Caching upstream DNS records
    - Accelerate local lookups, save bandwidth
  - Primary Master: Answers queries from its local database, acts as master for domain
    - If you register example.com, you need a master for example.com
  - Secondary Master: Repeats answers from a master
    - Mainly needed if primary is overloaded

Bind configuration

- Main abstraction: Zone
  - A domain without subdomains
  - Or, a suffix this server is responsible for
    - E.g., example.com running on 192.168.1.0/24
      - eg.example.com is a different zone
- You need a forward and reverse zone
  - DNS servers also map IPs back to hostnames
- A server hosts multiple zones
  - Each zone has a file under /etc/bind/

Example: Primary Server

- Add forward and reverse entries to /etc/bind/named.conf.local

```
zone "example.com" {
        type master;
        file "/etc/bind/db.example.com";
};

zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.192";
};
```

https://help.ubuntu.com/12.04/serverguide/dns-configuration.html

Contents of db.example.com

```
;
; BIND data file for example.com
;
$TTL 604800 @ IN SOA example.com. root.example.com. (
                                2 ;
                                Serial 604800 ;
                                Refresh 86400 ;
                                Retry 2419200 ;
                                Expire 604800 ) ; Negative Cache TTL
        IN A 192.168.1.10
;
@ IN NS ns.example.com.
@ IN A 192.168.1.10
@ IN AAAA ::1
ns IN A 192.168.1.10
```

Contents of db.192

```
;
; BIND reverse data file for local loopback interface
;
$TTL 604800
@ IN SOA ns.example.com. root.example.com. (
                                2 ;
                                Serial 604800 ;
                                Refresh 86400 ;
                                Retry 2419200 ;
                                Expire 604800 ) ; Negative Cache TTL
;
@ IN NS ns.
10 IN PTR ns.example.com.
; also list other computers
21 IN PTR box.example.com.
```

Linux DNS client configuration

- Can be pushed by DHCP
- Or configured locally in /etc/resolv.conf
  - Or sometimes /etc/resolv.conf is populated by entries in /etc/network/interfaces

resolv.conf

- Generally 2 entry types:

```
nameserver 8.8.8.8
search cs.stonybrook.edu stonybrook.edu
```

- Nameserver – IP of the DNS server to use
  - Generally a caching server
  - Why not a hostname?
- Search: suffixes to append if you just get a host
  - Auto-map mail to mail.cs.stonybrook.edu

Trick: Load balancing with DNS

- There isn't just one web server behind www.google.com
- Suppose there are 100 servers. How to evenly distribute client load?
- Each DNS query gets a different answer
  - Round-robin through the different hosts

Example

- Round-robin www to .1--.3

```
www  IN A 192.168.0.1
     IN A 192.168.0.2
      IN A 192.168.0.3
```

Active Directory

- Next most popular DNS server
  - After BIND
- Active Directory also includes a number of other services, including user management
- You probably want one of these if you are setting up a Windows network

Best practices for external DNS

- At least 2 authoritative servers
- Each on a separate network and power circuit
- In other words, have a DNS server off site
  - So your network stays up if someone trips on the power cord to your server rack

Zone Transfers

- Protocol by which DNS master updates slave
  - Exchange cryptographically signed messages
    - Why?
  - Keys called TSIG

Anti-Phishing

- Domain Keys Identified Mail (DKIM)
  - Basically, a mail server can now sign all outgoing messages with a private key
  - Public key distributed through DNS
  - Receiving server checks mail signature against public key
  - Detect mail that claims to come from stonybrook.edu but really isn't

DNSSEC

- Big topic: Basically add digital signatures to DNS records
- Underlying issue: As described, you can't really verify the integrity of any DNS response
- Idea: Integrate public key crypto to sign each message

Review of DNS server types

- Does the server speak for this zone?
  - Authoritative vs. caching
- If authoritative, where does it get its information?
  - Master vs. slave
- Does is say "I don't know"?
  - Recursive vs. non-recursive