14. How the Internet Works (Part 2) & How the Web Works (Part 1)

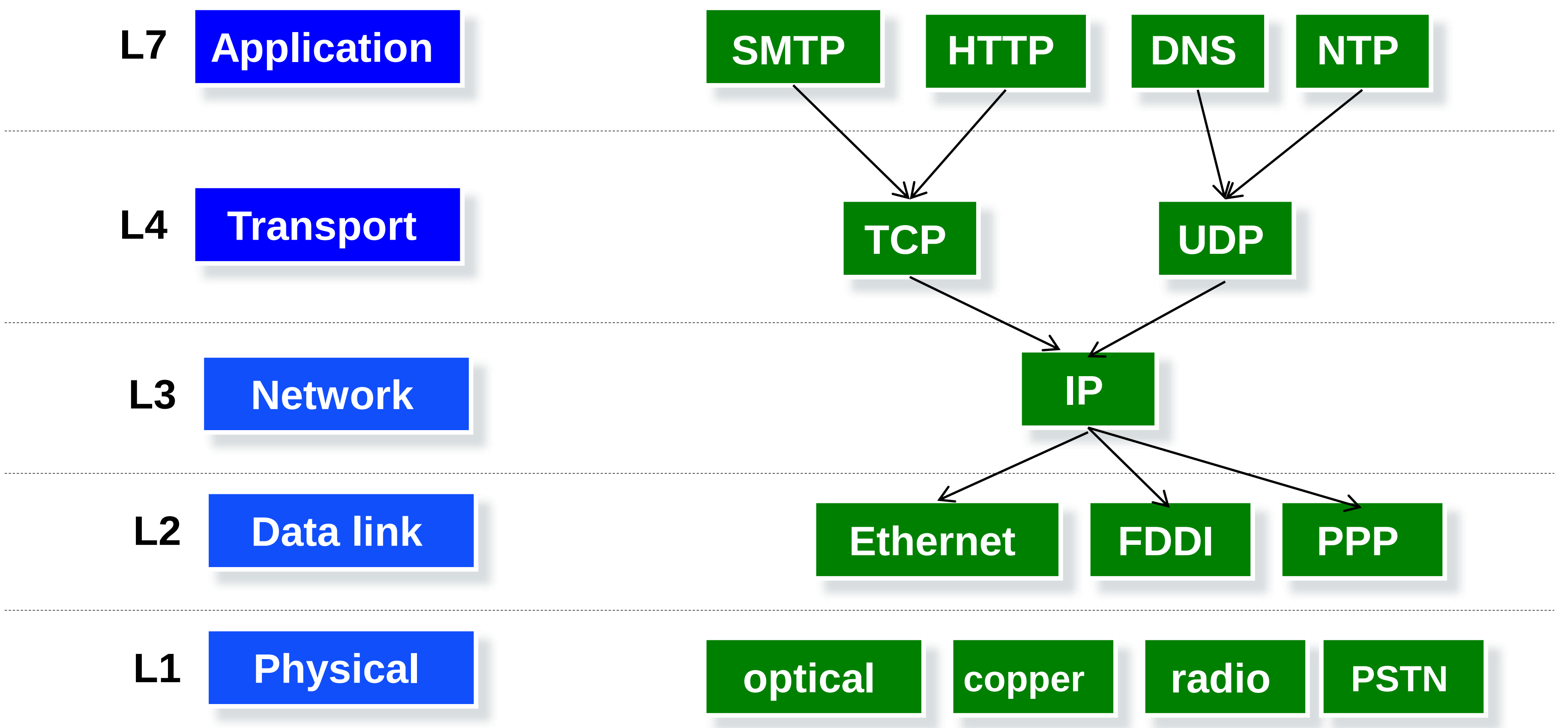
Blase Ur and David Cash
(Some slides borrowed from Ben Zhao)

February 11th, 2022

CMSC 23200 / 33250

THE UNIVERSITY OF CHICAGO

Protocols at different layers



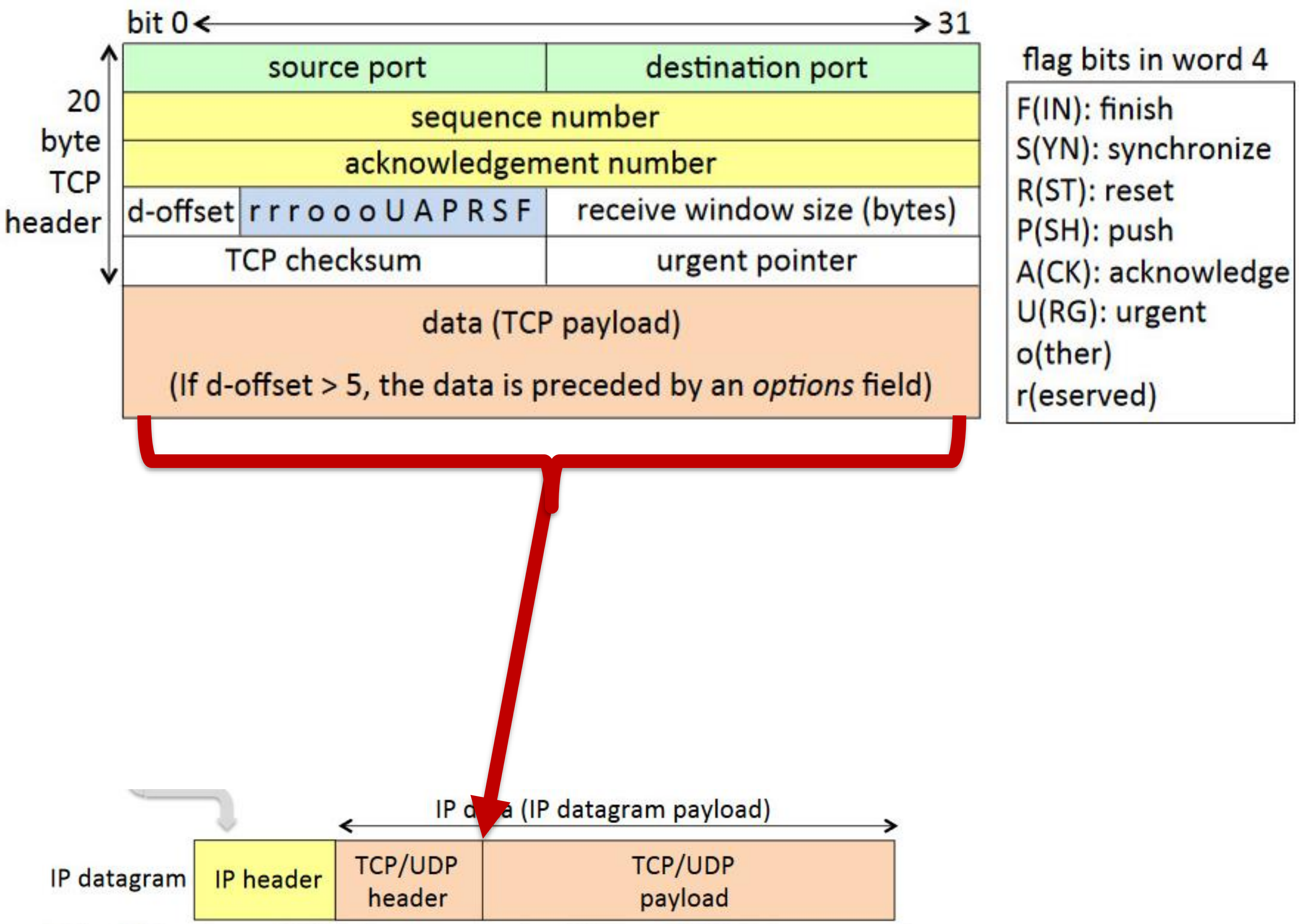
Goal: Get ALL of the data to its destination

Solution (Protocol): TCP at the transport layer

TCP (Transmission Control Protocol)

- Multiplexes between services
- Multi-packet connections
- Handles loss, duplication, & out-of-order delivery
  — all received data ACKnowledged
- Flow control
  — sender doesn't overwhelm recipient
- Congestion control
  — sender doesn't overwhelm network

TCP header



bit 0
31
20 byte TCP header
source port
destination port
sequence number
acknowledgement number
d-offset
r r r o o o U A P R S F
receive window size (bytes)
TCP checksum
urgent pointer
data (TCP payload)
(If d-offset > 5, the data is preceded by an *options* field)
flag bits in word 4
F(IN): finish
S(YN): synchronize
R(ST): reset
P(SH): push
A(CK): acknowledge
U(RG): urgent
o(ther)
r(eserved)
IP datagram
IP header
TCP/UDP header
TCP/UDP payload

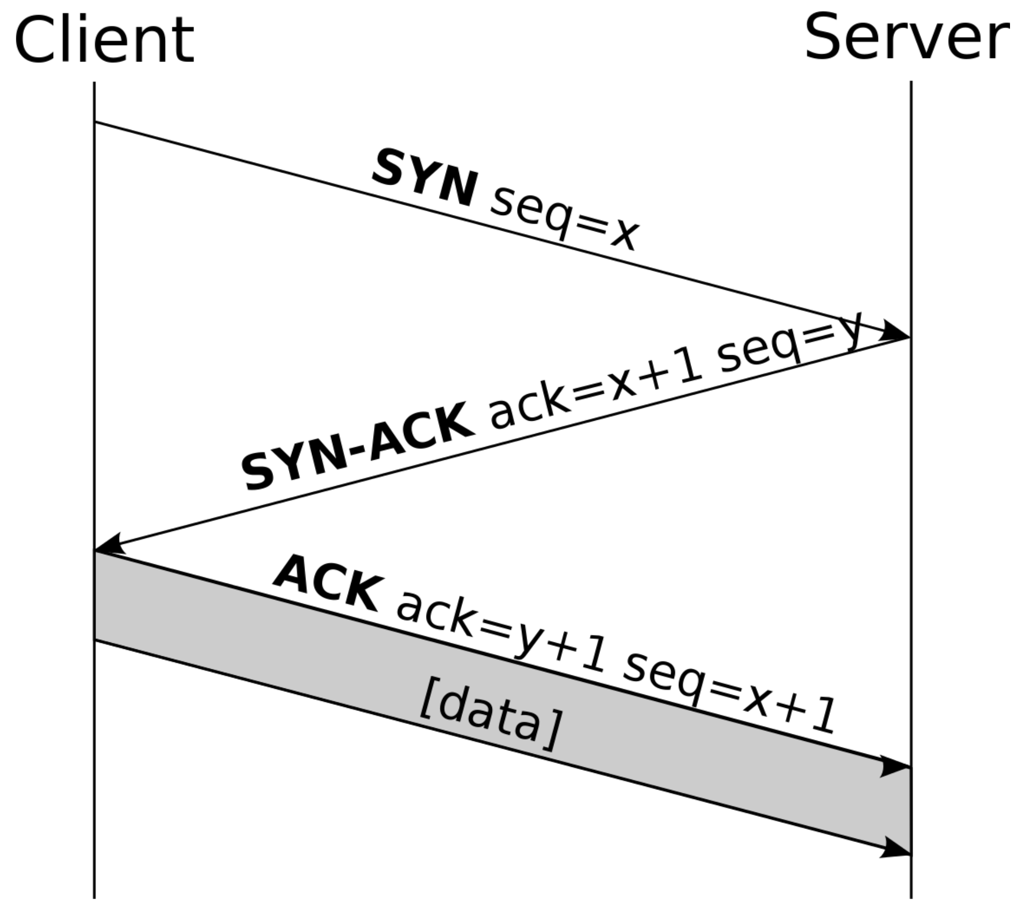
Common TCP (Default) Ports

- 22: SSH
- 25: SMTP
- 53: DNS
- 67, 68: DHCP
- 80: HTTP
- 143: IMAP
- 443: HTTPS
- Ports 49152-65535 are used by client programs

TCP connections

- Explicit connection setup & teardown
- Explicit control flags (e.g., SYN, ACK, FIN, RST)
- Sequence numbers — reliability & ordering

Setup: 3-way handshake



Source: Wikimedia commons

TCP Sequence Numbers

- Bytes in a TCP sequence are numbered (and acked)

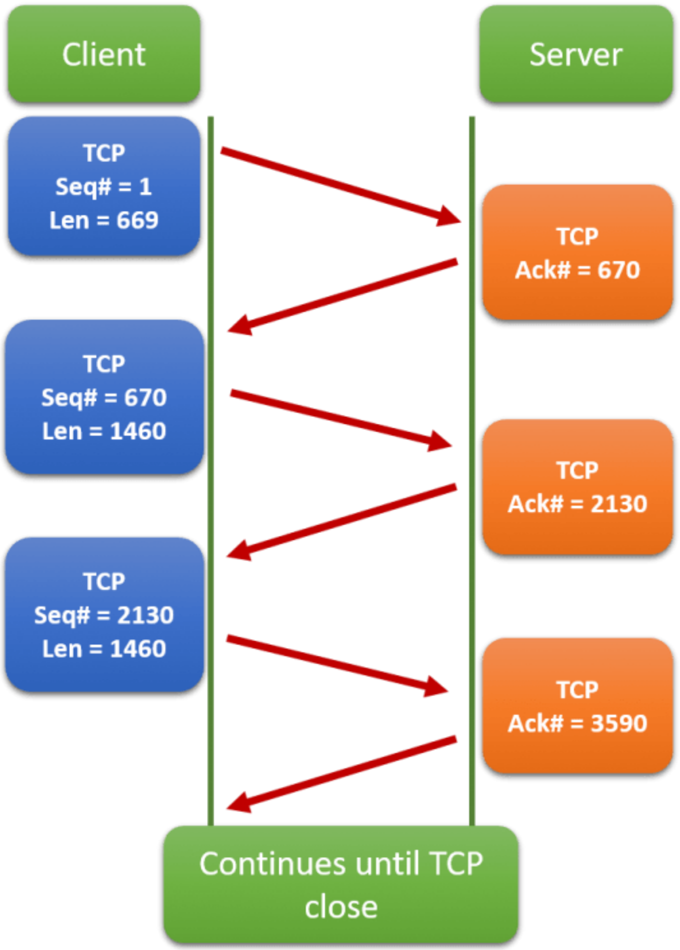
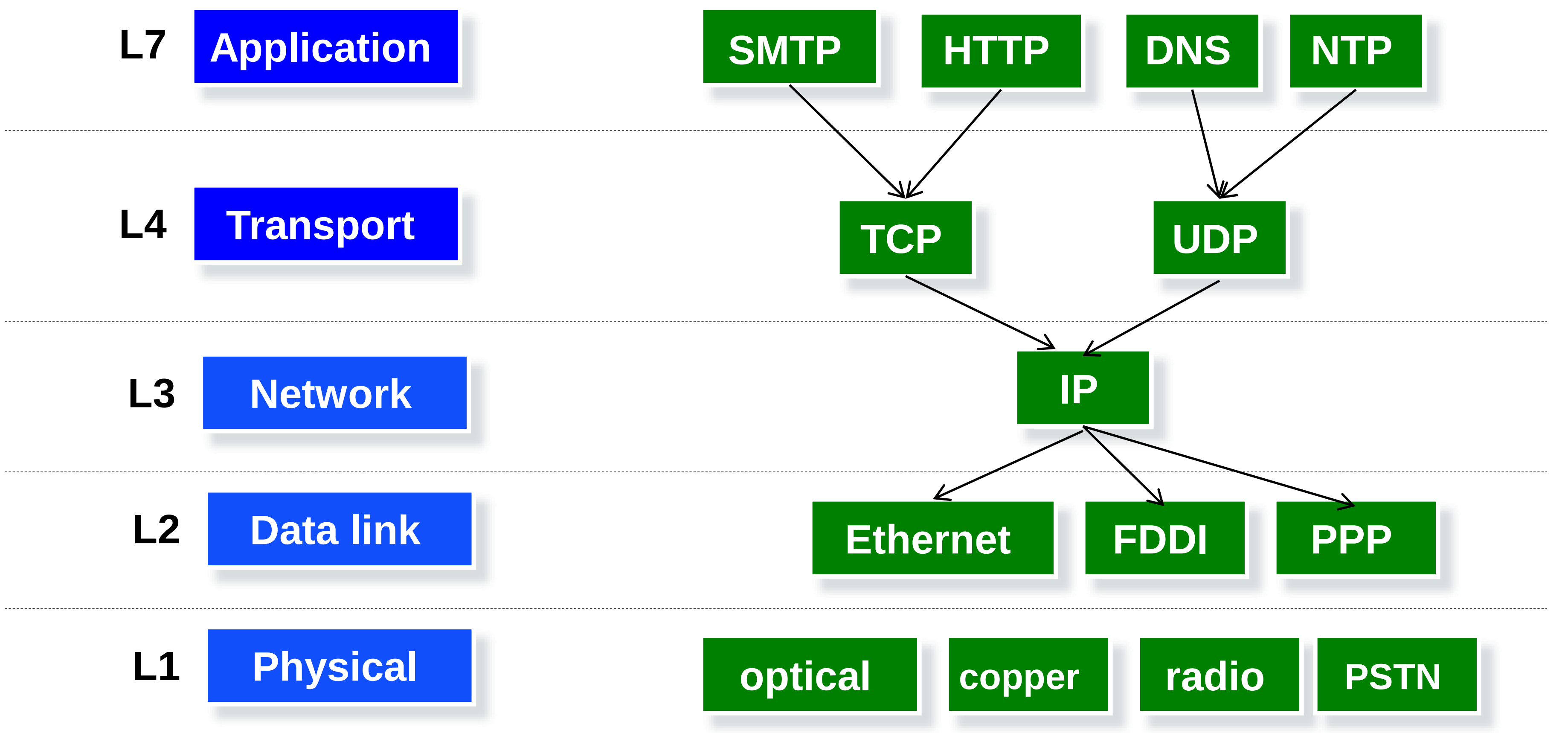
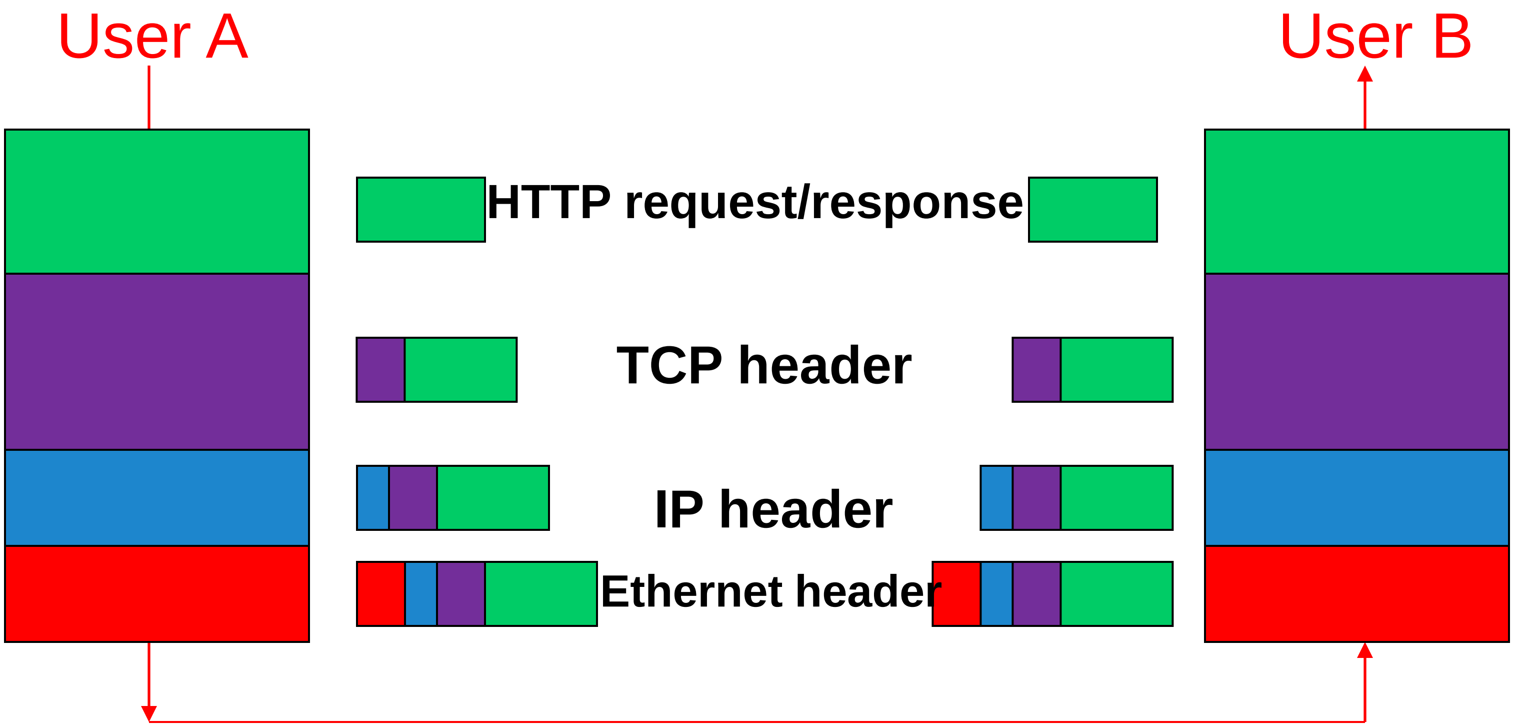


Image from https://madpackets.com/2018/04/25/tcp-sequence-and-acknowledgement-numbers-explained/

Protocols at different layers



Layer Encapsulation: Protocol Headers



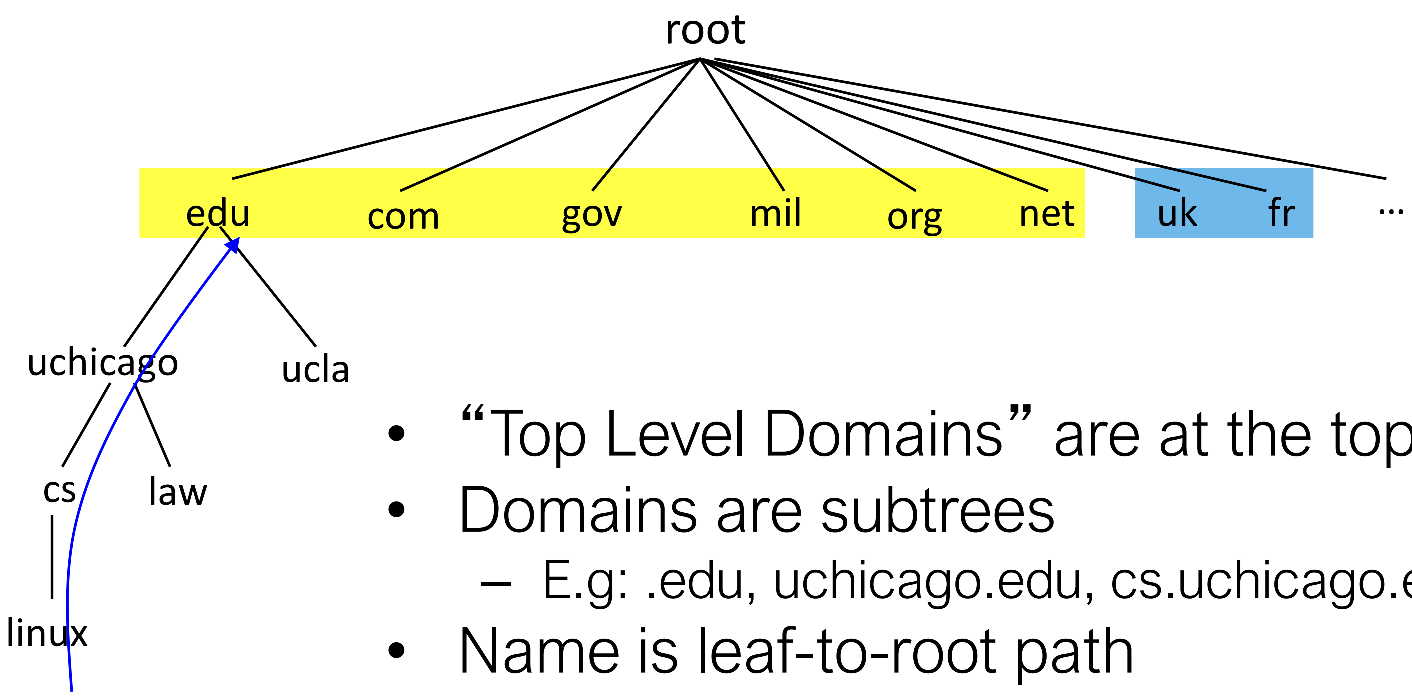
Goal: Be addressable in ways humans can remember on the Internet

Solution: Domain Names

DNS (Domain Name System)

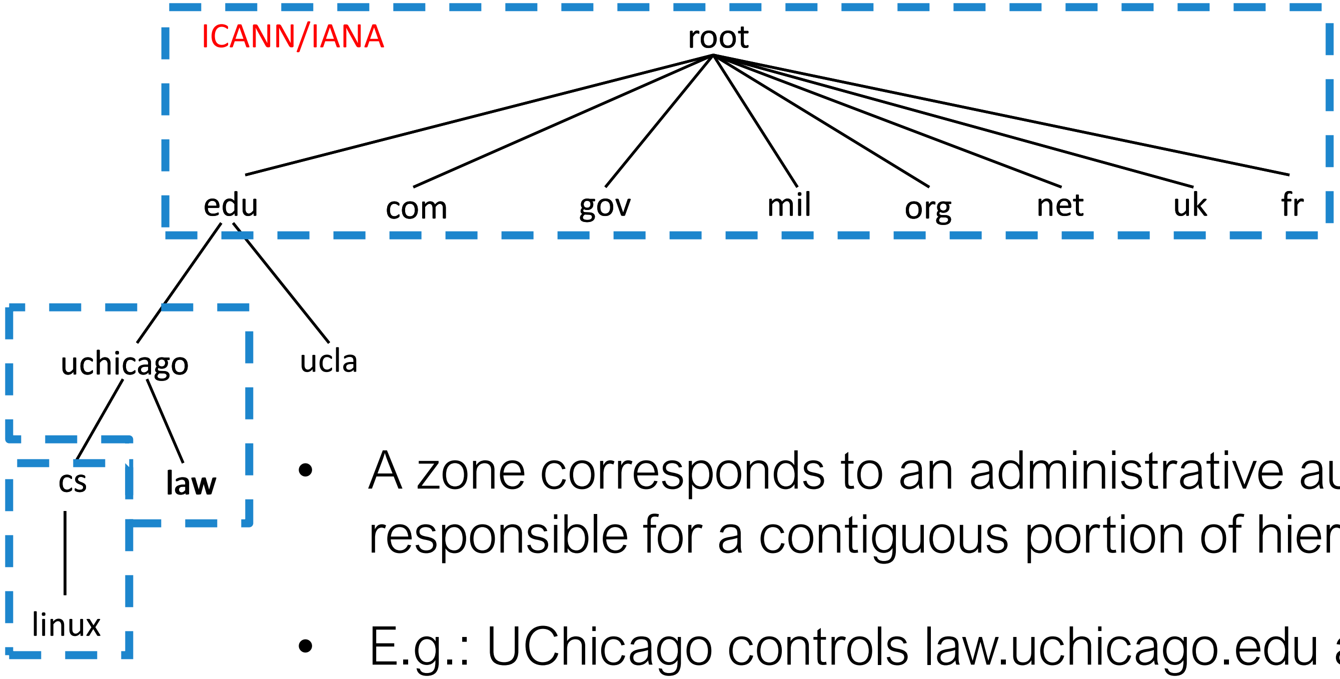
- Host addresses: e.g., *128.135.11.239*
  - a number used by protocols
  - conforms to network structure (the "where")
- Host names: e.g., *super.cs.uchicago.edu*
  - usable by humans
  - conforms to organizational structure (the "who")
- Domain Name System (DNS) is how we map from one to other
  - a directory service for hosts on the Internet
  - See *nslookup*

Hierarchical Namespace



- "Top Level Domains" are at the top
- Domains are subtrees
  - E.g: .edu, uchicago.edu, cs.uchicago.edu
- Name is leaf-to-root path
  - linux.cs.uchicago.edu
- Name collisions trivially avoided!
  - each domain's responsibility

Hierarchical Administration



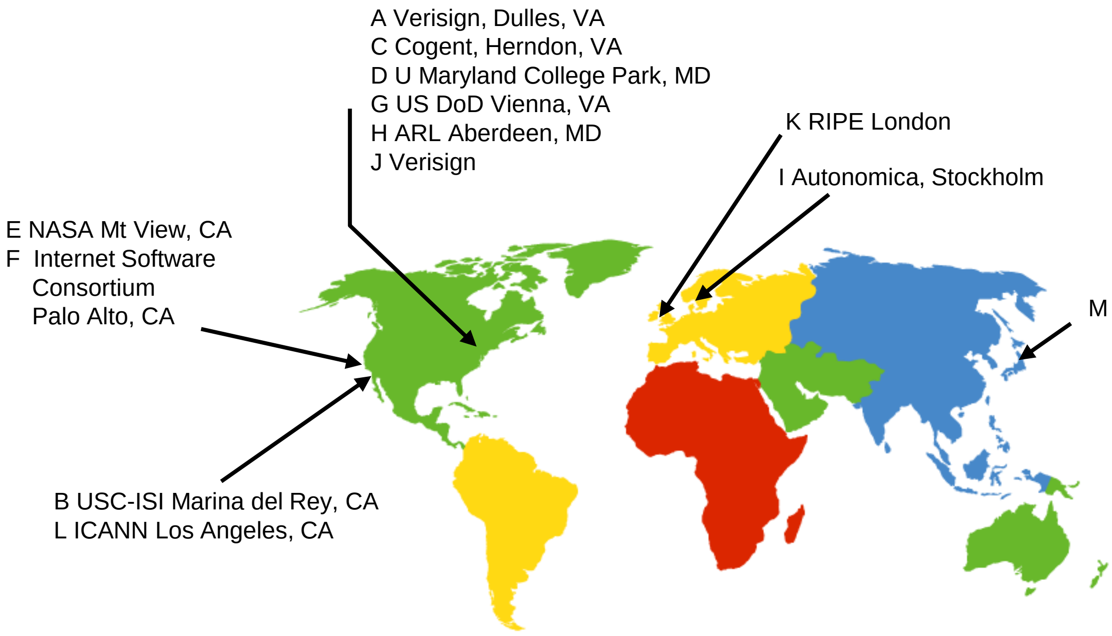
- A zone corresponds to an administrative authority responsible for a contiguous portion of hierarchy
- E.g.: UChicago controls law.uchicago.edu and *.cs.uchicago.edu
  while CS controls *.cs.uchicago.edu

Political Environment For Domains

- Internet Corporation for Assigned Names and Numbers (ICANN) is a non-profit that controls the assignment of both IP addresses and domain names

DNS Root Servers

- 13 root servers (labeled A-M; see http://www.root-servers.org/)



DNS Root Servers

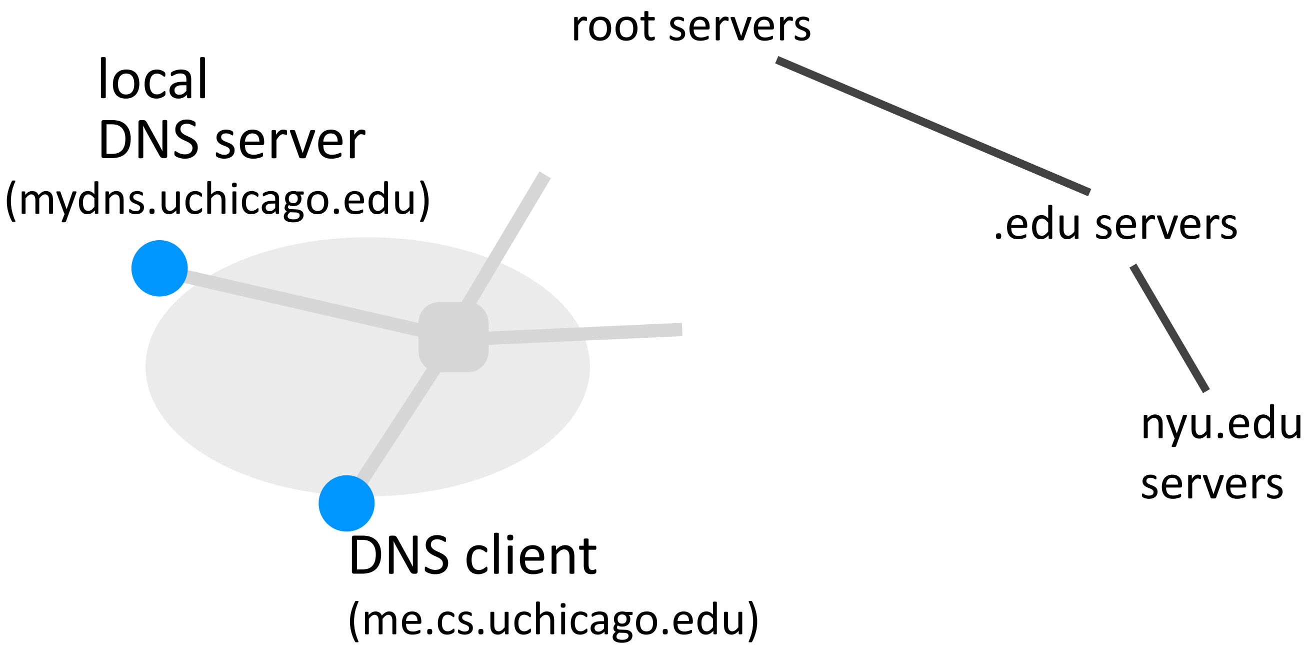
- 13 root servers (labeled A-M; see http://www.root-servers.org/)
- All replicated via anycast

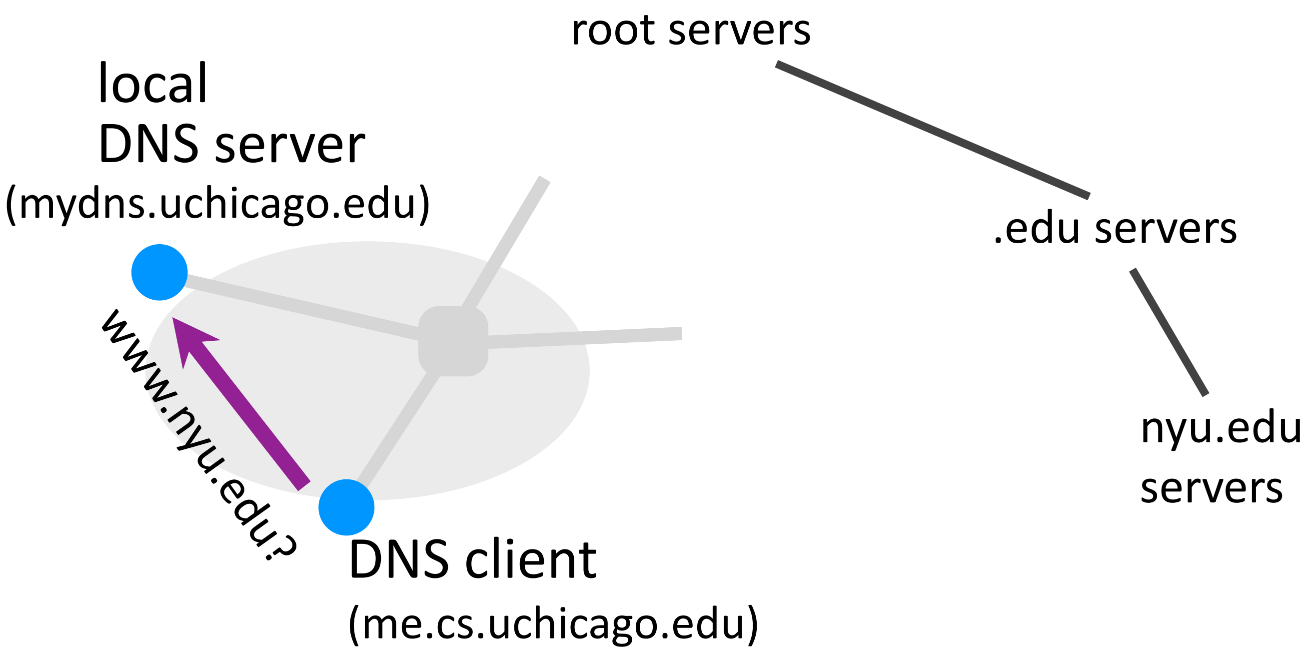
DNS Records

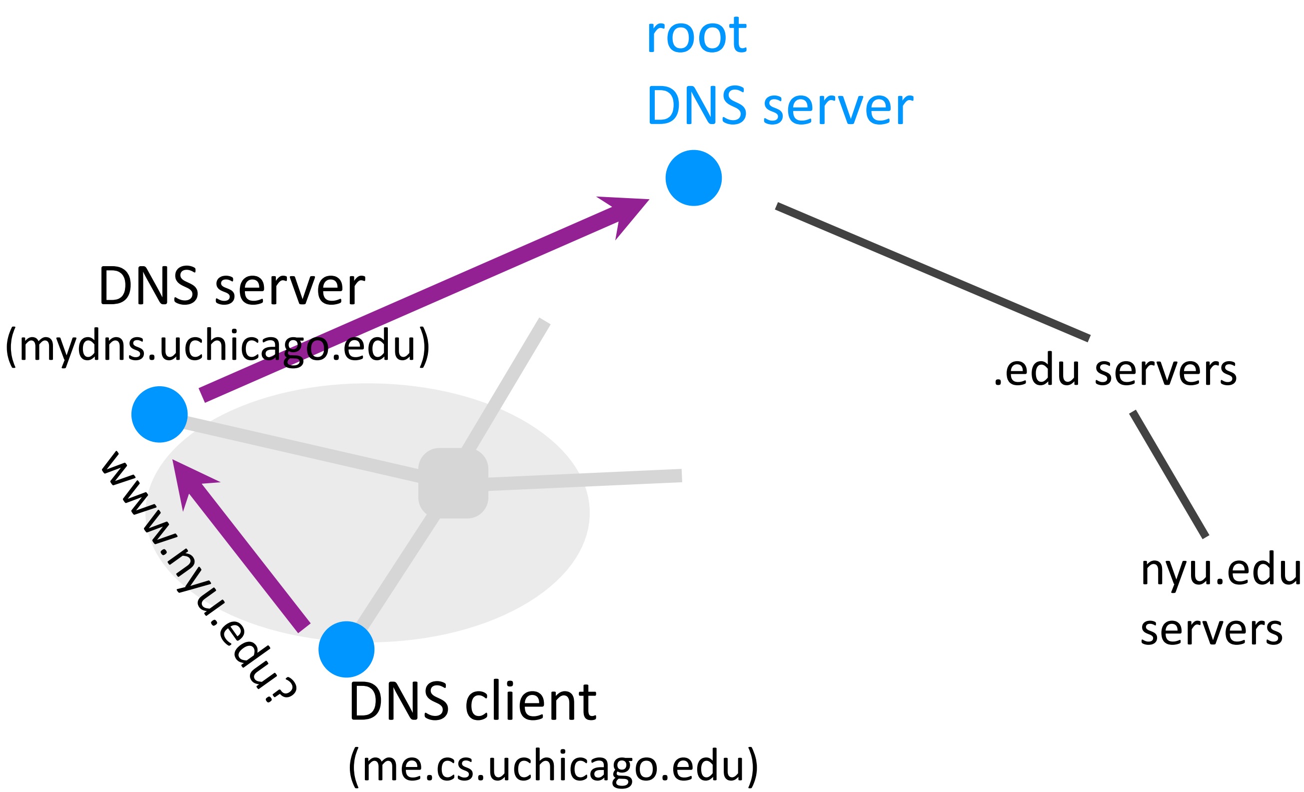
- DNS servers store Resource Records (RRs)
  - RR is (name, value, type, TTL)
- Type = A: *(→ Address)*
  - name = hostname
  - value = IP address
- Type = NS: *(→ Name Server)*
  - name = domain
  - value = name of dns server for domain
- Type = MX: *(→ Mail eXchanger)*
  - name = domain in email address
  - value = name(s) of mail server(s)

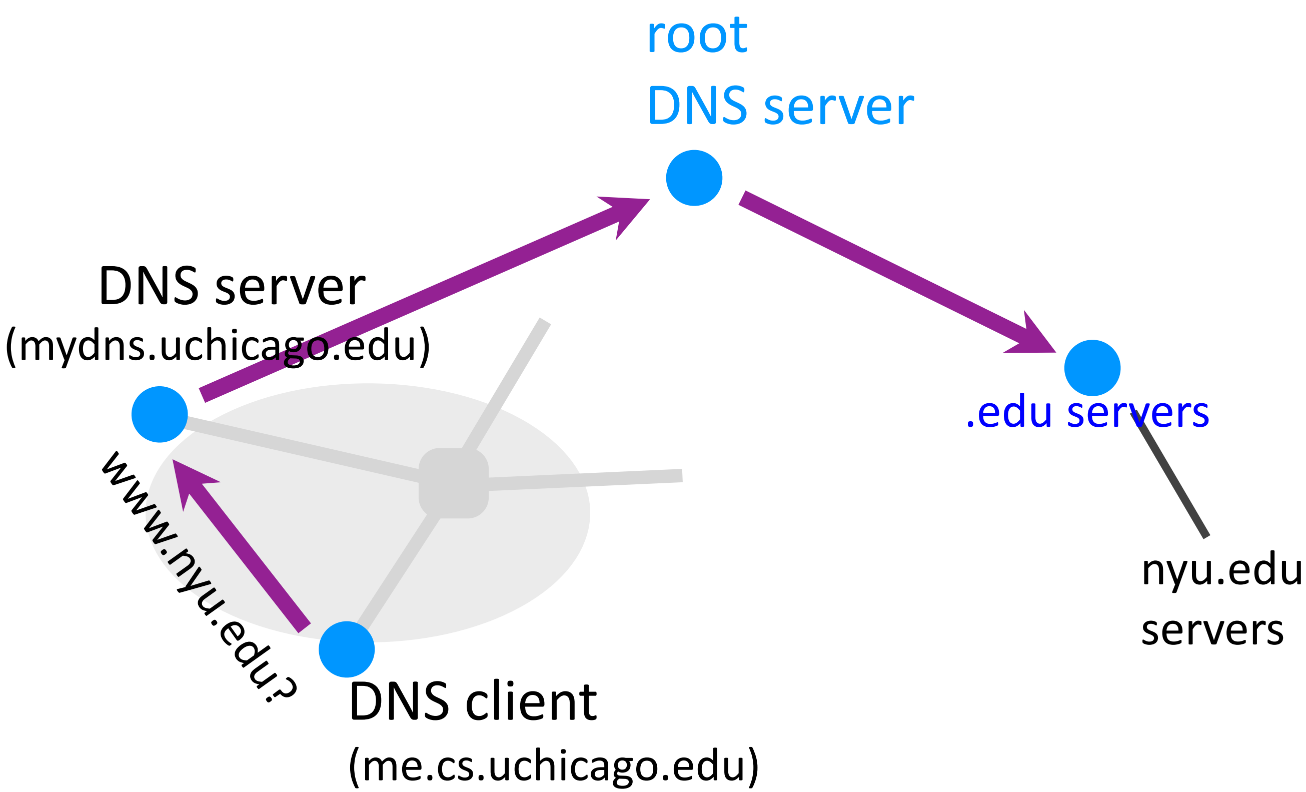
Inserting Resource Records into DNS

- Example: you want “blaseur.com”
- Register blaseur.com at registrar (e.g., GoDaddy)
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts into the .com TLD server who your name servers are
- Store resource records in your server
  - e.g., type A record for www.blaseur.com
  - e.g., type MX record for blaseur.com

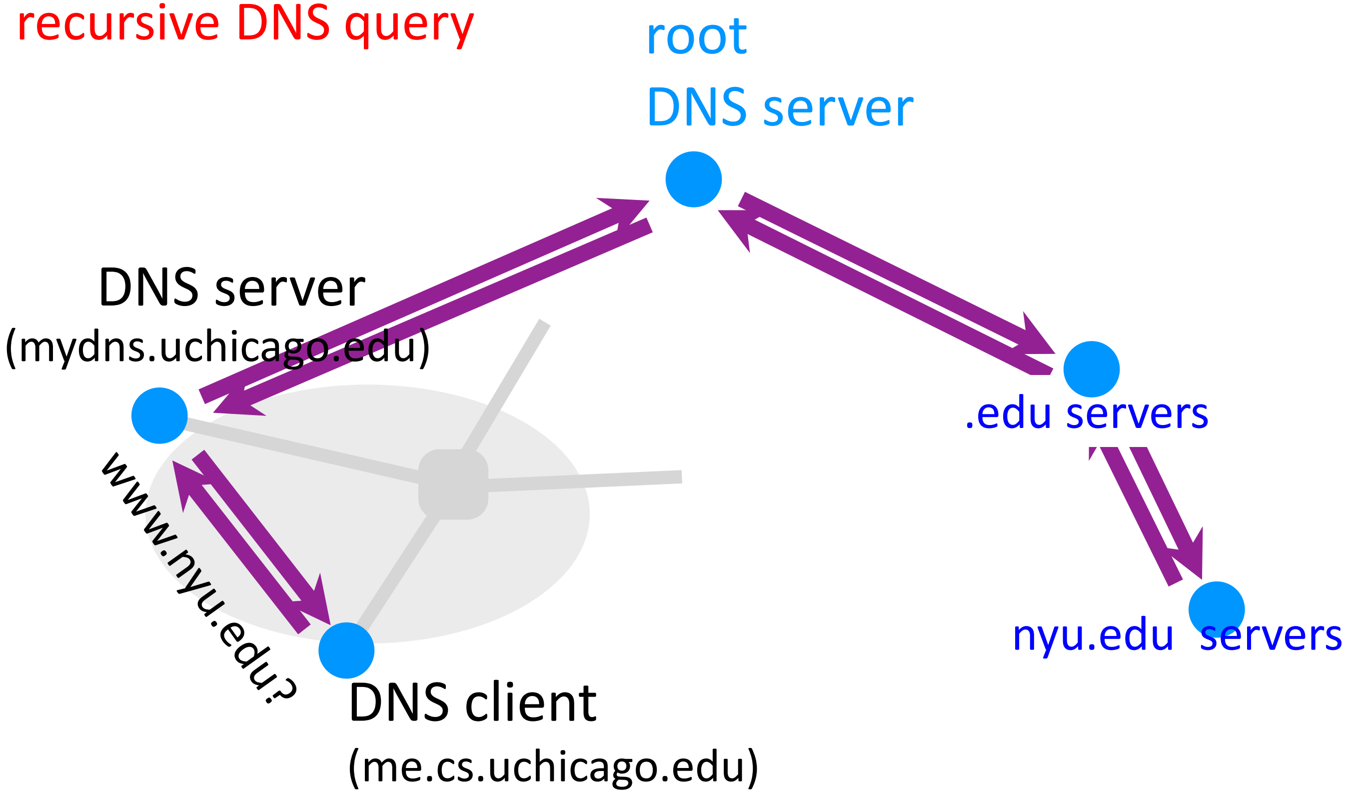

root servers
local
DNS server
(mydns.uchicago.edu)
.edu servers
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)

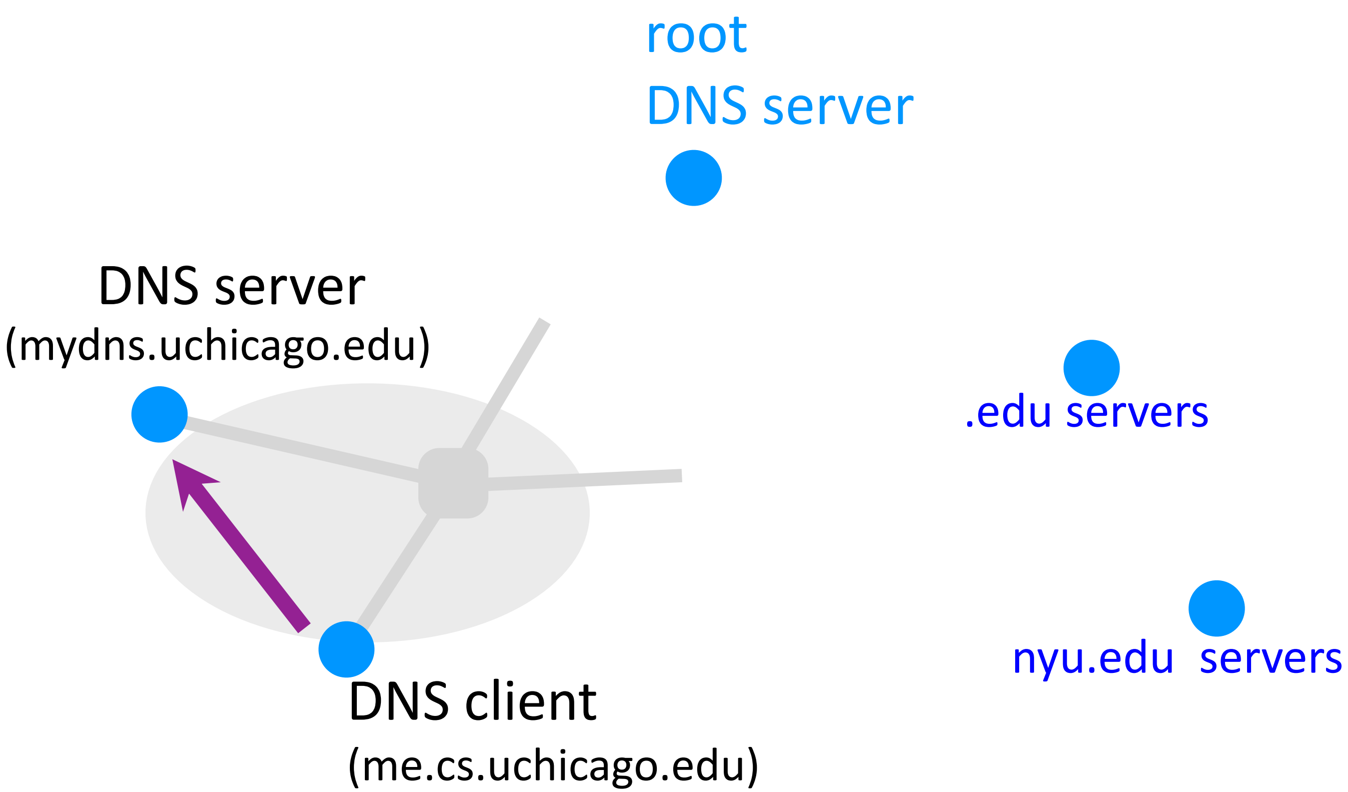

root servers
local
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)


root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)

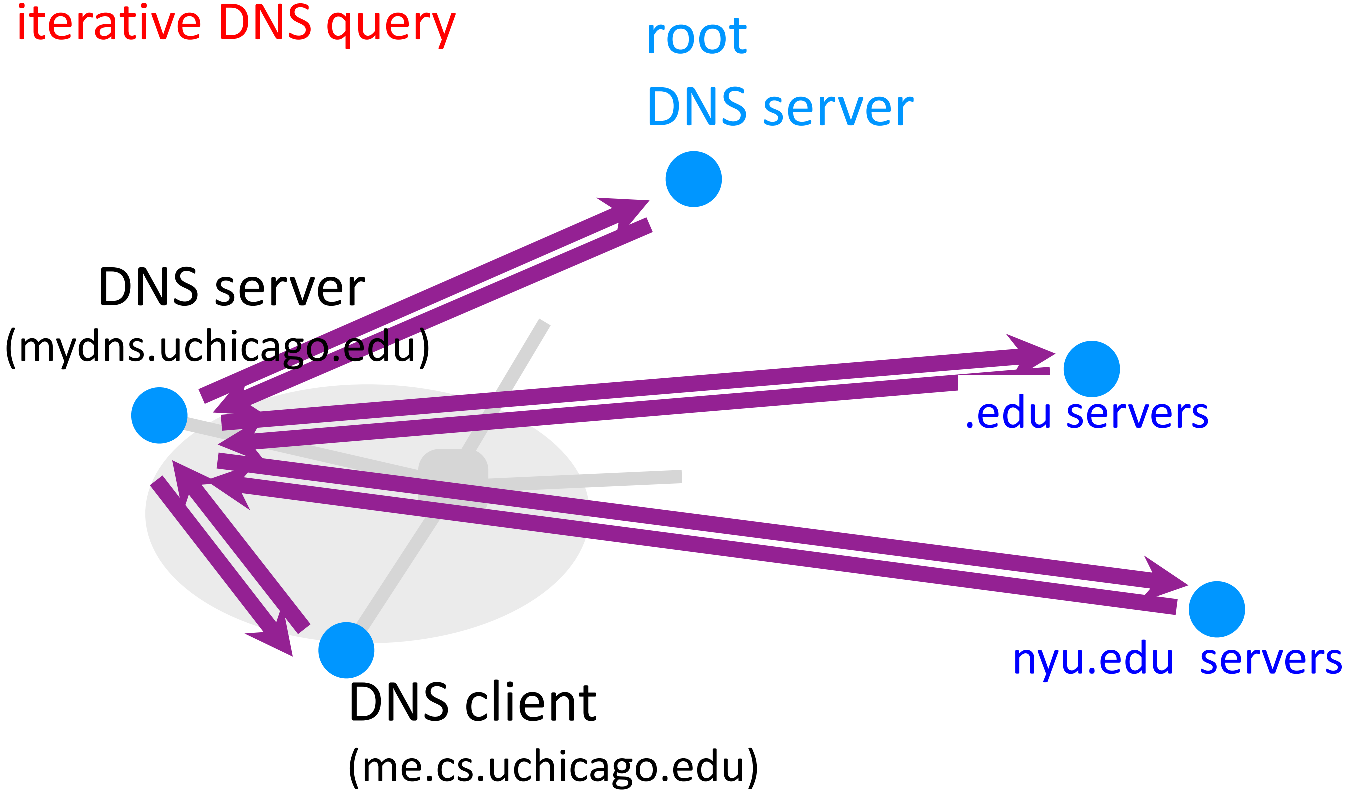

root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)

recursive DNS query




root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
nyu.edu servers
DNS client
(me.cs.uchicago.edu)

iterative DNS query



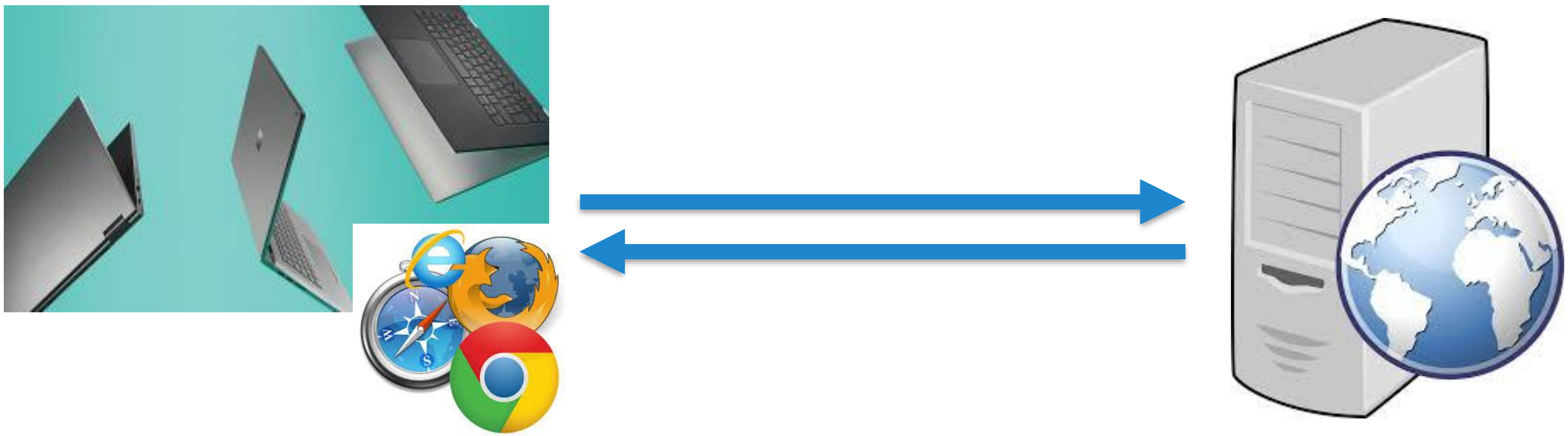
DNS FAQs

- Do you have to follow that recursive process every time?
  - No (DNS queries are cached)
- Is DNS “secure” / “private”?
  - No
- Have people tried to make DNS secure
  - Yes. See, e.g., DNSSEC, which aims to provide integrity by signing DNS records. These efforts are ongoing!

Now, let’s see how the web works!

Your interface to the web

- Your web browser contacts a web server



A 10,000 Foot View of Technologies

- Where things run:

HTTP(S)

HTML / CSS

JavaScript (Angular/React)

Browser Extensions

Django (Python) / CGI (Perl) / PHP / Node.js / Ruby on Rails

Databases (MySQL)

The Anatomy of a URL

- https://www.uchicago.edu/fun/funthings.html?query=music&year=2022#topsection

The Anatomy of a URL

- https://www.uchicago.edu/fun/funthings.html?query=music&year=2022#topsection
  - Protocol: https
  - Hostname: www.uchicago.edu
    - .edu is the top level domain (TLD)
  - Path: /fun/funthings.html
  - Parameters: (key=value pairs, & delimited)
  - Named anchor: #topsection
- Some technologies (e.g., Django) parse the path differently (e.g., parameters are there)

The Anatomy of a Webpage

- view-source:https://www.cs.uchicago.edu/
- HTML (hypertext markup language)
  - Formatting of a page
  - All sorts of formatting: <div><p>Hi</p></div>
  - Links: Click here
  - Pictures:
  - Forms
- HTML 5 introduced many media elements

The Anatomy of a Webpage