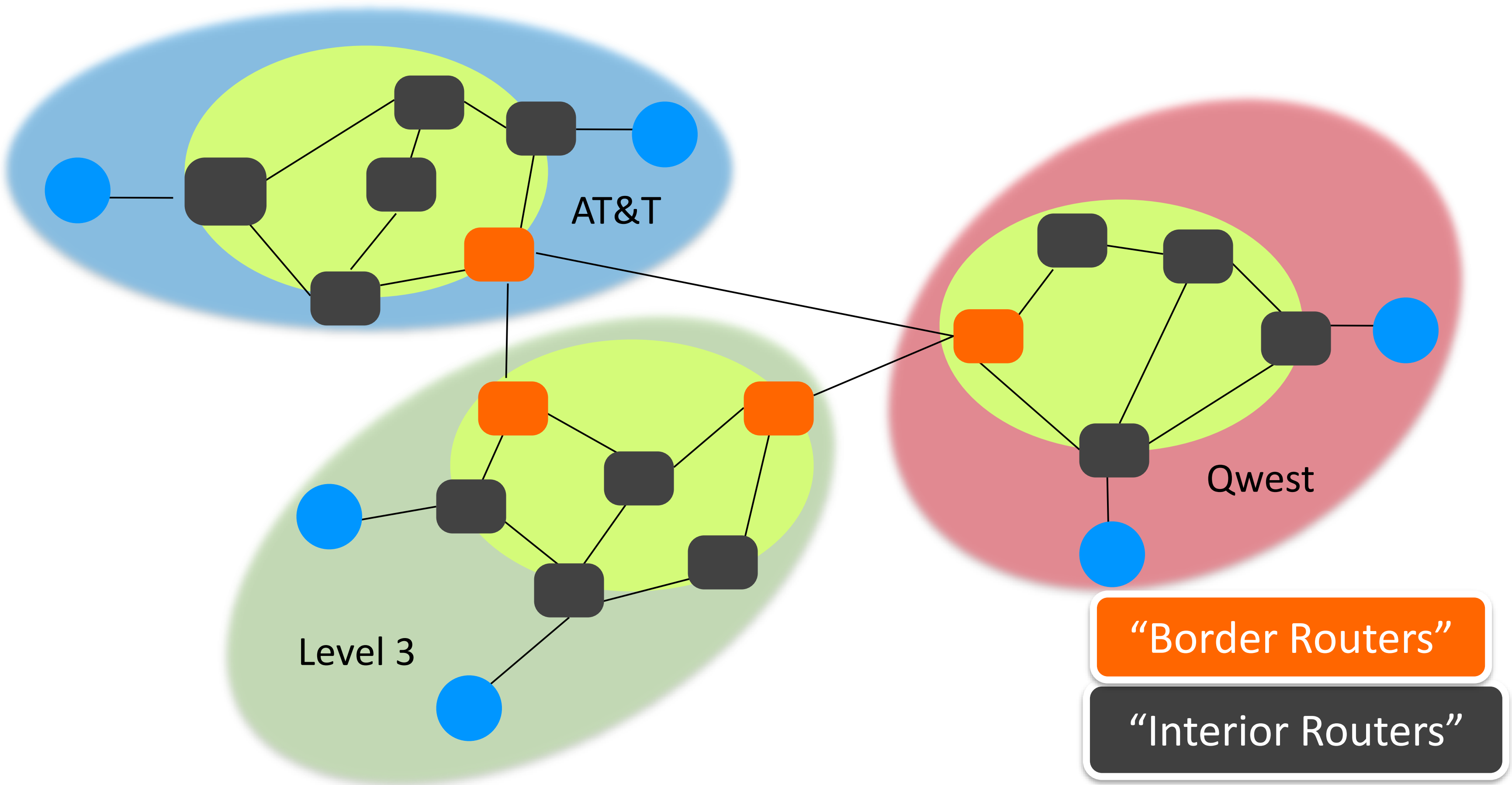
09. How the Internet Works

Blase Ur and David Cash
(Some slides borrowed from Ben Zhao)
February 1st, 2021
CMSC 23200 / 33250

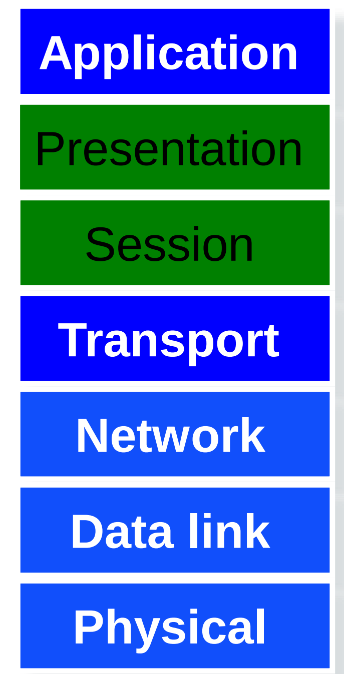
THE UNIVERSITY OF CHICAGO

The Internet From 10,000 Feet



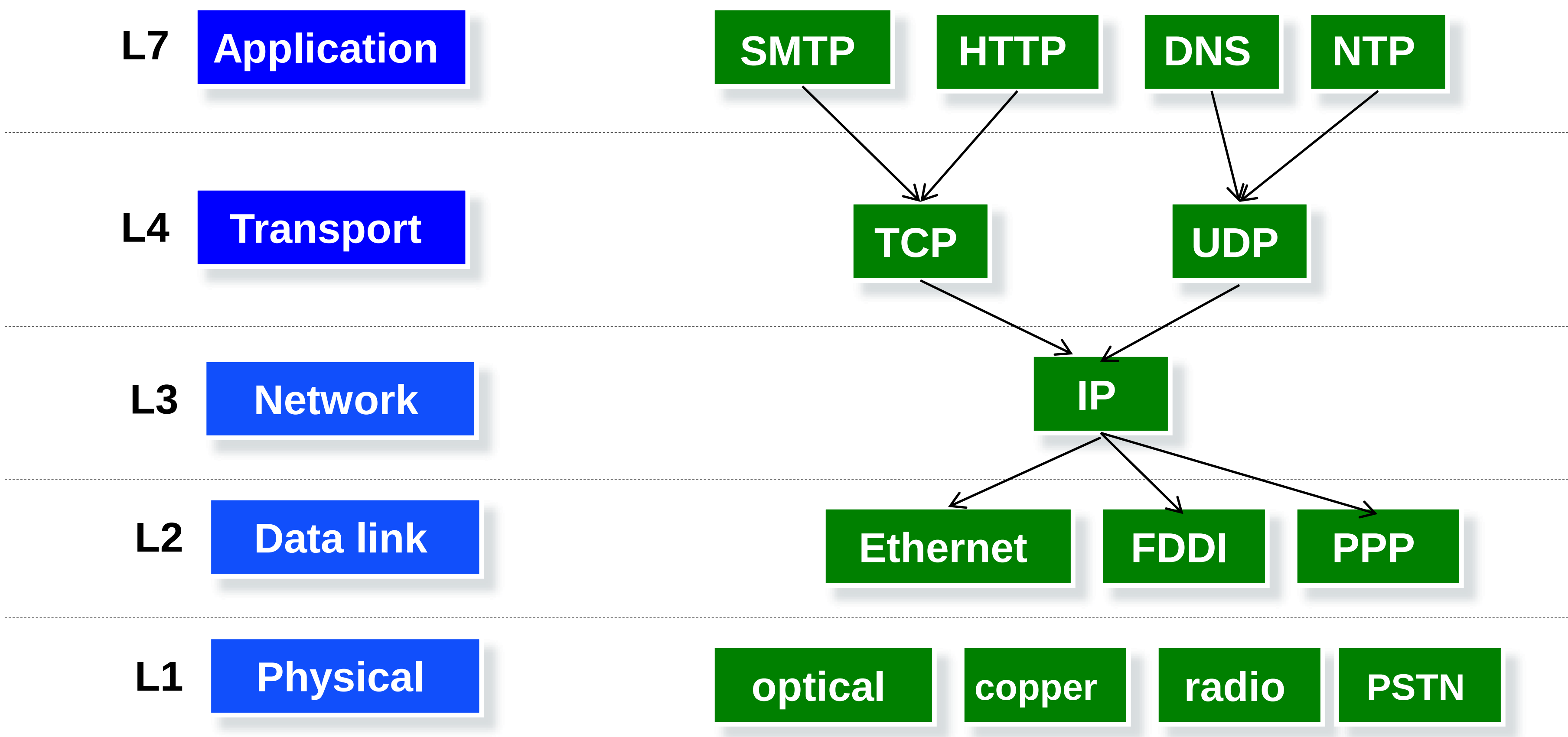
Layers (OSI Model)

- Layer = a part of a system with well-defined interfaces to other parts
- A layer interacts only with layer above and layer below



Networking's own version of modularity

Protocols at different layers



Goal: Be addressable on a local network

Solution: MAC Addresses (Link Layer)

MAC (Media Access Control) Address

- Unique-*ish* 48-bit number associated with network interface controller (NIC)

12:34:56:78:9A:BC

- Usually assigned by manufacturers
  - In theory, doesn't ever change for a piece of hardware
  - In practice, MAC addresses can be spoofed
- See *ifconfig* and similar commands

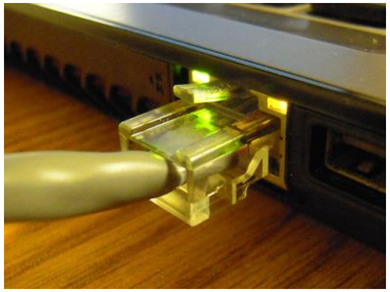
MAC (Media Access Control) Address

- Broadcast address received by everyone (as opposed to unicast/multicast)

FF:FF:FF:FF:FF:FF

- NICs filter traffic by MAC Address
  - Exception: promiscuous/monitor modes (relevant to Assignment 3)
- On the link layer, data is split into packets/frames (often 1500 bytes)

MAC Addresses Used on Link Layer

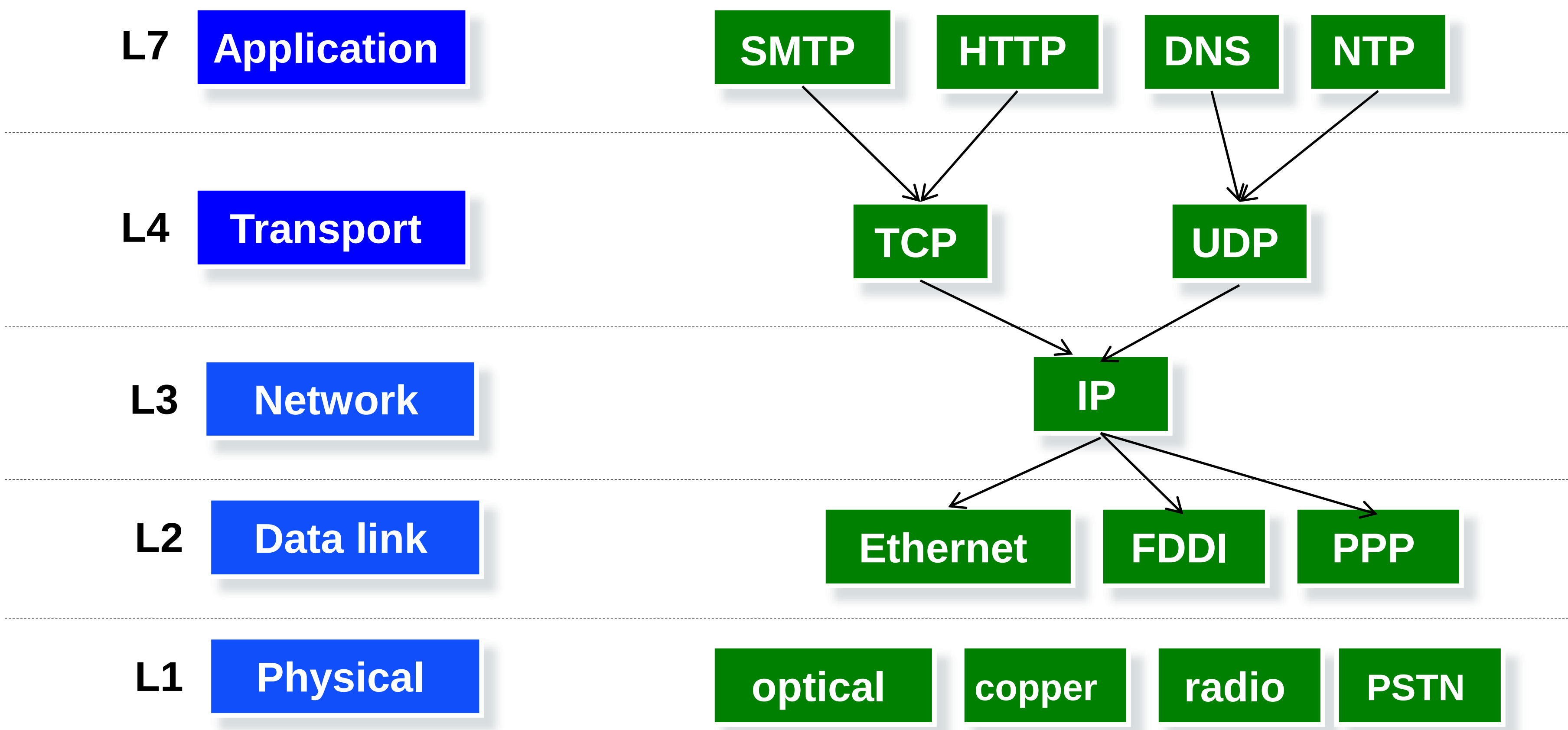


- Ethernet (plugged in)
  - Some hardware (e.g., hubs) repeats all traffic
  - Some hardware (e.g., switches) filters by MAC address
- Wi-Fi (802.11)
  - Your Wi-Fi card typically filters only unicast traffic for you and broadcast traffic
  - Exception: promiscuous/monitor modes (relevant to Assignment 4)

Wi-Fi Encryption

- WEP (Wired Equivalent Privacy)
  - Broken; hard to configure
  - Abandoned in 2004
- WPA (Wi-Fi Protected Access)
  - Vulnerable, particularly the WPS feature
- WPA2
  - Uses AES
- WPA3 recently introduced
  - Device-specific encryption on public networks

Protocols at different layers



Goal: Be addressable on the Internet

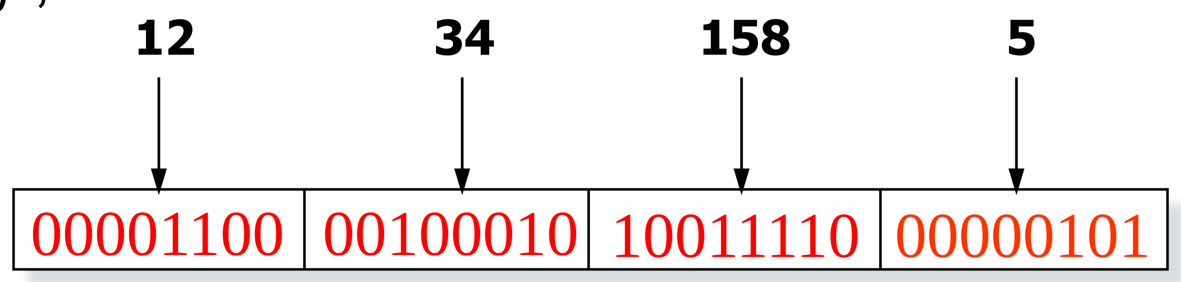
Solution: IP Addresses (Network Layer)

IP Addresses (IPv4)

- Unique-*ish* 32-bit number associated with host

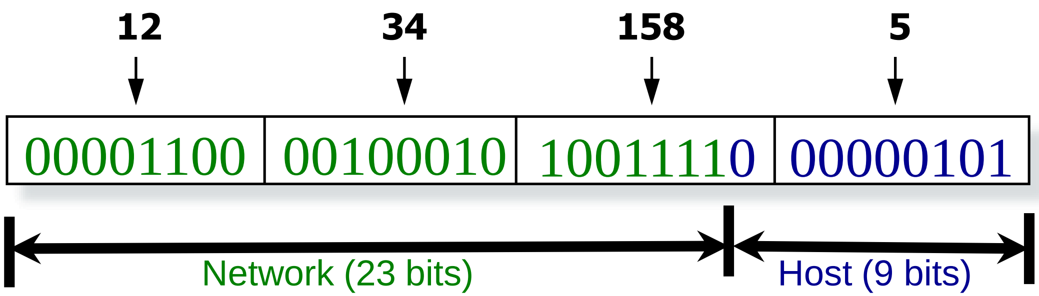
00001100 00100010 10011110 00000101

- Represented with "dotted quad" notation
  - e.g., 12.34.158.5



Hierarchy in IP Addressing

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the network component; suffix is host component



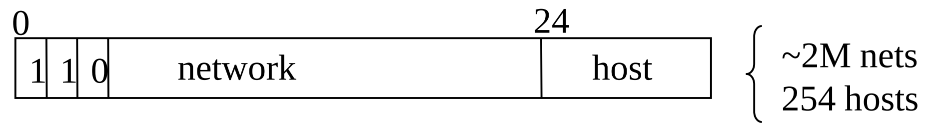
- Interdomain routing operates on the network prefix

Early Design: “Classful” Addressing

- Three main classes
  - Class A

  - Class B

  - Class C



Problem: Networks only come in three sizes!

Today's Addressing

- CIDR = Classless Interdomain Routing
- Idea: Flexible division between network and host addresses
  - Offer better tradeoff between size of routing table and use of IP address space

CIDR (example)

- Suppose a network has 50 computers
  - allocate 6 bits for host addresses (since $2^5 < 50 < 2^6$)
  - remaining 32 - 6 = 26 bits as network prefix
- Flexible boundary means the boundary must be explicitly specified with the network address!
  - informally, “slash 26” → 128.23.9/26
  - formally, prefix represented with a 32-bit mask: 255.255.255.192
    where all network prefix bits set to “1” and host suffix bits to “0”

Allocation Done Hierarchically

- Internet Corporation for Assigned Names & Numbers (ICANN) gives large blocks to…
  - Regional Internet Registries, such as American Registry for Internet Names (ARIN), which give blocks to…
- Large institutions (ISPs), which give addresses to…
- Individuals and smaller institutions

e.g. ICANN ➔ ARIN ➔ Qwest ➔ UChicago ➔ CS

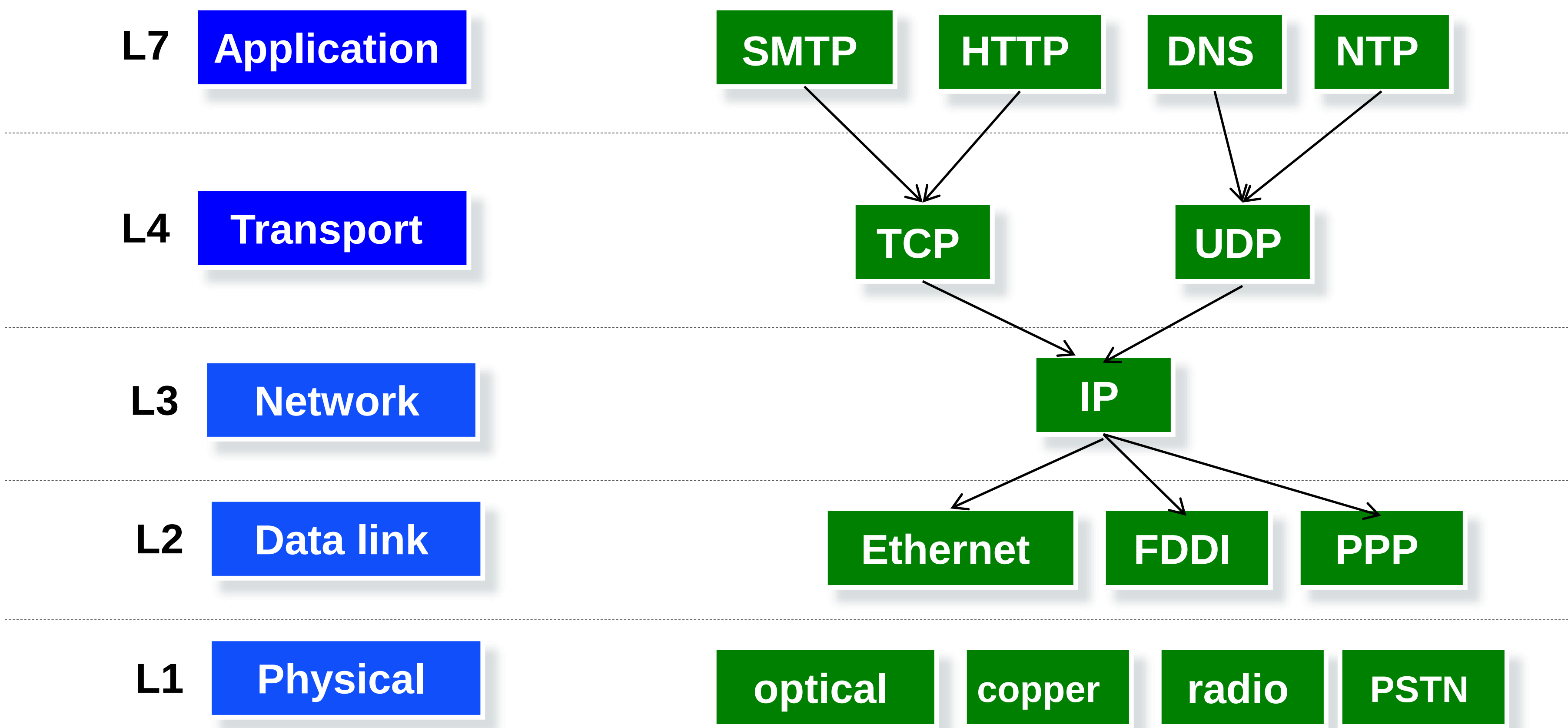
Example in More Detail

- ICANN gives ARIN several /8s
- ARIN gives Qwest one /8, 128.0/8
  - Network Prefix: 10000000
- Qwest gives UChicago a /16, 128.135/16
  - Network Prefix: 1000000010000111
- UChicago gives CS a /24, 128.135.11/24
  - Network Prefix: 100000001000011100001011
- CS gives me a specific address 128.135.11.176
  - Address: 10000000100001110000101110110000

IP Address FAQs

- How do you get an IP Address?
  - Typically use Dynamic Host Configuration Protocol (DHCP) upon connection to networks
- Does your IP address change over time?
  - Yes, frequently when you switch networks or reconnect
- Why is my router usually 192.168.1.1?
  - Private IP Addresses: 192.168.*.* and 10.*.*.* and 172.16.*.* through 172.31.*.*
- Can you share an IP address?
  - Yes! Especially behind routers / NATs / middleboxes

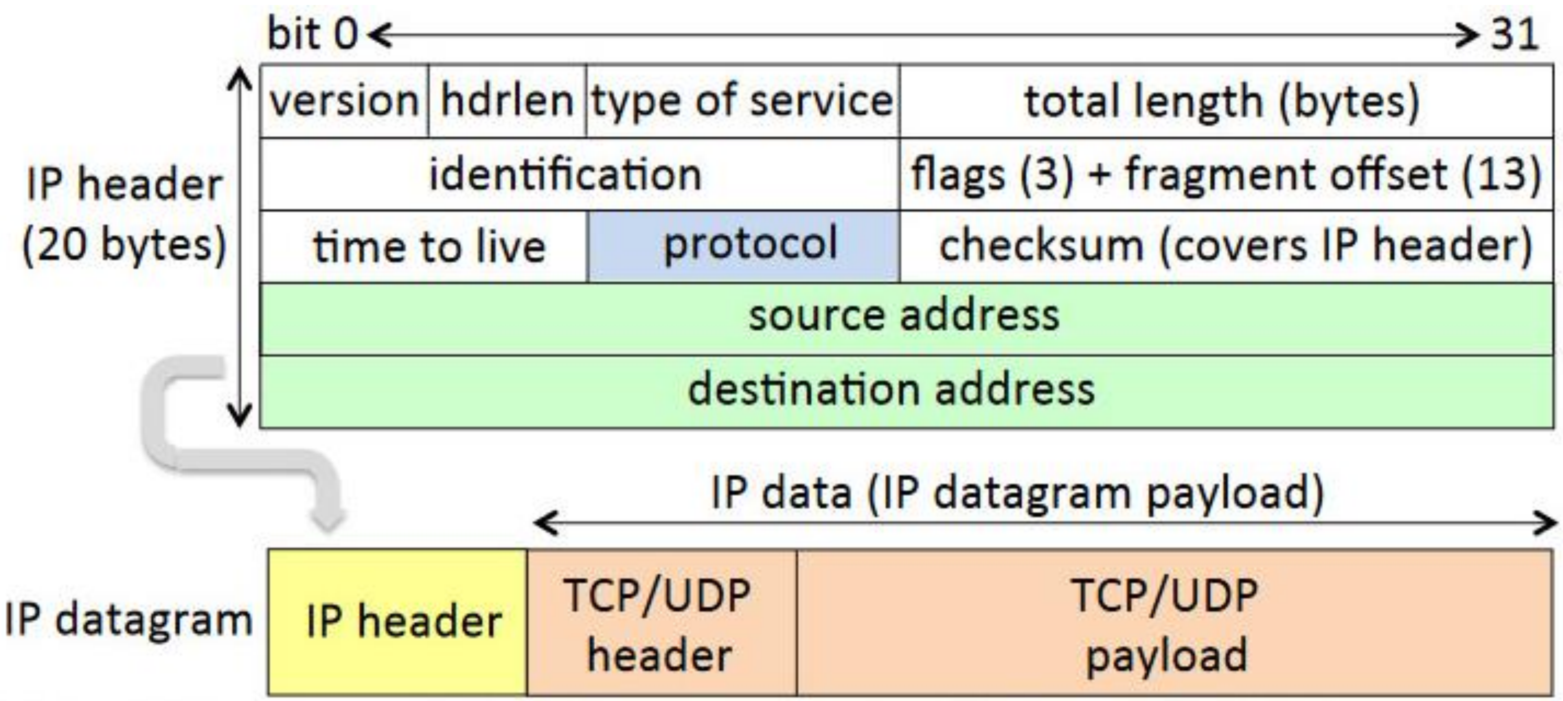
Protocols at different layers



Goal: Get data to its destination

Solution (Protocol): IP at the network layer

IP (Internet Protocol)



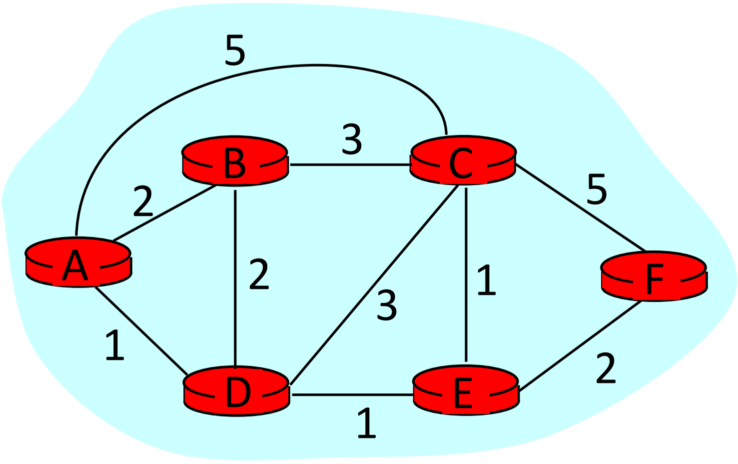
Goal: Get data to its destination

Solution (Part 2): Routing

Routing

- Goal: determine "good" path through network from source to destination

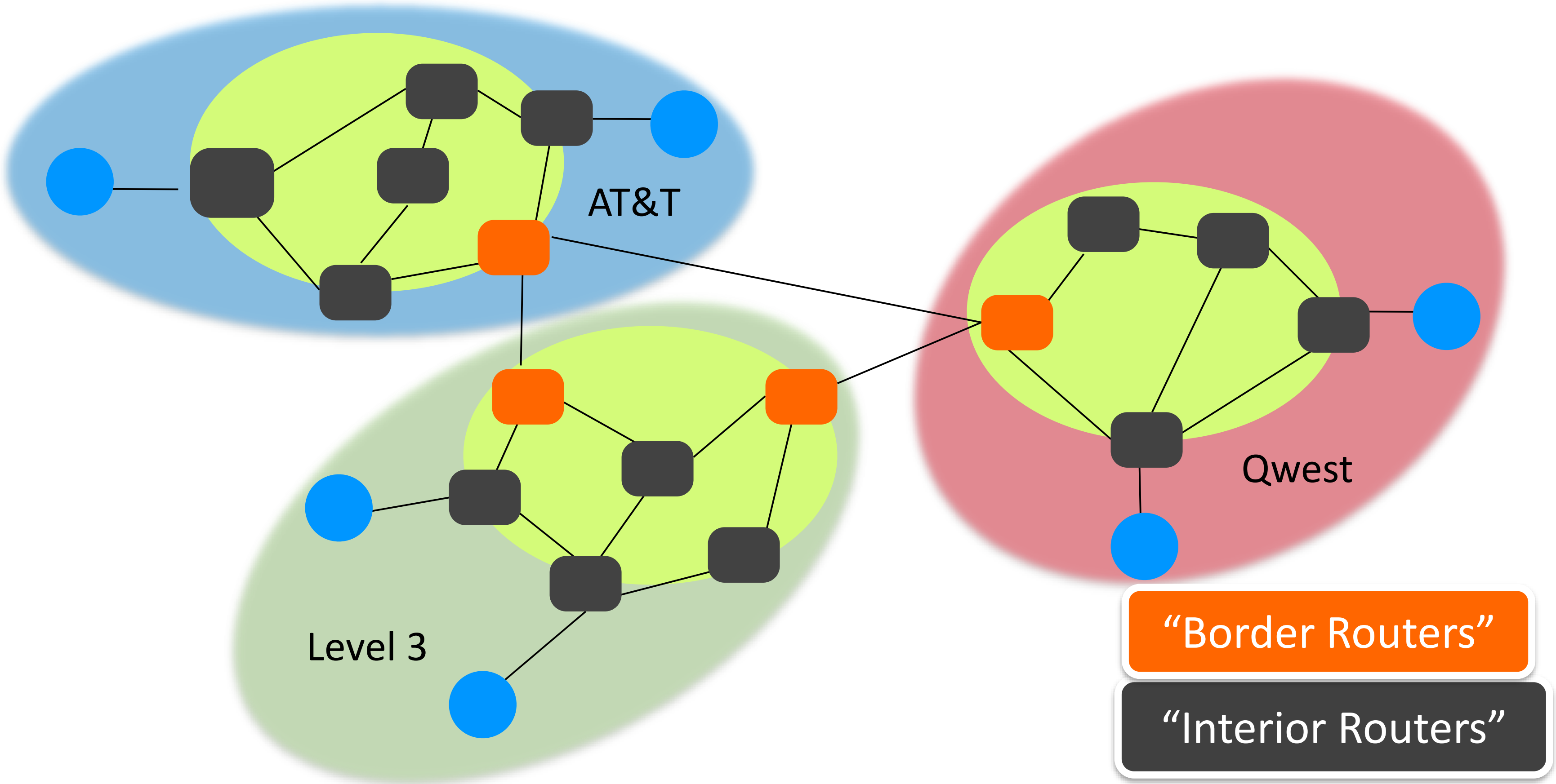
- Network modeled as a graph
  - Routers → nodes, Link →edges
    - Edge cost: delay, congestion level,.
  - A node knows only its neighbors and the cost to reach them



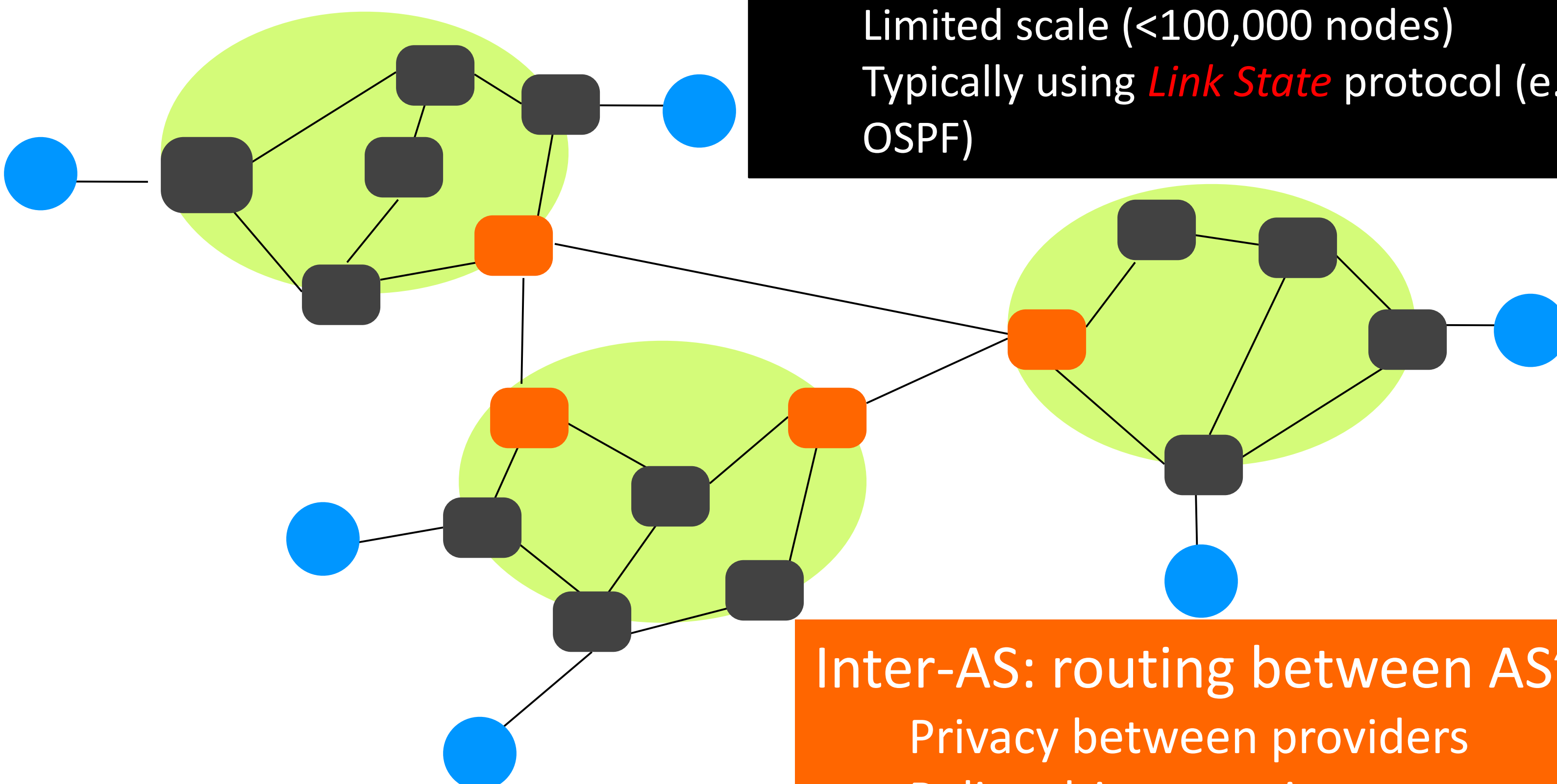
- How does each node learns how to reach every other node along the shortest path?

Autonomous System (AS)

- Collection of IP prefixes under the control of a single administrative entity
- 92,000+ ASes as of August 2019



Intra-AS & Inter-AS Routing



Intra-AS: routing within a single AS

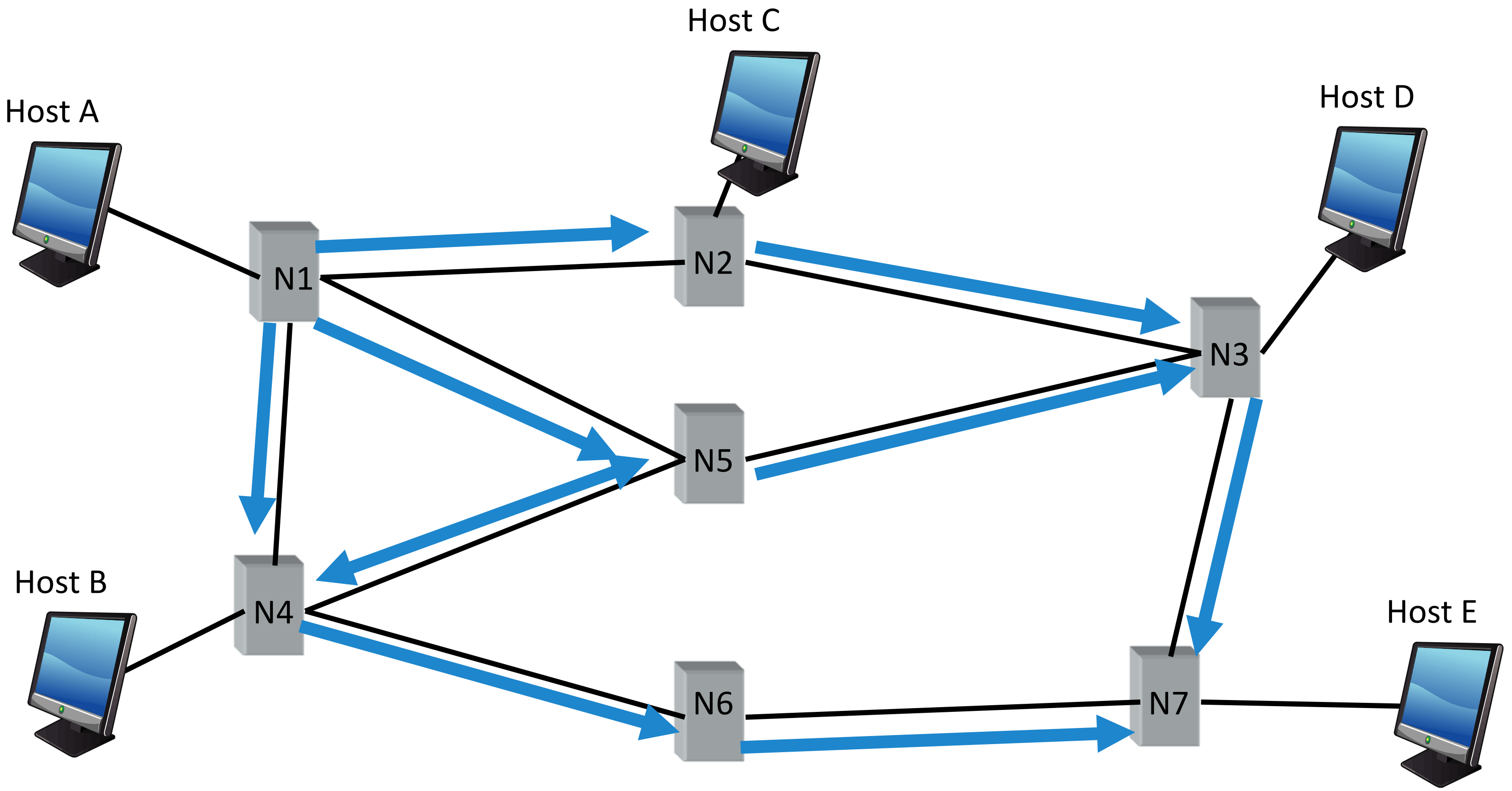
- Trusted domain (within one company)
- Limited scale (<100,000 nodes)
- Typically using *Link State* protocol (e.g. OSPF)

Inter-AS: routing between AS's

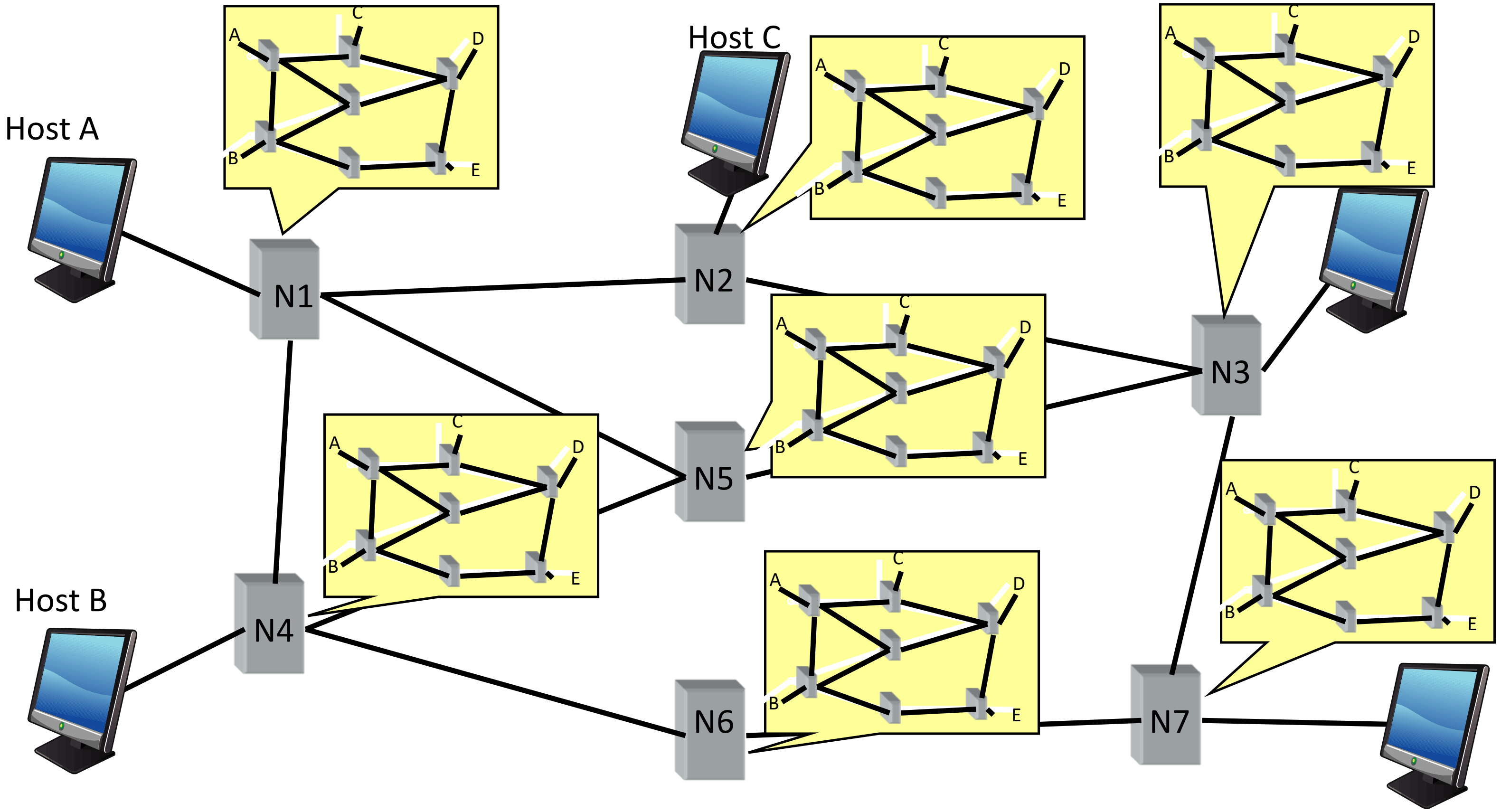
- Privacy between providers
- Policy-driven routing
- BGP, a *Path Vector* protocol
  - Variant of *Distance Vector* routing

Link State: Control Traffic

- Each node floods its local information to every other node in network
- Each node ends up knowing entire network topology
  → use Dijkstra to compute shortest path to every other node

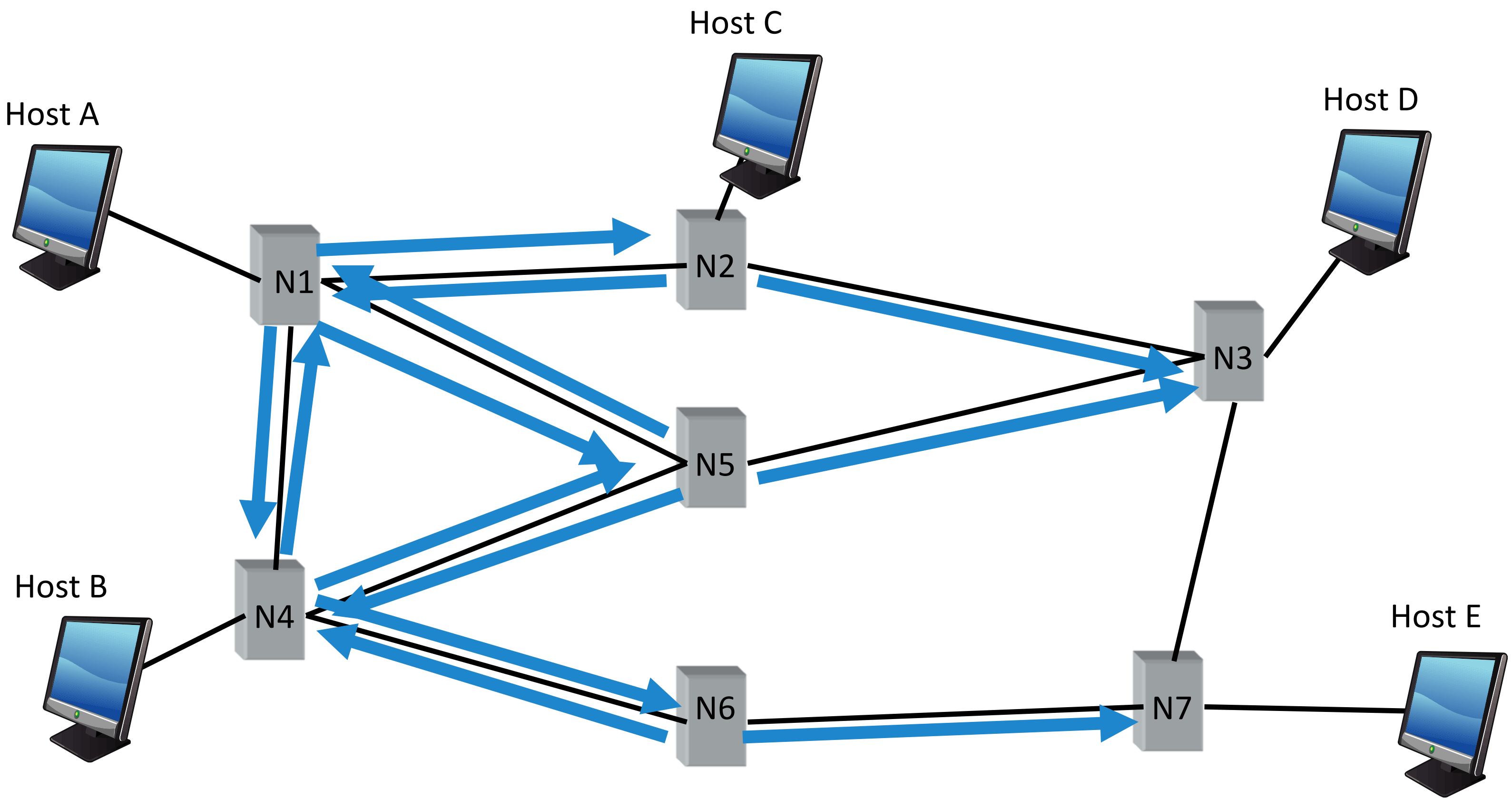


Link State: Node State

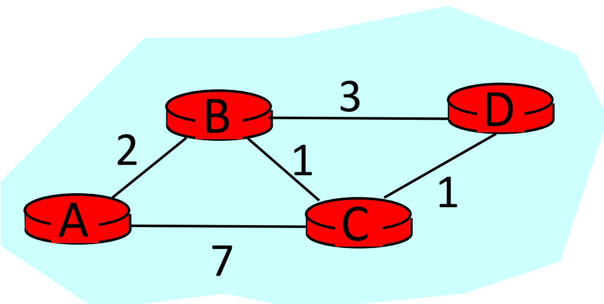


Distance Vector: Control Traffic

- When the routing table of a node changes, it sends table to neighbors
  - A node updates its table with information received from neighbors



Example: Distance Vector Algorithm



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

Node C

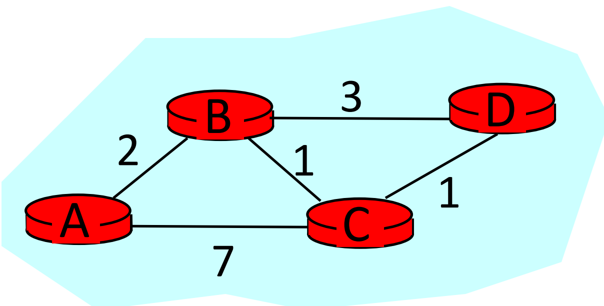
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
Initialization:
  for all neighbors V do
    if V adjacent to A
      D(A, V) = c(A,V);
  else
      D(A, V) = ∞;
...
```

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

(D(C,A), D(C,B), D(C,D))

Node C

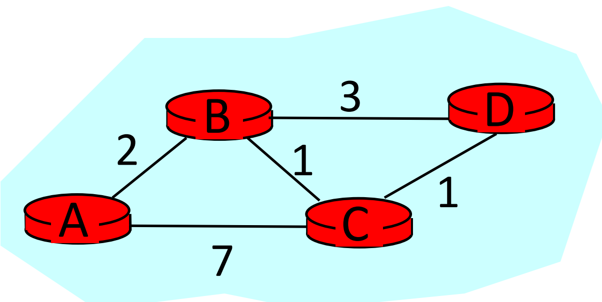
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	**8**	**C**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D)= min(D(A, D),D(A,C)+D(C,D)
= min(∞ , 7 + 1) = 8

(D(C,A), D(C,B), D(C,D))

Node C

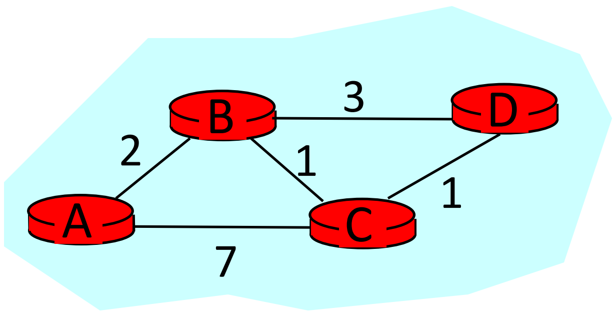
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	8	C

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

Node C

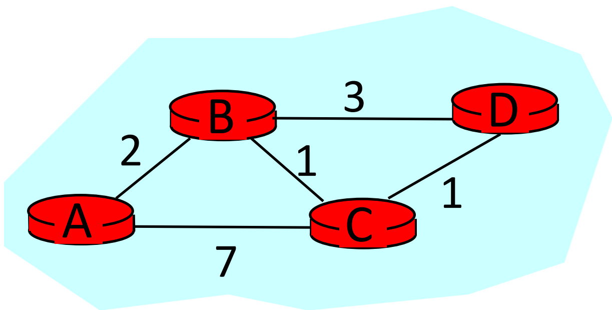
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Example: 1st Iteration (B→A, C→A)



Node A

Dest.	Cost	NextHop
B	2	B
C	**3**	**B**
D	**5**	**B**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D) = min(D(A,D), D(A,B) + D(B,D))
= min(8, 2 + 3) = 5

D(A,C) = min(D(A,C), D(A,B) + D(B,C))
= min(7, 2 + 1) = 3

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

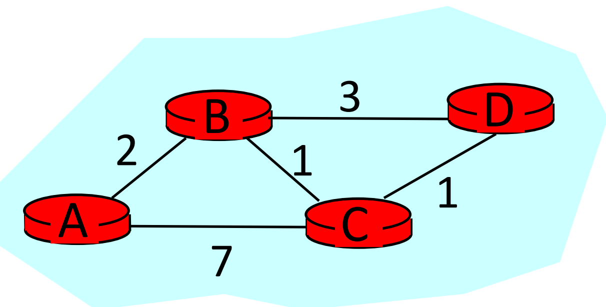
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: End of 1st Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	**3**	**B**
D	**5**	**B**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	**2**	**C**

Node C

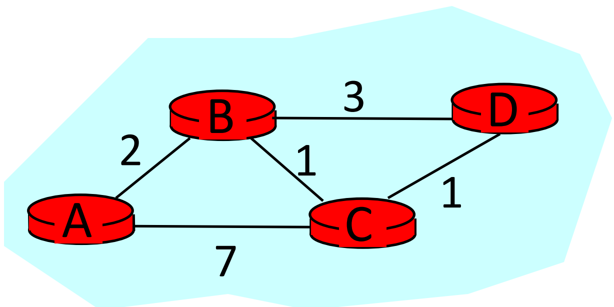
Dest.	Cost	NextHop
A	**3**	**B**
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	**4**	**B**
B	3	B
C	1	C

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Example: End of 3rd Iteration




```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

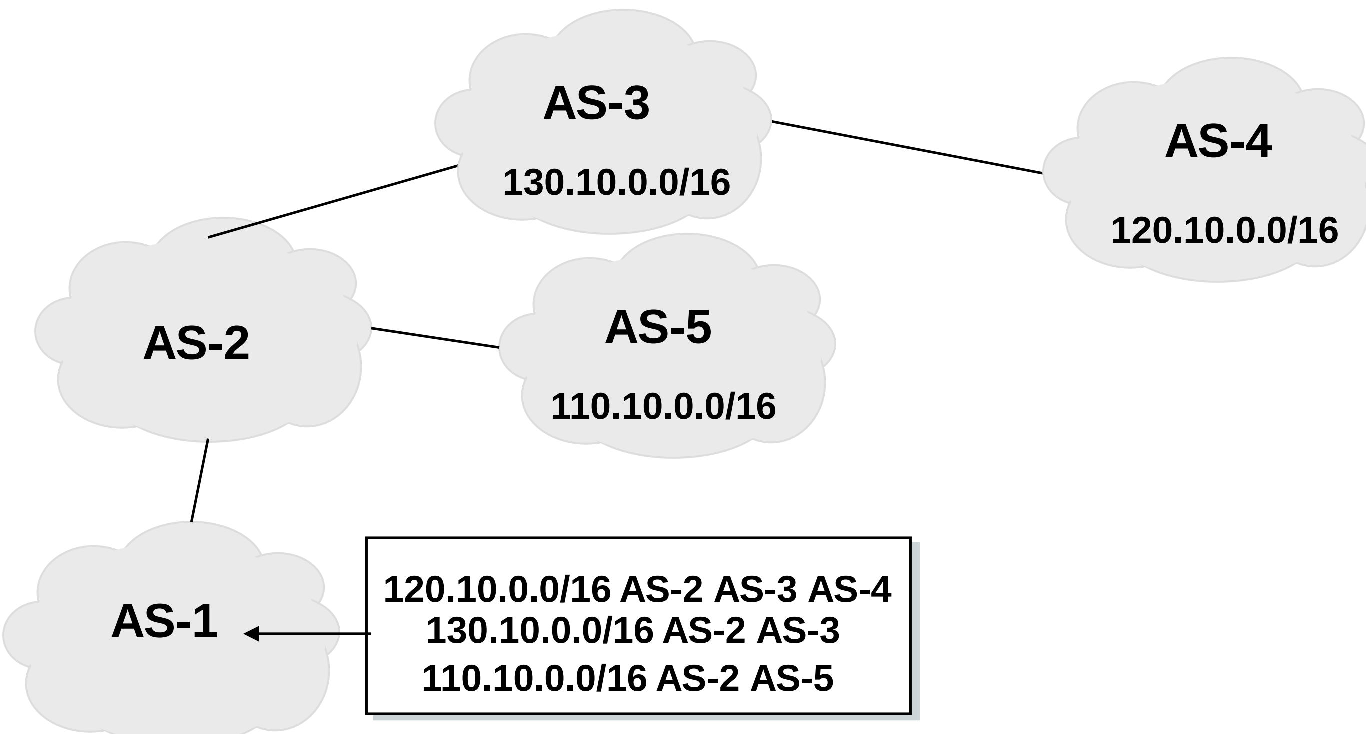
Node D

Dest.	Cost	NextHop
A	4	C
B	2	C
C	1	C

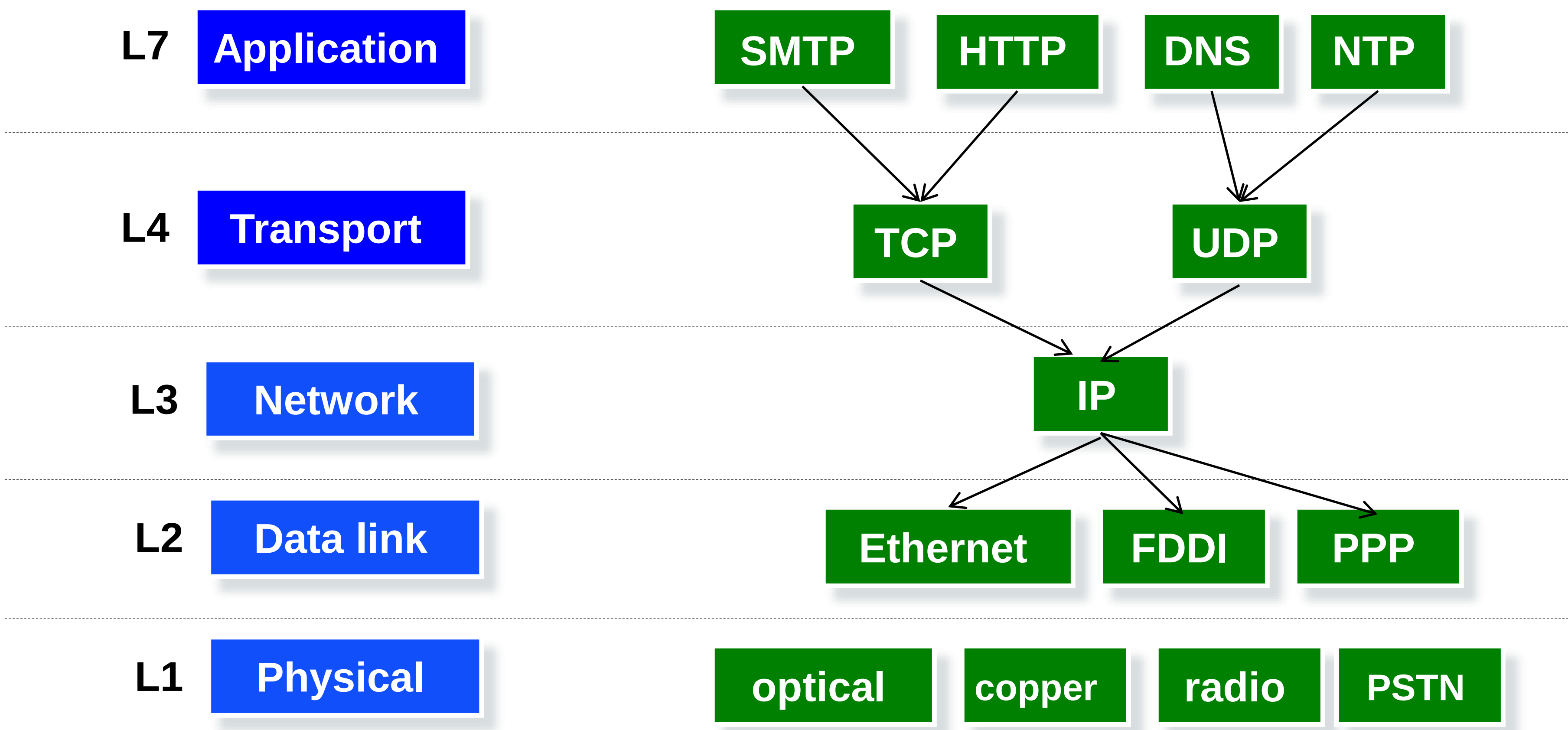
Nothing changes → algorithm terminates

BGP: a Path-Vector Protocol

- An AS-path: sequence of AS's a route traverses
- Used for loop detection and to apply policy
- *Possible* default choice: route with fewest # of AS's



Protocols at different layers



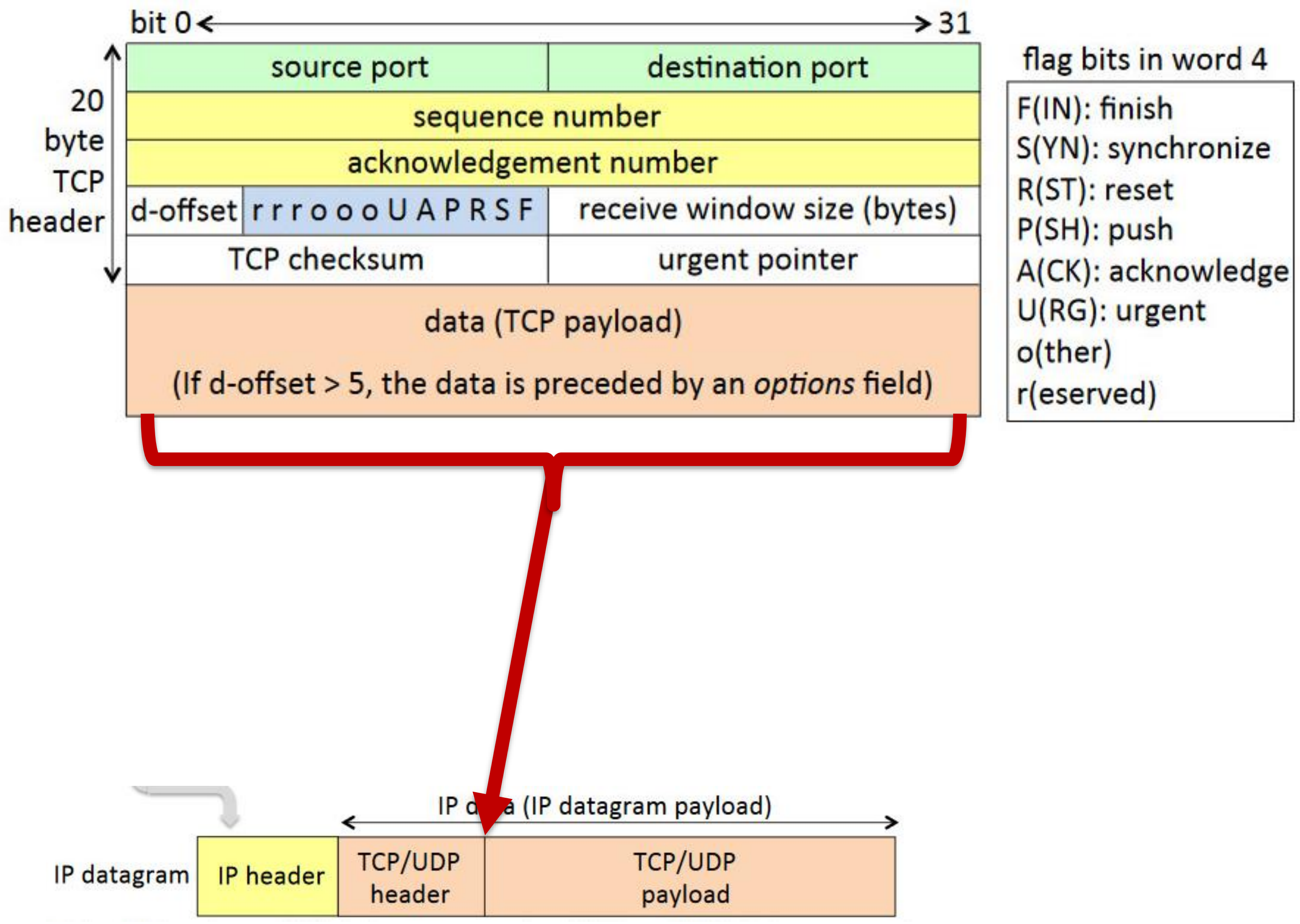
Goal: Get <u>ALL</u> of the data to its destination

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

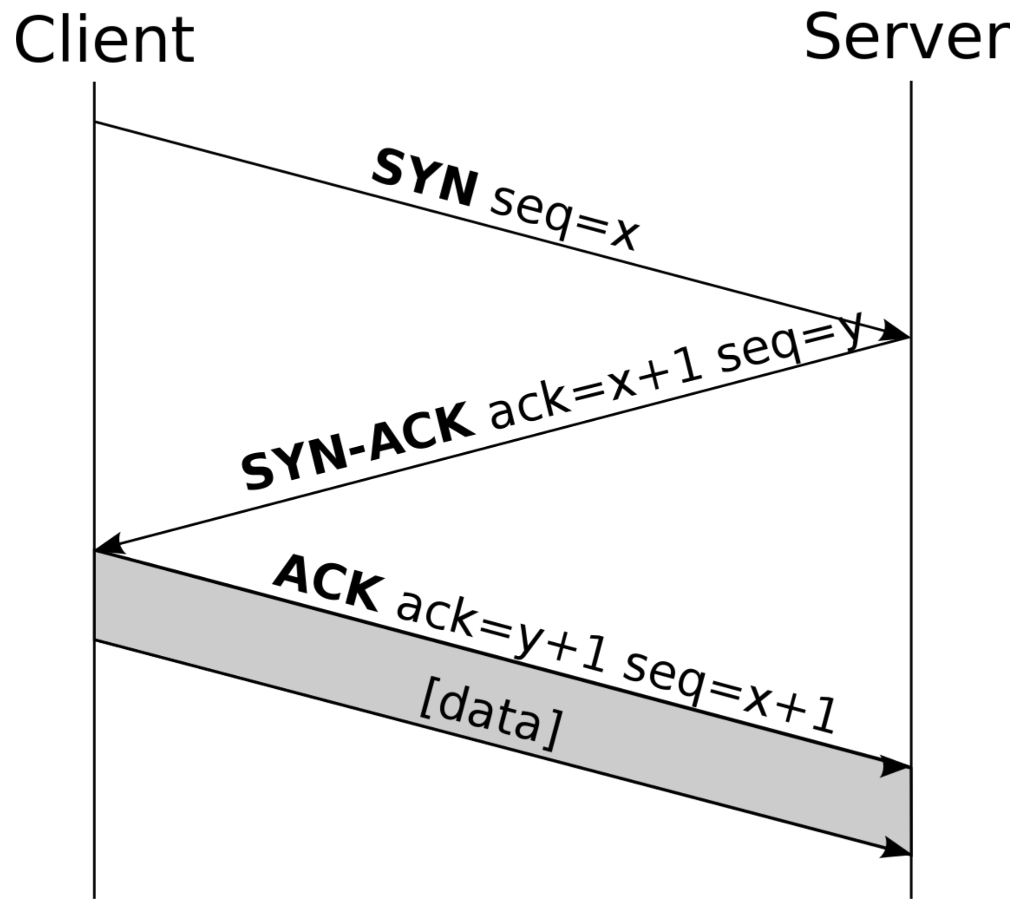


bit 0
31
20 byte TCP header
source port
destination port
sequence number
acknowledgement number
d-offset
r r r o o o U A P R S F
receive window size (bytes)
TCP checksum
urgent pointer
data (TCP payload)
(If d-offset > 5, the data is preceded by an *options* field)
flag bits in word 4
F(IN): finish
S(YN):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

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

Common TCP 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 Sequence Numbers

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

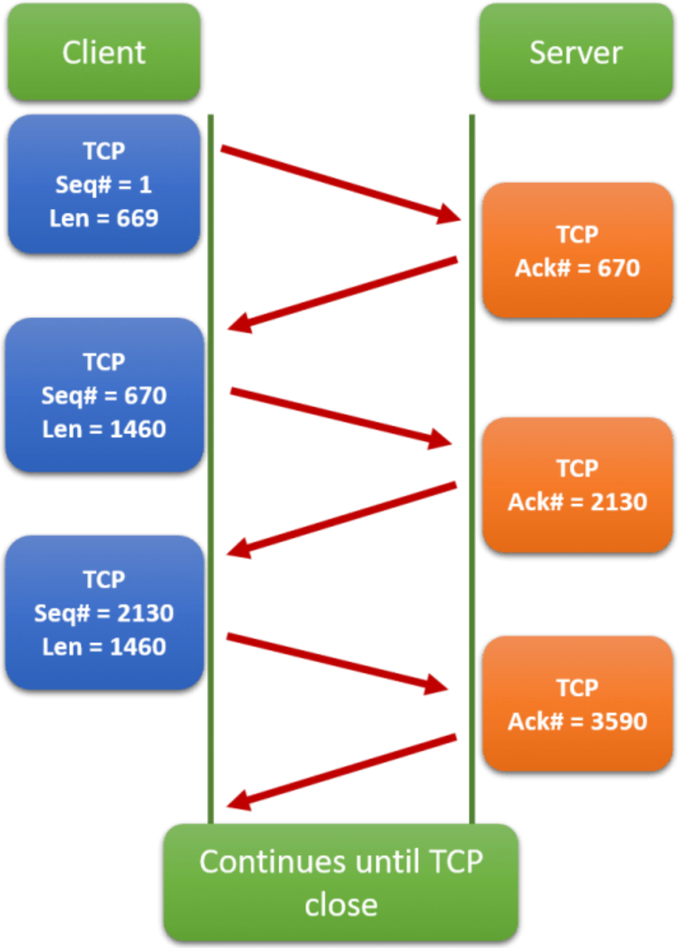
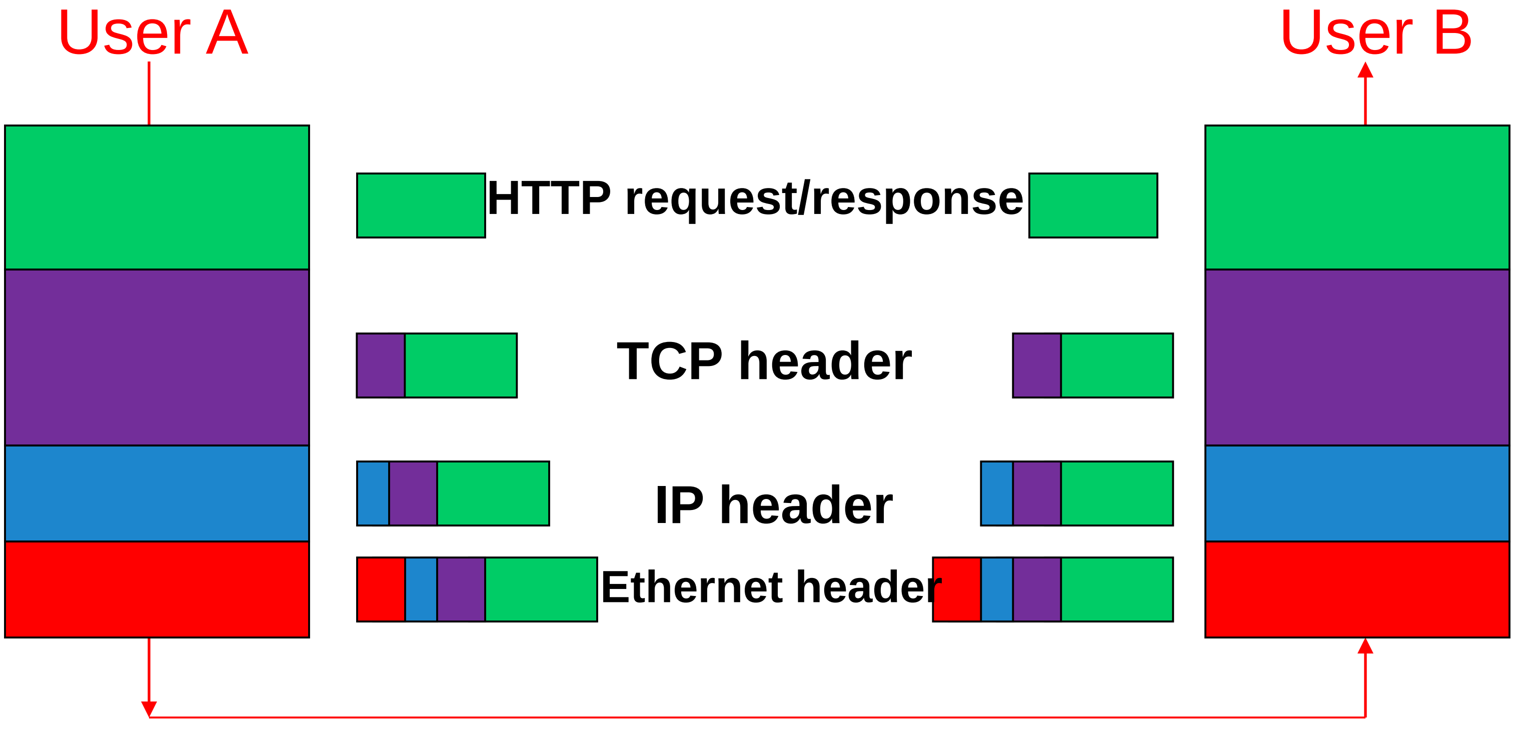
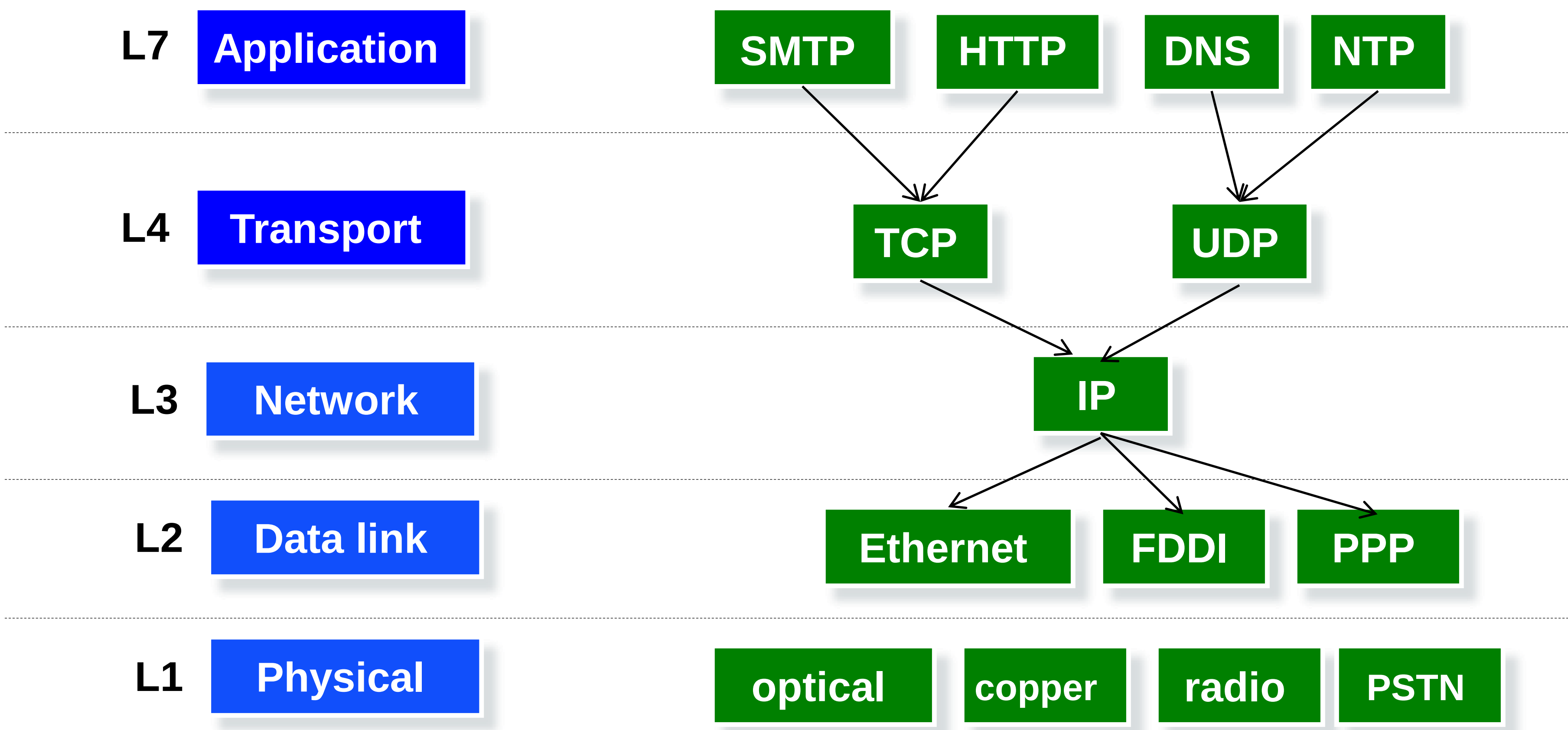


Image from https://madpackets.com/2018/04/25/tcp-sequence-and-acknowledgement-numbers-explained/

Layer Encapsulation: Protocol Headers



Protocols at different layers



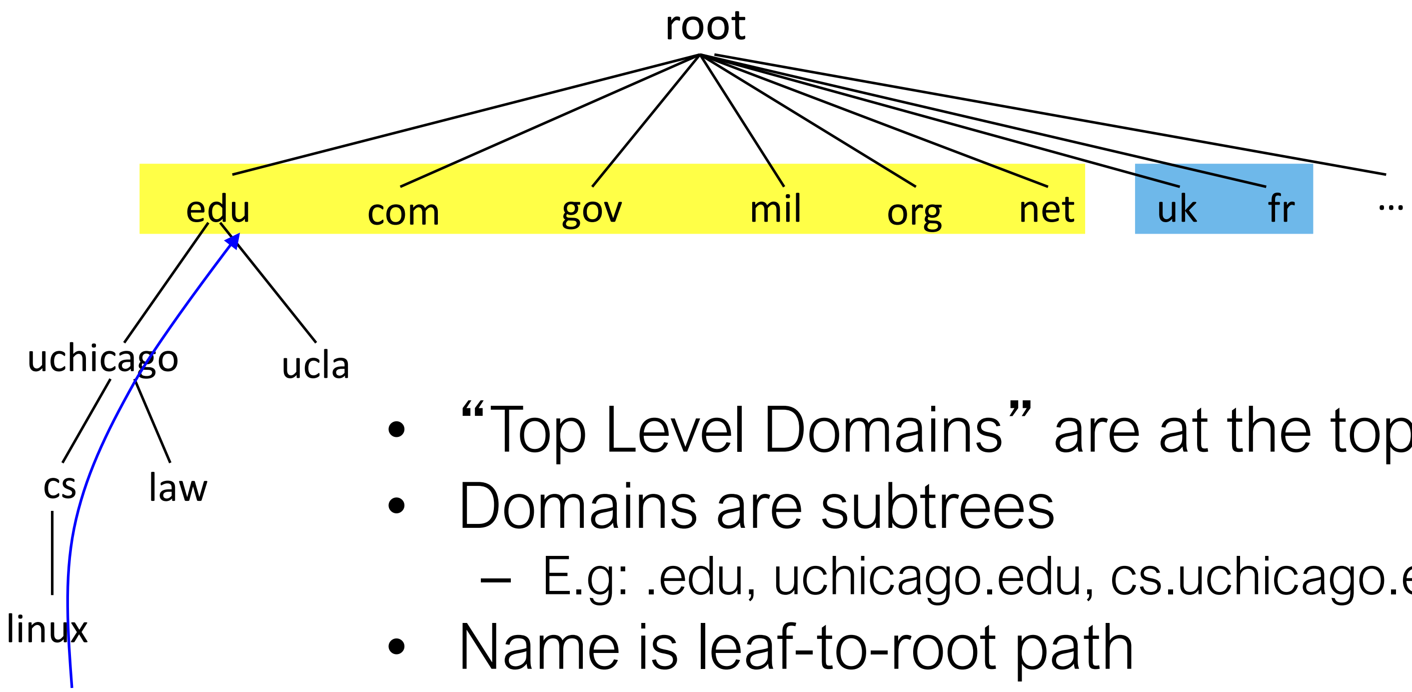
Goal: Be addressable in ways humans can remember on the Internet

Solution: Domain Names

DNS (Domain Name System)

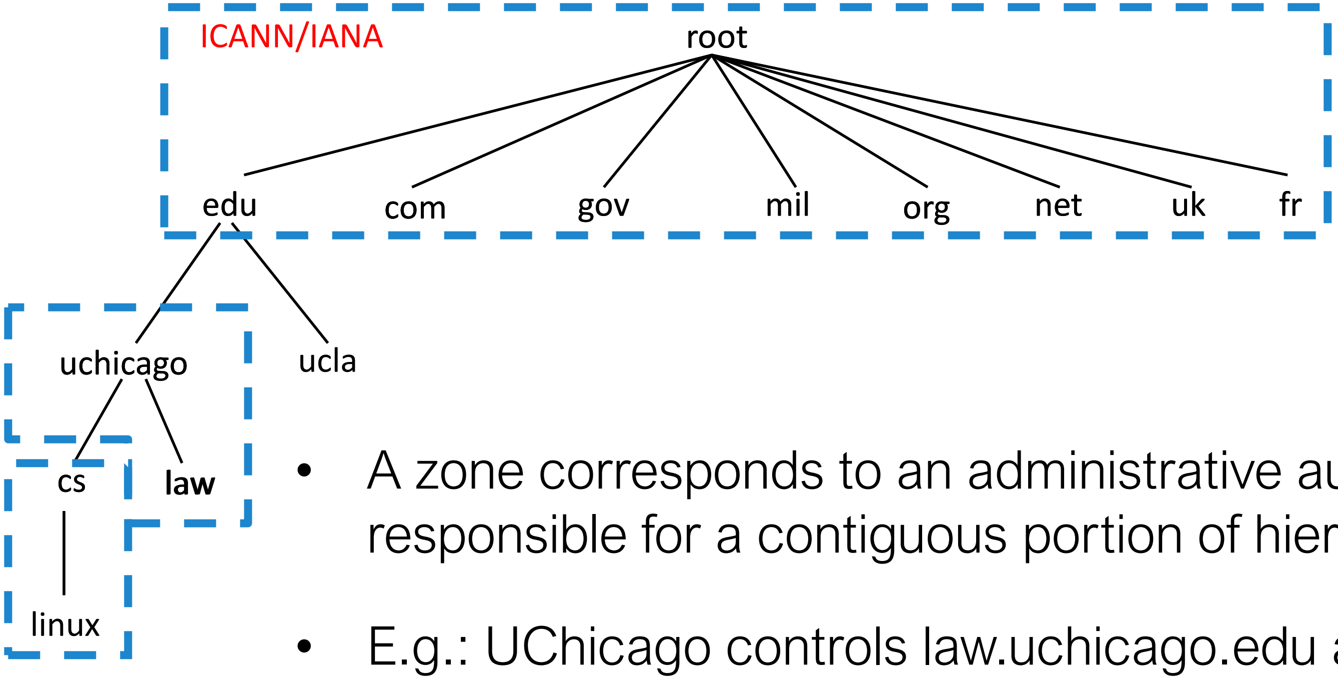
- Host addresses: e.g., *128.135.11.239*
  - a number used by protocols
  - conforms to network structure (the "where")
- Host names: e.g., *super.cs.uchicago.edu*
  - usable by humans
  - conforms to organizational structure (the "who")
- Domain Name System (DNS) is how we map from one to other
  -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

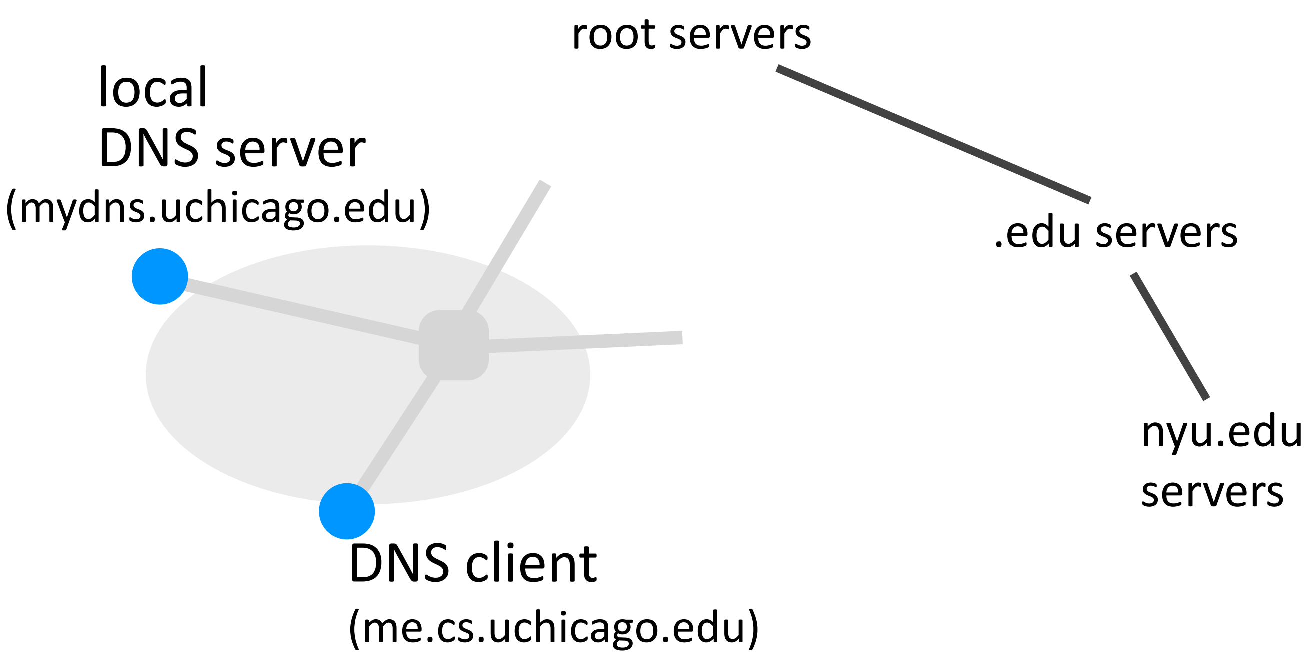
- 13 root servers (labeled A-M; see http://www.root-servers.org/)
- All replicated via anycast

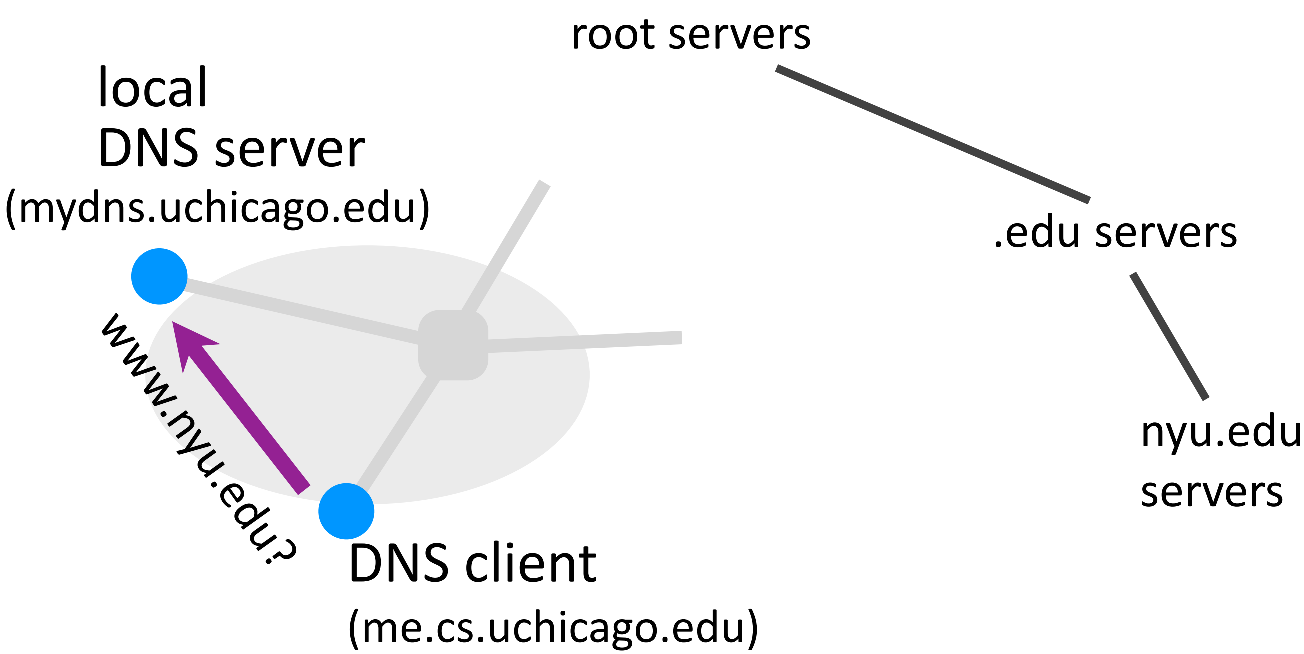
DNS Records

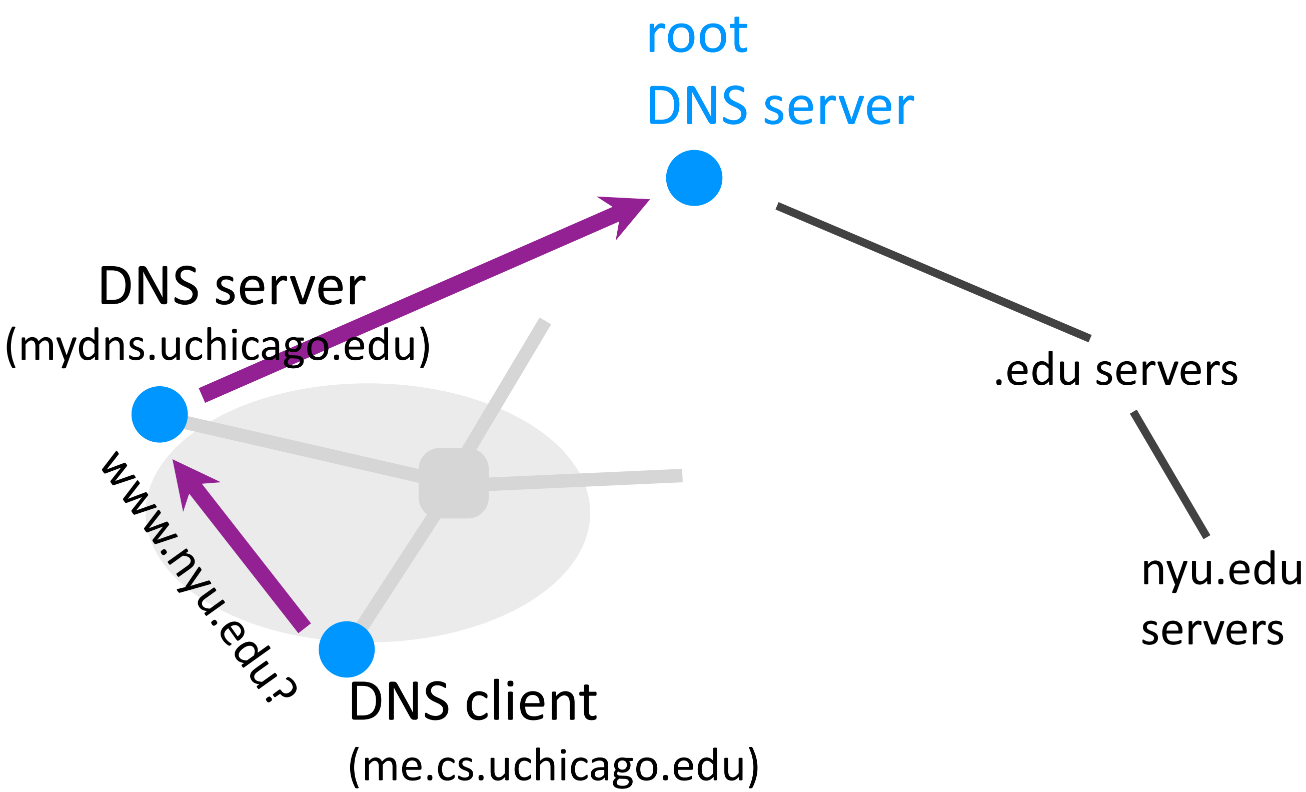
- DNS servers store Resource Records (RRs)
  - RR is (name, value, type, TTL)
- Type = A: *(→ Address)*
  - name = hostname
  - value = IP address
- Type = NS: *(→ Name Server)*
  - name = domain
  - value = name of dns server for domain
- Type = MX: *(→ Mail eXchanger)*
  - name = domain in email address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

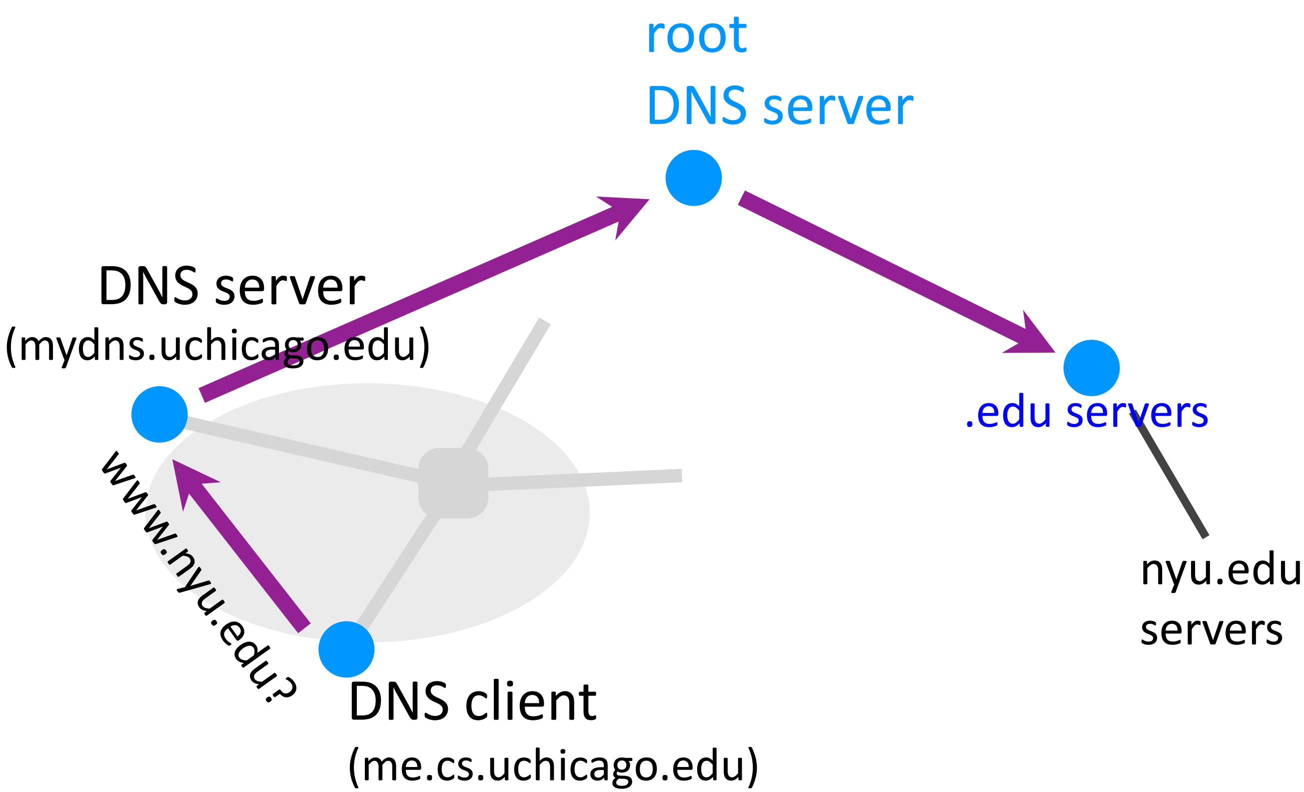
Inserting Resource Records into DNS

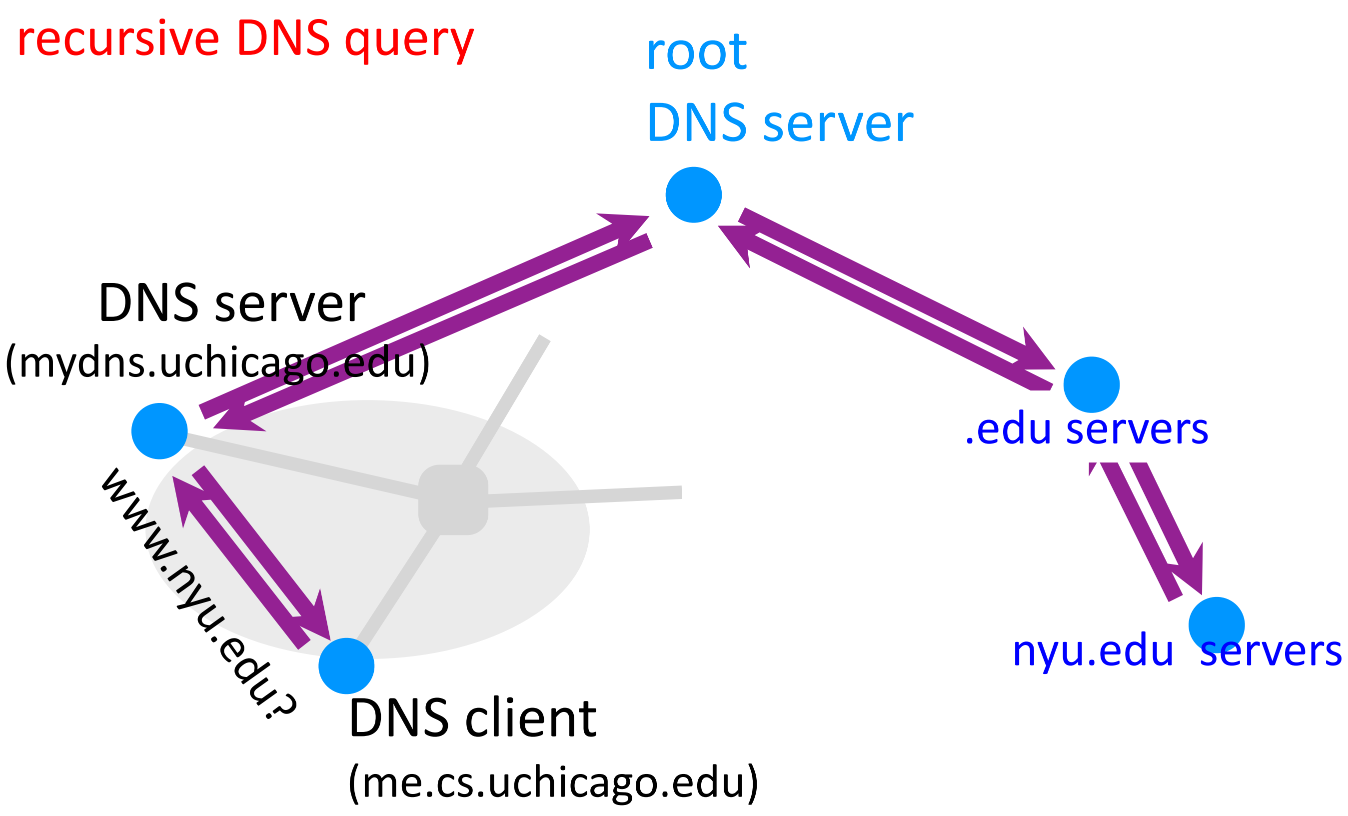
- Example: you want “blaseur.com”
- Register blaseur.com at registrar (e.g., GoDaddy)
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts into the .com TLD server who your name servers are
- Store resource records in your server
  - e.g., type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

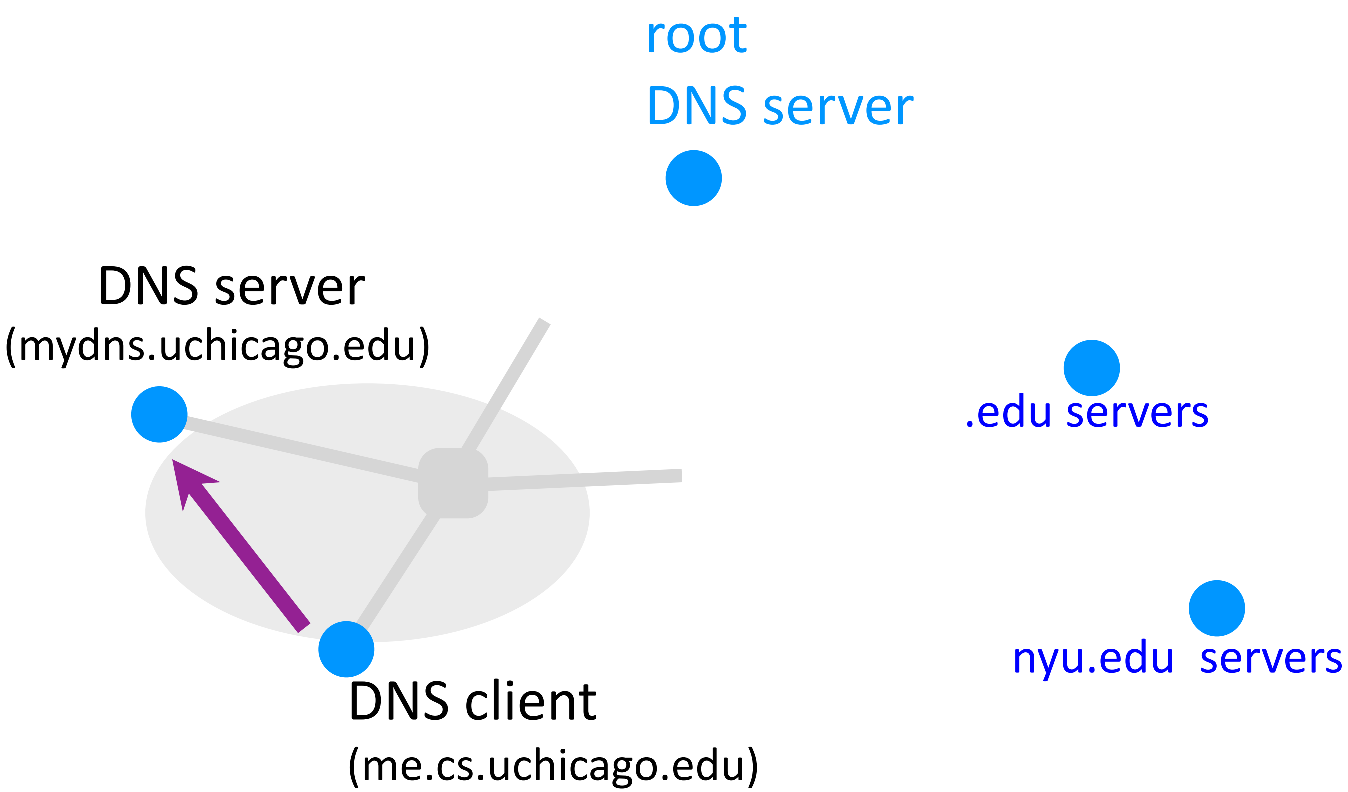

root servers
local
DNS server
(mydns.uchicago.edu)
.edu servers
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)

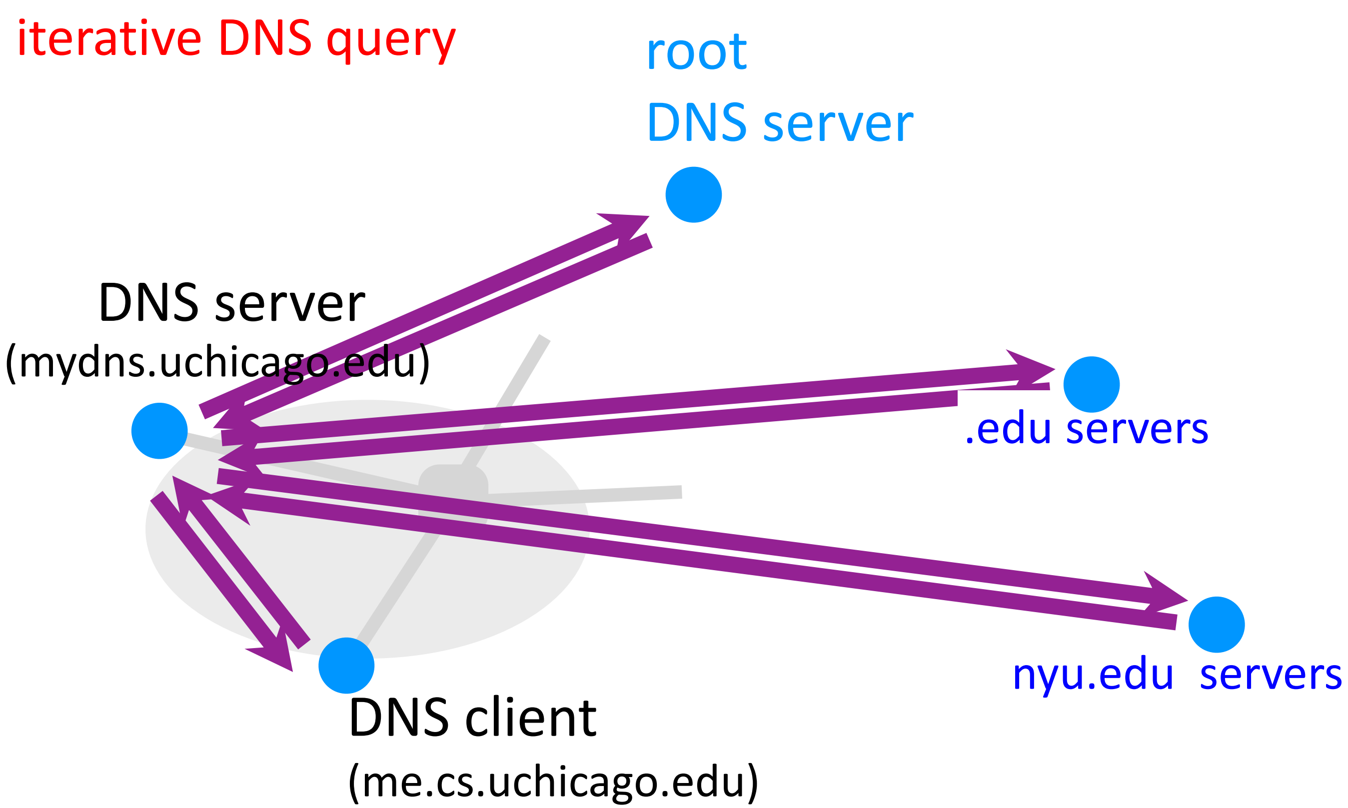

root servers
local
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)


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

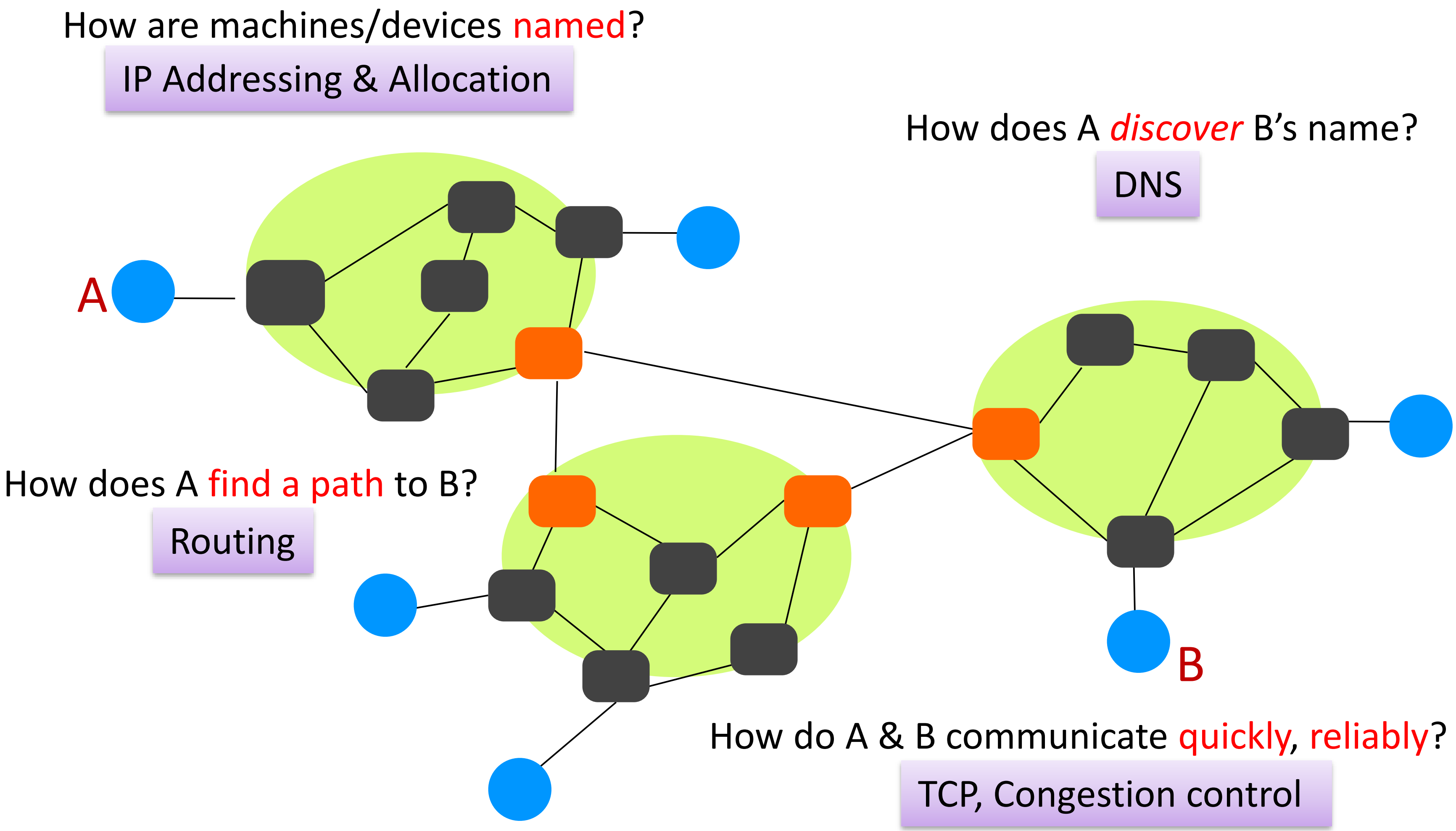

root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
nyu.edu servers
DNS client
(me.cs.uchicago.edu)


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

DNS FAQs

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

Recap


How are machines/devices named?
IP Addressing & Allocation
How does A discover B's name?
DNS
A
How does A find a path to B?
Routing
B
How do A & B communicate quickly, reliably?
TCP, Congestion control