Networking Basics

Ben Zhao
Oct 20, 2018
CS 232/332

THE UNIVERSITY OF CHICAGO

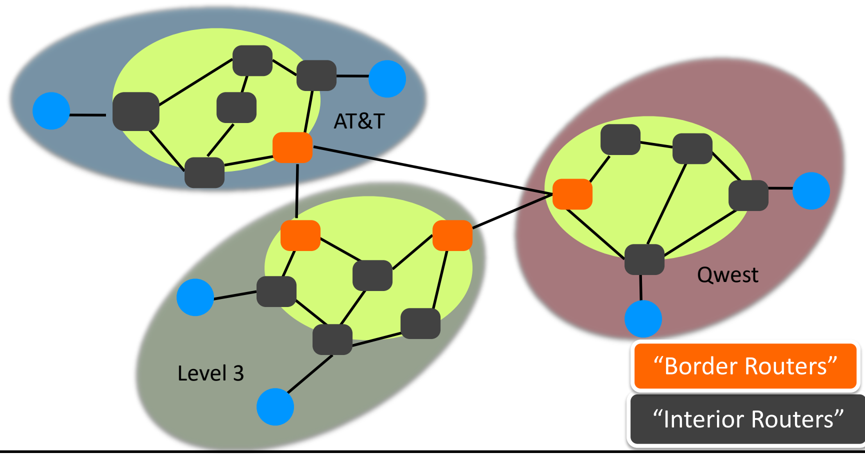
Some Logistics Before We Start…

Date		Topic	Readings
Oct 22 – Oct 29	4	Networking Basics and Basic Attacks	Earlybird; Potemkin
Oct 31 – Nov 14	7	*Blase: Web & Software Security*	
Nov 16 – Nov 21	3	Network Measurements, Underground Markets, Anonymous Routing	Spamalytics; Cybercriminal markets
Nov 26, Nov 28	2	Adversarial Machine Learning	TBA
Nov 30 – Dec 5	3	*DCash & Blase: Current Topics*	TBA

Today: One lecture intro to networking!
Brace yourselves…

The Internet, from 20,000 Feet

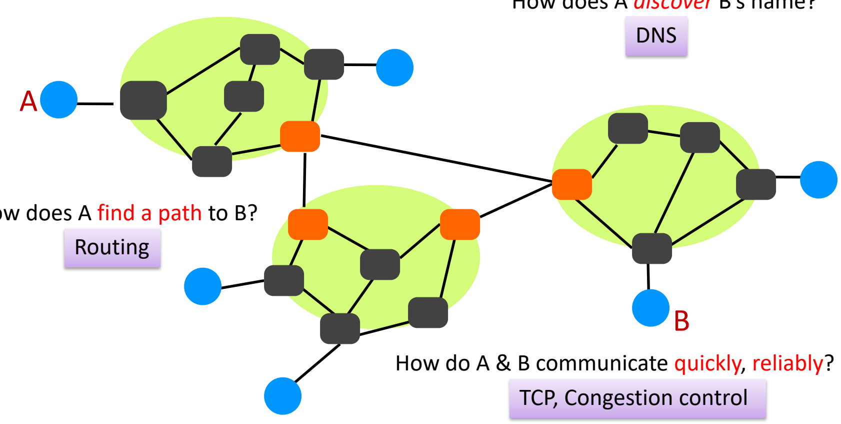
“Autonomous System (AS)” or “Domain”
Region of a network under a single administrative entity



Key Questions

How are machines/devices named?
IP Addressing & Allocation

How does A discover B’s name?
DNS

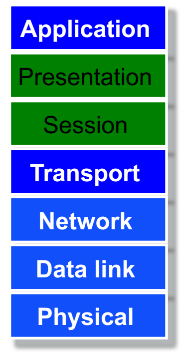


How does A find a path to B?
Routing

How do A & B communicate quickly, reliably?
TCP, Congestion control

Layers

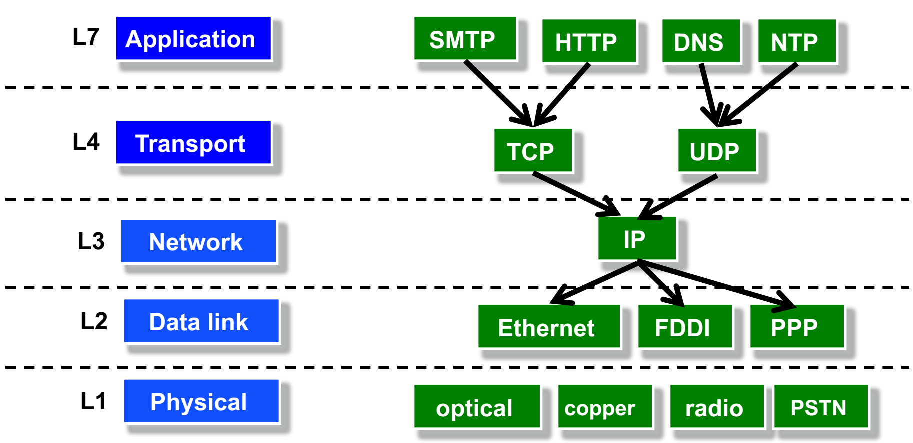
- Layer = a part of a system with well-defined interfaces to other parts



- One layer interacts only with layer above and layer below
- Two layers interact only through the interface between them

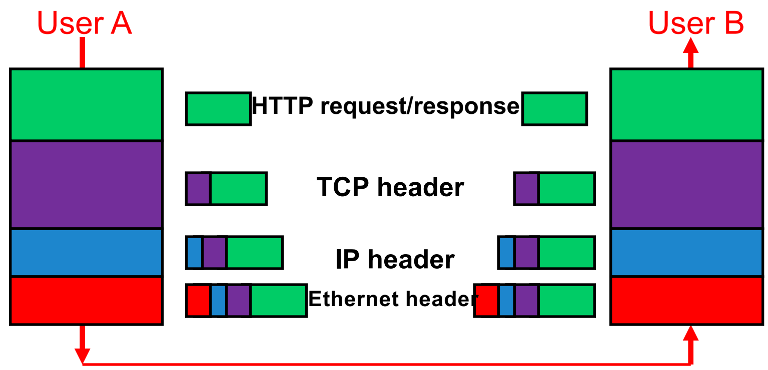
Networking's own Version of Modularity

Protocols at different layers



NOTE: just one network-layer protocol!

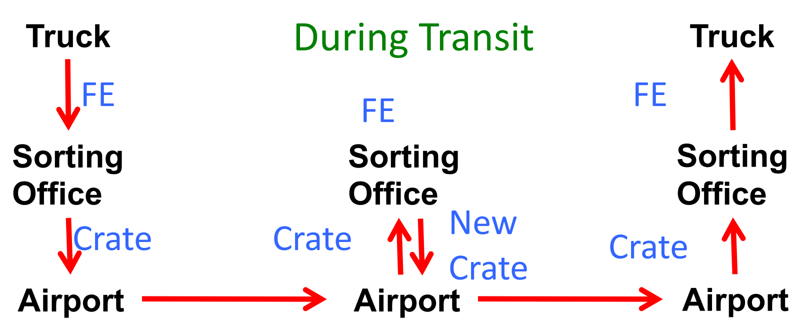
Layer Encapsulation: Protocol Headers



The Path Through FedEx

Higher "Stack" at Ends

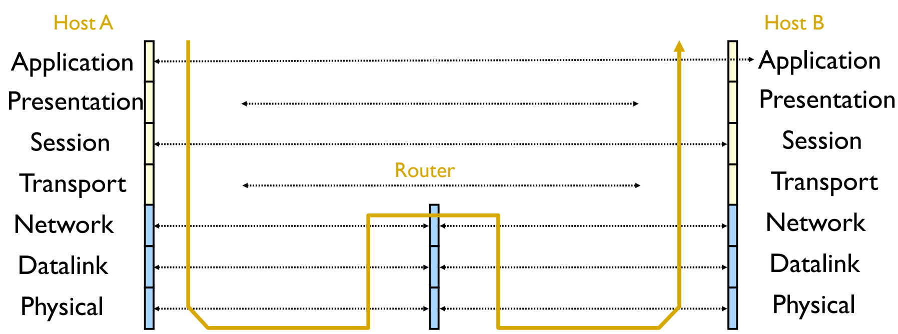
Partial "Stack" During Transit



Deepest Packaging (Envelope+FE+Crate) at the Lowest Level of Transport

Layering on the Internet

- Communication goes down to physical network, then to peer, then up to relevant layer

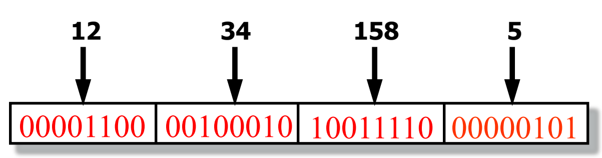


IP Addresses (IPv4)

- Unique 32-bit number associated with host

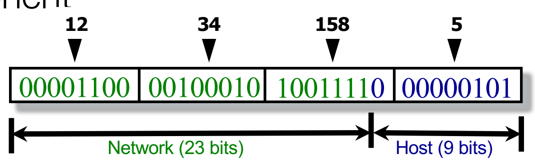
  00001100 00100010 10011110 00000101

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



Hierarchy in IP Addressing

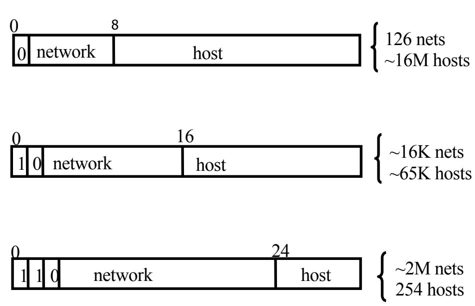
- 32 bits are partitioned into a prefix and suffix components
- Prefix is the network component; suffix is host component



- 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

- 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

The Tour Continues…

- IP Addressing and Allocation
- DNS
- IP Routing
- Transport layer (TCP, congestion control)

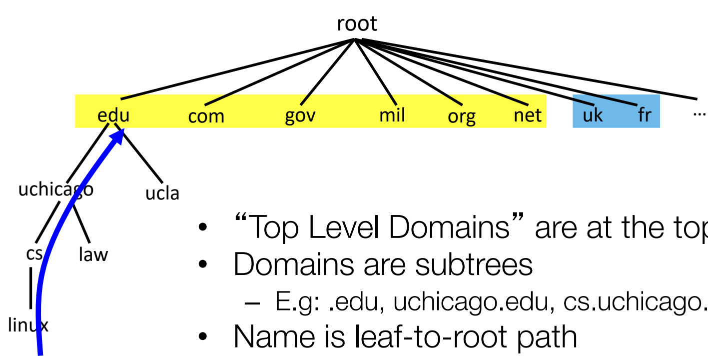
DNS (Domain Name System)

- Host addresses: e.g., *128.135.250.222*
  - a number used by protocols
  - conforms to network structure (the “where”)
- Host names: e.g., *groot.uchicago.edu*
  - mnemonic name 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

DNS: Early days

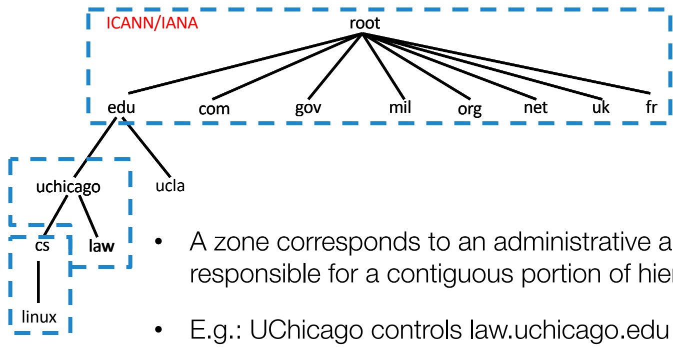
- Mappings stored in a hosts.txt file (in /etc/hosts)
  - maintained by the Stanford Research Institute (SRI)
  - new versions periodically copied from SRI (via FTP)
- As Internet grew, this system broke down
  - SRI couldn't handle the load
  - conflicts in selecting names
  - hosts had inaccurate copies of hosts.txt
- Domain Name System (DNS) invented to fix this
  - First name server implementation done by 4 Berkeley students!

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

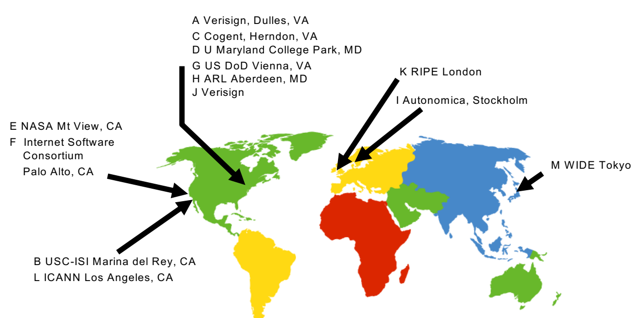
Hierarchical Administration



- 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

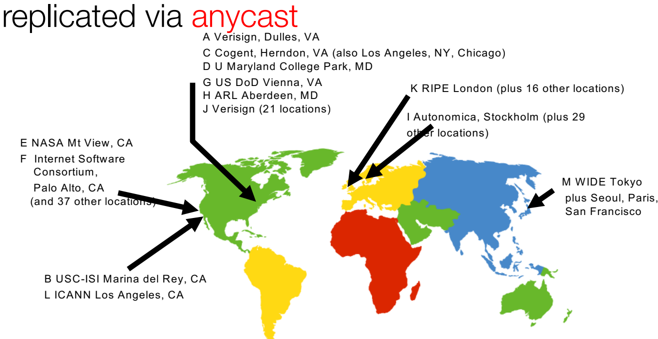
DNS Root Servers

- 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



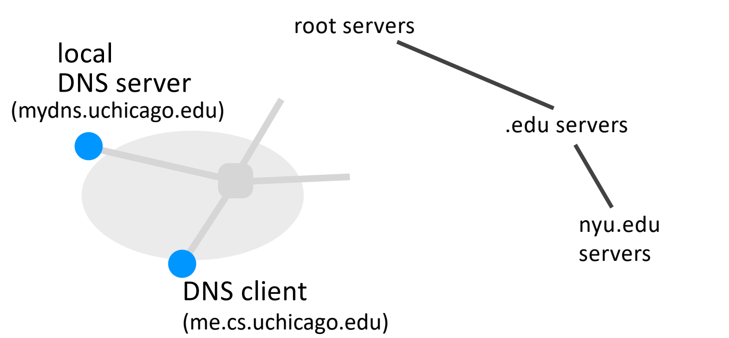
DNS Records

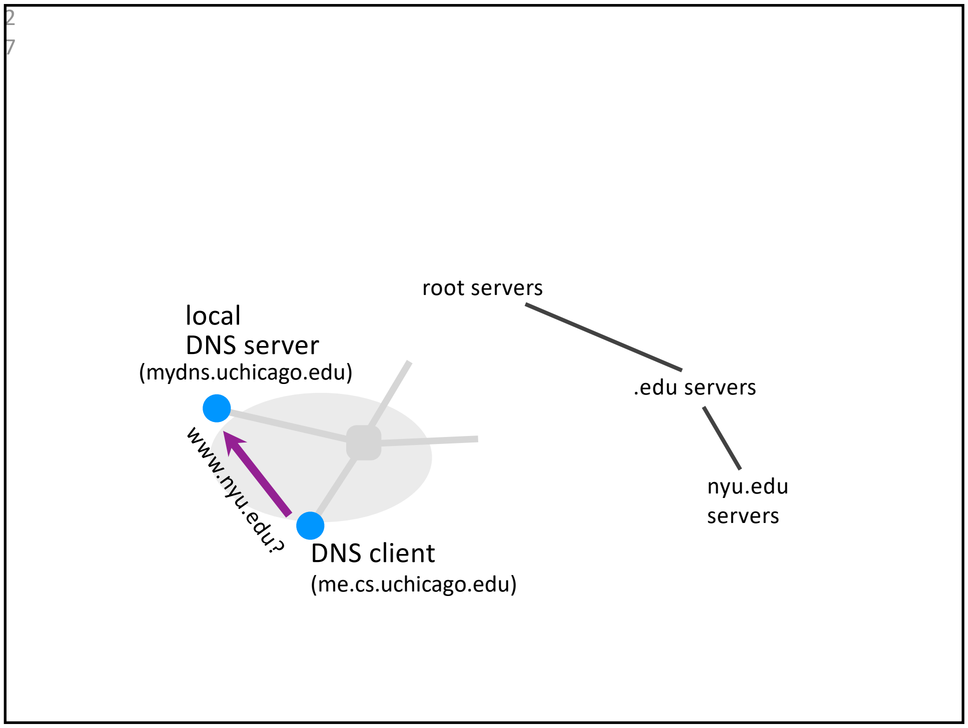
- DNS servers store resource records (RRs)
  - RR is (name, value, type, TTL)
- Type = A: *(→ Address)*
  - name = hostname
  - value = IP address
- Type = NS: *(→ Name Server)*
  - name = domain
  - value = name of dns server for domain
- Type = MX: *(→ Mail eXchanger)*
  - name = domain in email address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

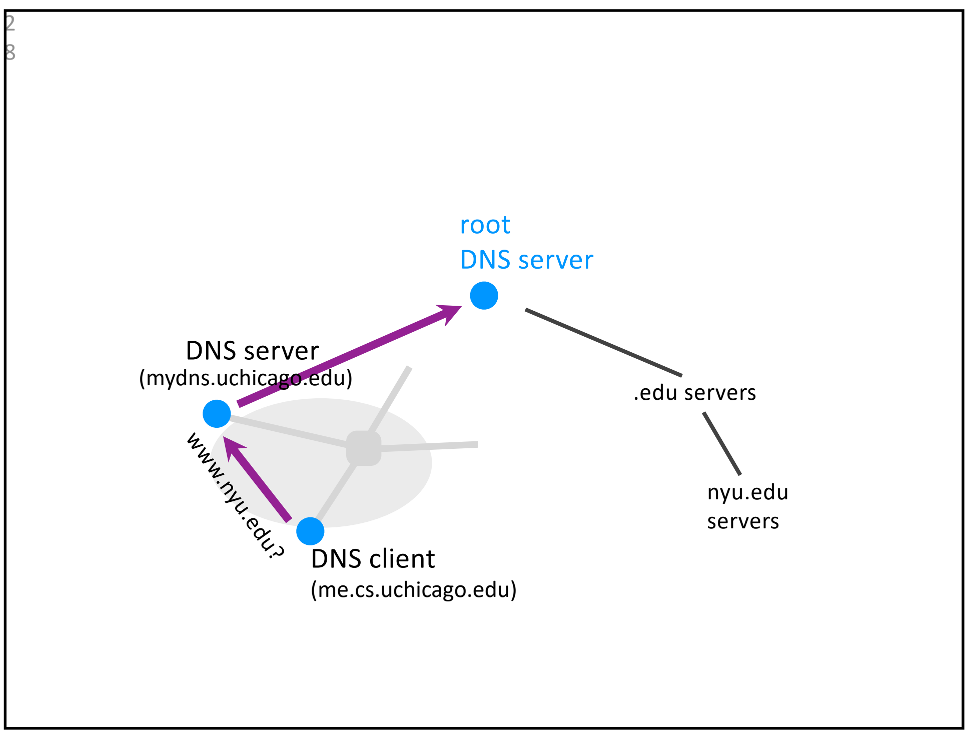
Inserting Resource Records into DNS

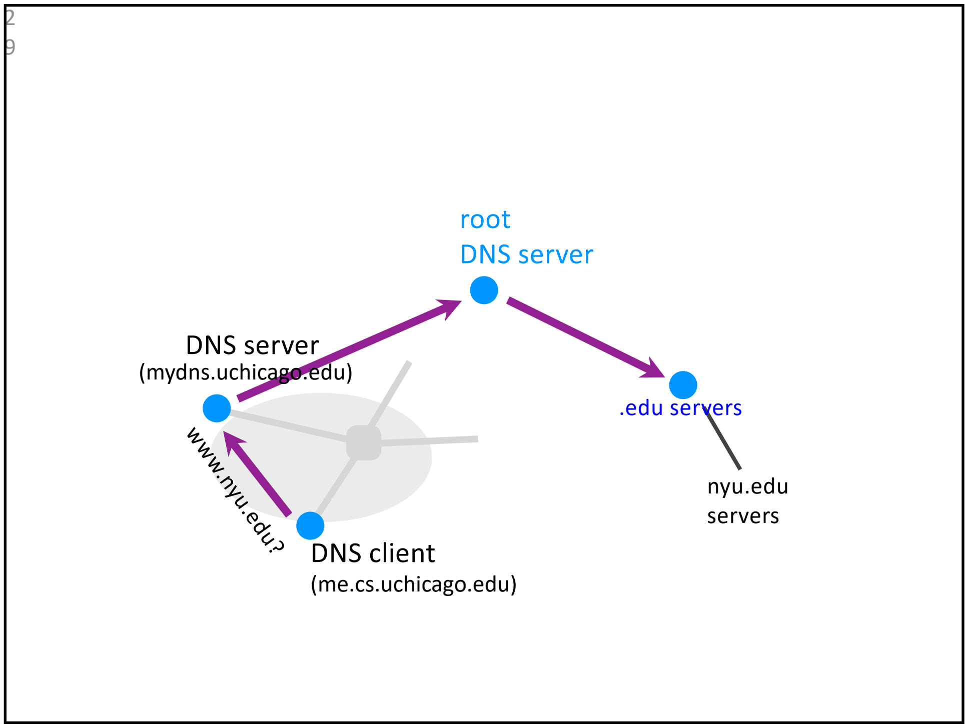
- Example: you just created company "FooBar"
- You get a block of IP addresses from your ISP
  - say 212.44.9.128/25
- Register foobar.com at registrar (e.g., Go Daddy)
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts RR pairs into the .com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.44.9.129, A)
- Store resource records in your server dns1.foobar.com
  - e.g., type A record for www.foobar.com
  - e.g., type MX record for foobar.com

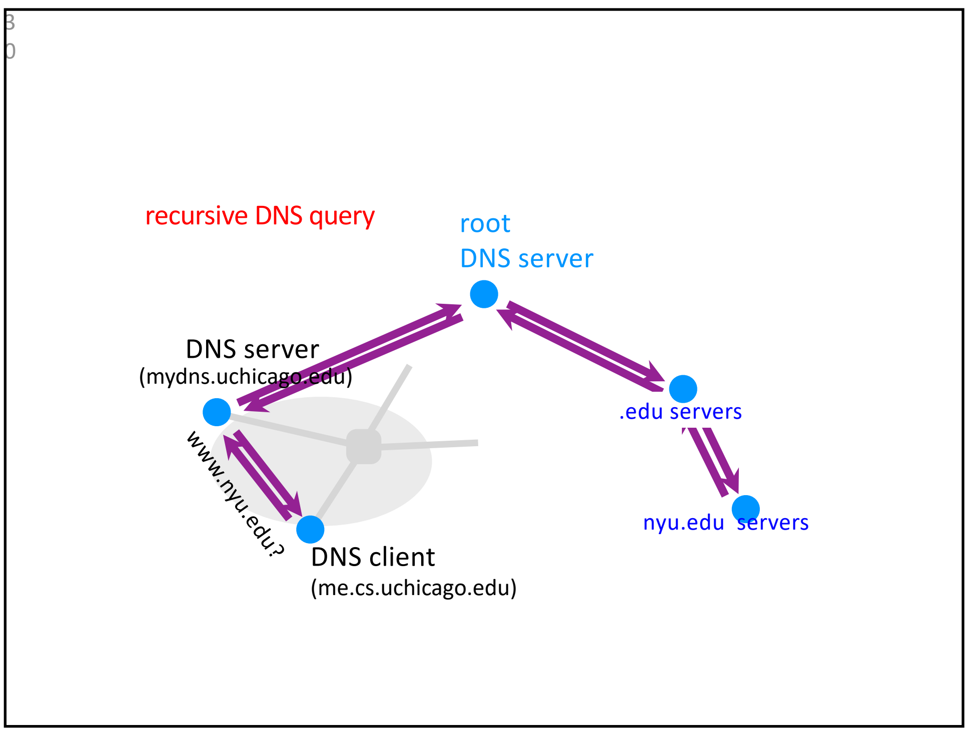
2
6



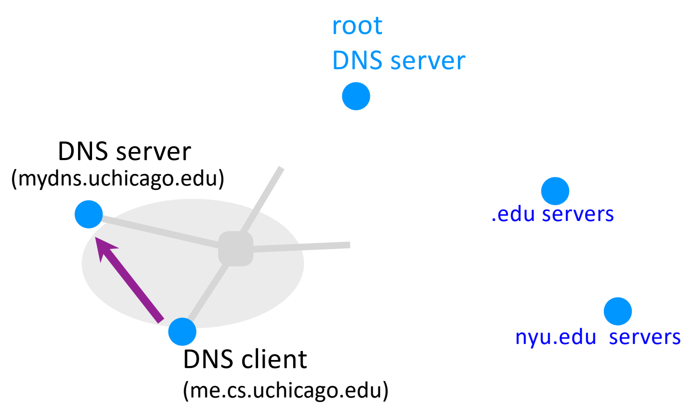

2
7
root servers
local
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)


2
8
root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)

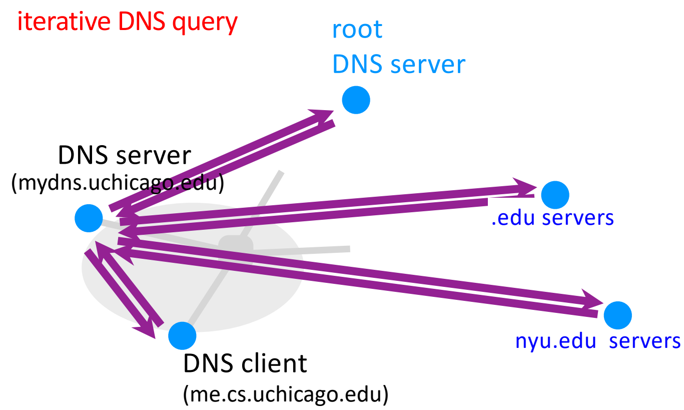

2
9
root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu
servers
DNS client
(me.cs.uchicago.edu)


3
0
recursive DNS query
root
DNS server
DNS server
(mydns.uchicago.edu)
.edu servers
www.nyu.edu?
nyu.edu servers
DNS client
(me.cs.uchicago.edu)

3
1



iterative DNS query



The Tour Continues…

- IP Addressing and Allocation
- DNS
- IP Routing
- Transport layer (TCP, congestion control)

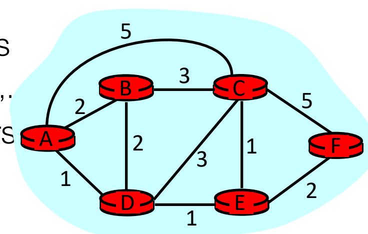
Addressing, Forwarding, Routing

- Addressing: we covered already
- Forwarding: Local router process determines output link (a.k.a "next hop") for each packet
  - *read address from packet's network layer header*
  - *search forwarding table*
- Routing: Network-wide process that determines the content of forwarding tables

  → *determines the end-to-end path for each destination*

Routing

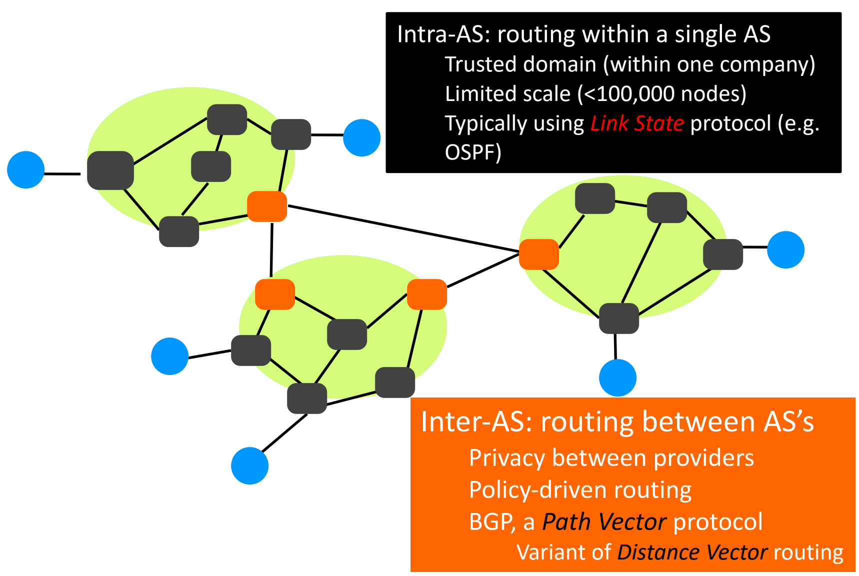
- Goal: determine "good" path through network from source to destination
- Network modeled as a graph
  - Routers → nodes, Link →edges
    - Edge cost: delay, congestion level,..
  - A node knows only its neighbors and the cost to reach them



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

35

Intra-AS & Inter-AS Routing

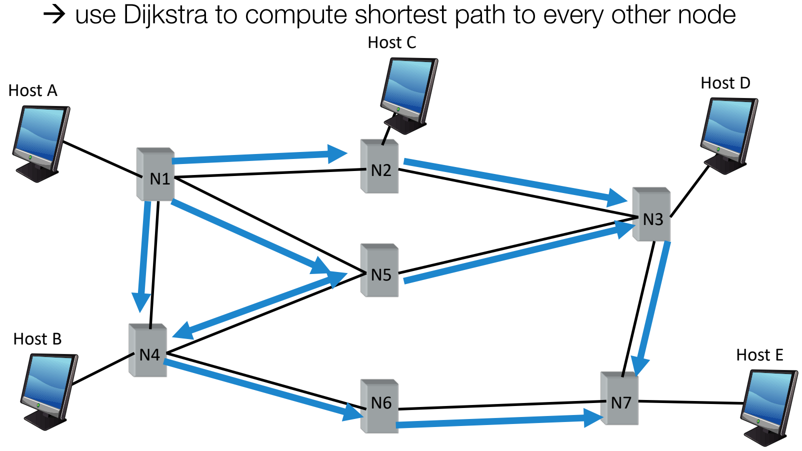


Intra-AS & Inter-AS Routing

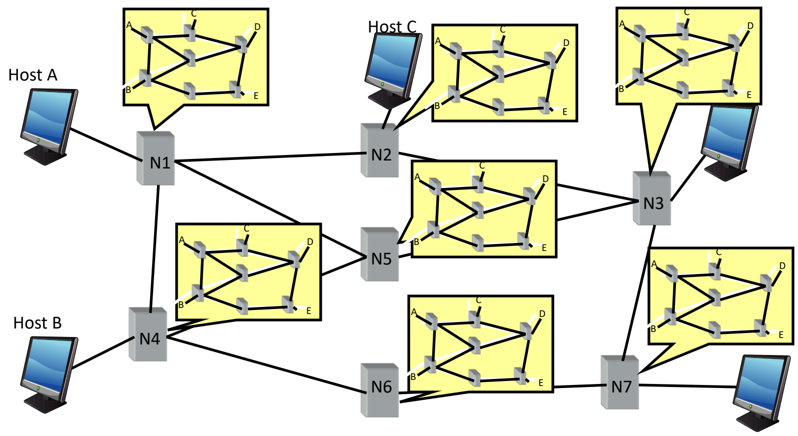
- Intra-AS: routing within a single AS
  - Trusted domain (within one company)
  - Limited scale (<100,000 nodes)
  - Typically using *Link State* protocol (e.g. OSPF)
- Inter-AS: routing between AS's
  - Privacy between providers
  - Policy-driven routing
  - BGP, a *Path Vector* protocol
    - Variant 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

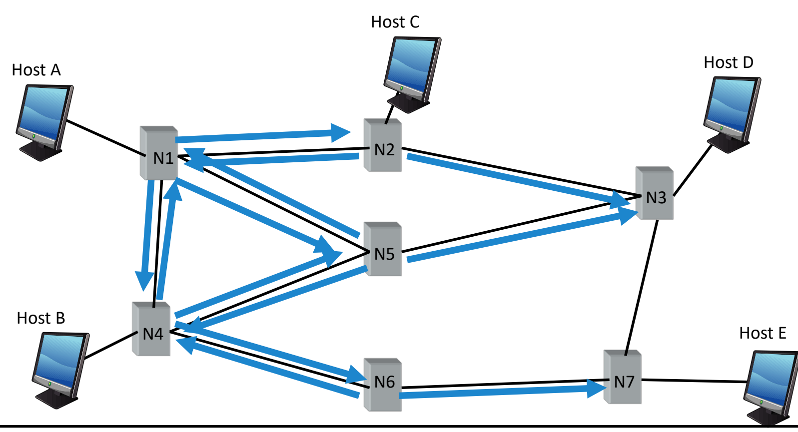


Link State: Node State



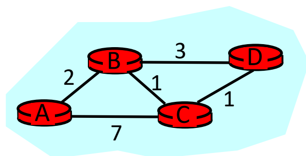
Distance Vector: Control Traffic

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



40

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

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

Node C

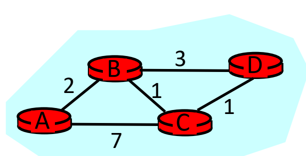
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

4

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))

…
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

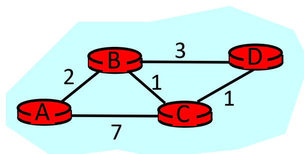
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

4

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

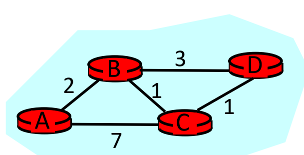
Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
…
7 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
```

4

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

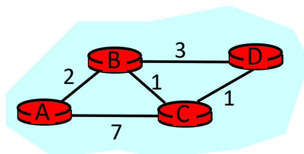
Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
…
7 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
```

4

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

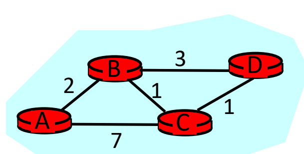
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

4

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

```
…
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

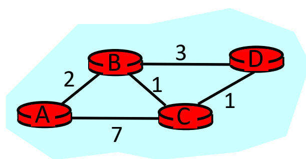
Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

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

4

Example: End of 3nd 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

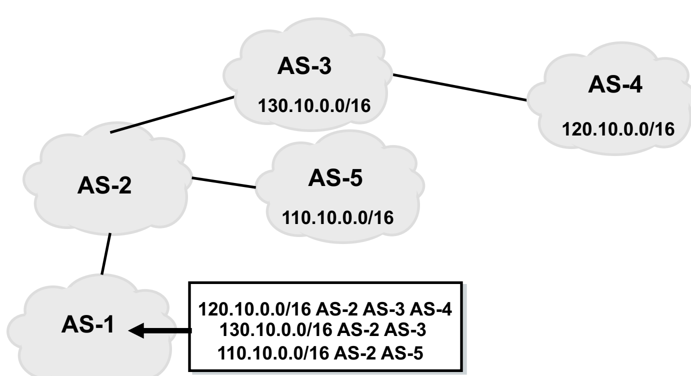
Node D

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
- Default choice: route with fewest # of AS's



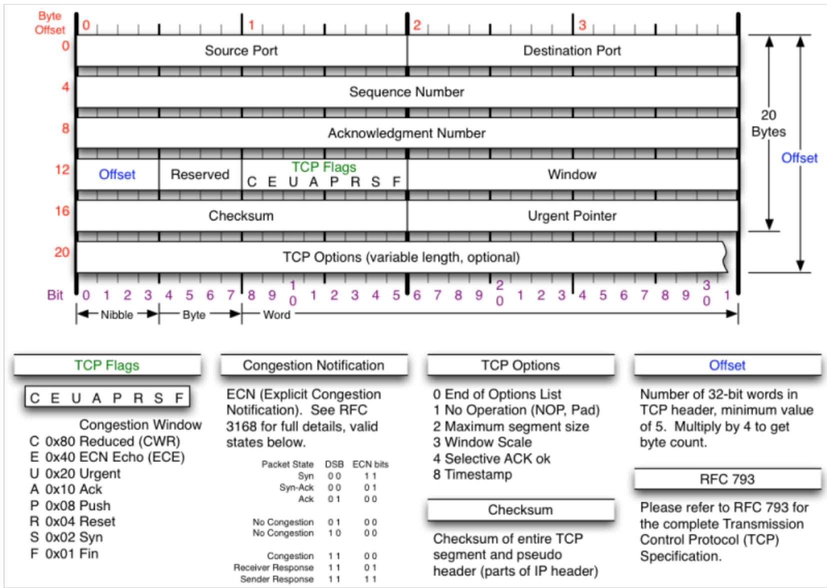
The Tour Continues…

- IP Addressing and Allocation
- DNS
- IP Routing
- Transport layer (TCP, congestion control)

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

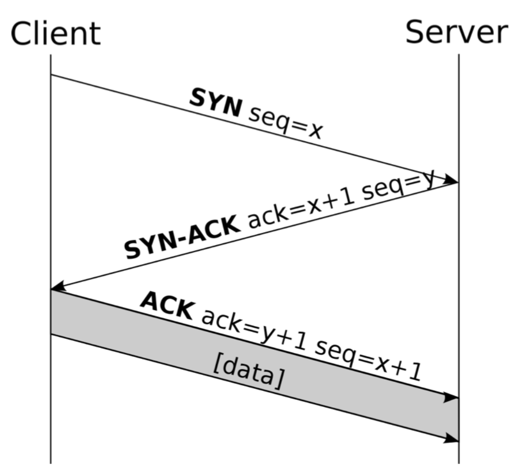
TCP header



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