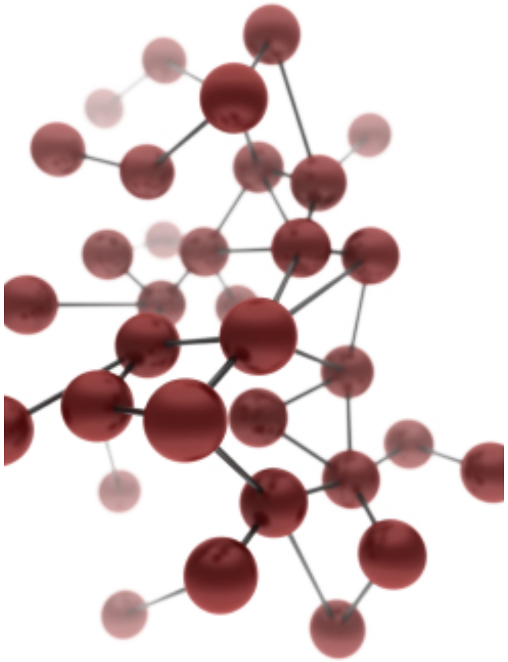


Distance Vector & BGP
Lecture 5

Today

- Least-cost path routing
- Approach 1: link-state routing
- Approach 2: distance-vector routing
- Routing in the Internet

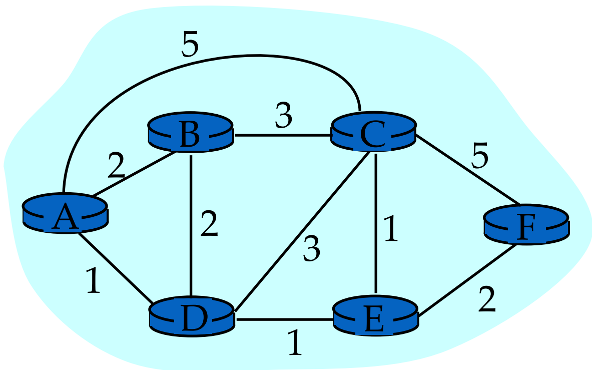
Experiment

- Your job: find the person with the smallest perm # (last 4 digits) in the room
- Ground Rules
  - You may not leave your seat, nor shout loudly across the class
  - You may talk with your immediate neighbors (hint: "exchange updates" with them)
- At the end of 5 minutes, I will pick a victim and ask:
  - who has the smallest perm # in the room? (name , perm #)
  - which one of your neighbors first told you this information?

Go!

Routing Problem

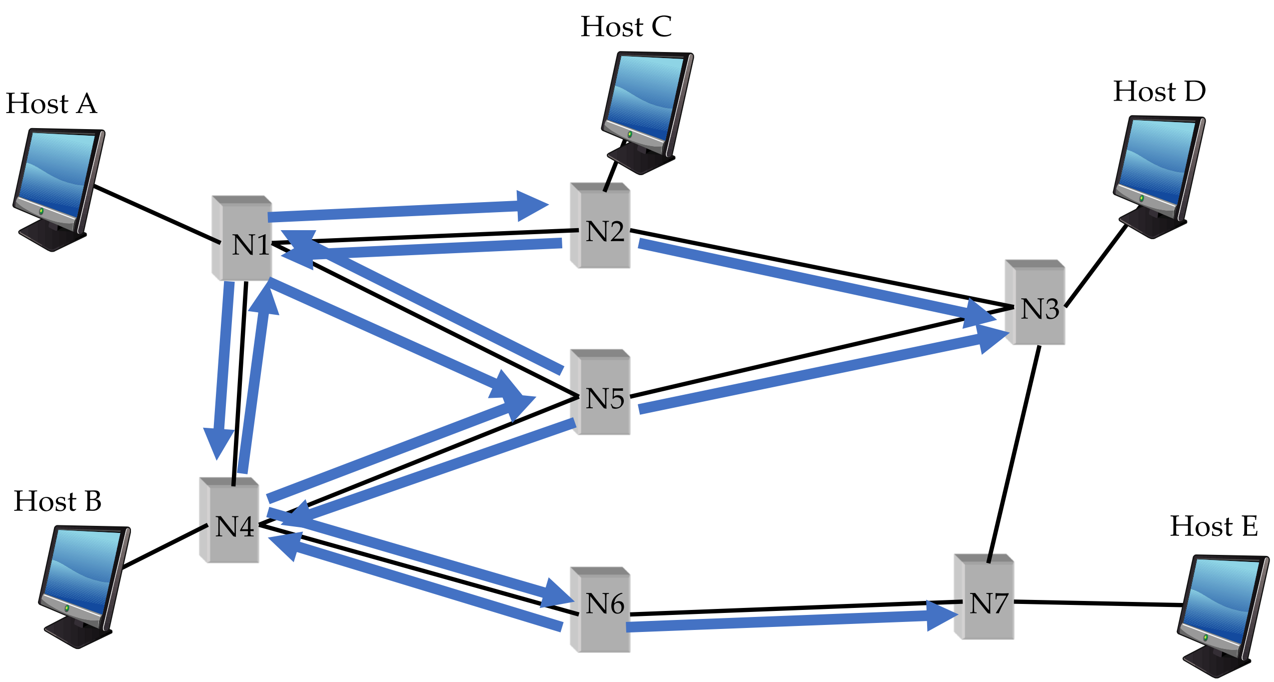
- Assume
  - A network with N nodes, where each edge is associated a cost
  - A node knows only its neighbors and the cost to reach them



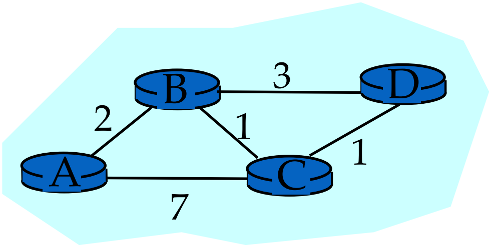
- How does each node learns how to reach every other node along the shortest path?

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

1 ***Initialization:***
2 **for all** neighbors *V* **do**
3 **if** *V* adjacent to *A*
4 D(*A*, *V*) = c(*A*,*V*);
5 **else**
6 D(*A*, *V*) = ∞;
…

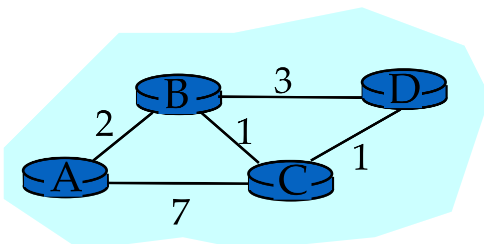
Node C

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

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

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

```
…
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
```

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

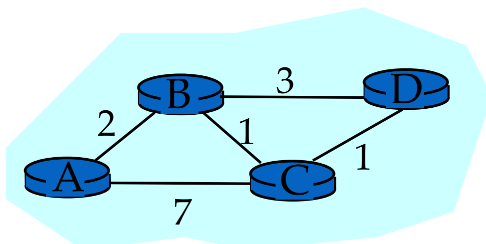
Node C

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

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: 1st Iteration (C → A)



Node A

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

$D(A,D) = \min(D(A,D), D(A,C)+D(C,D))$
$= \min(\infty, 7 + 1) = 8$

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

Node B

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

Node C

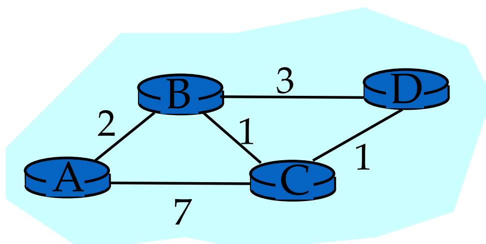
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
…
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

```
…
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
```

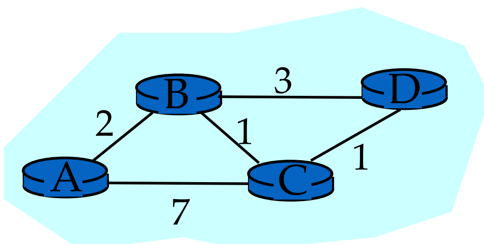
Node C

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: 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
```

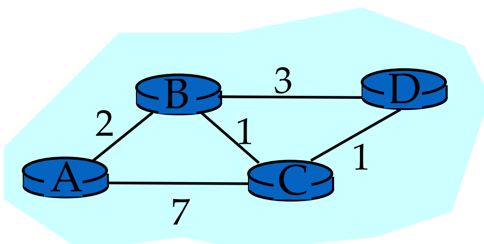
Node C

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

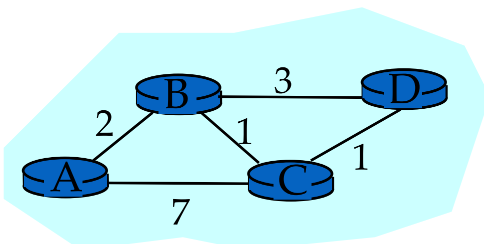
Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

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 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

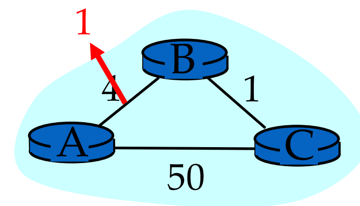
Node D

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

Nothing changes → algorithm terminates

Distance Vector: Link Cost Changes

```
loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) =  D(A,Y) + d
  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 destination Y)
    send D(A, Y) to all neighbors
forever
```



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

D	C	N
A	2	B
B	1	B

time

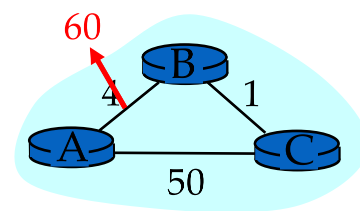
Link cost changes here

Algorithm terminates

"good news travels fast"

DV: Count to Infinity Problem

```
loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
     for all destinations Y through V do
        D(A,Y) =  D(A,Y) + d
   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 destination Y)
     send D(A, Y) to all neighbors
  forever
```



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	8	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	7	B
B	1	B

D	C	N
A	2	B
B	1	B

…

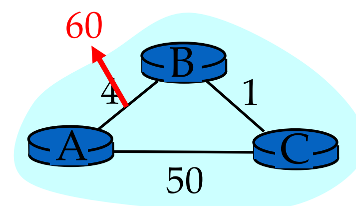
time

"bad news travels slowly"

Link cost changes here; note that B also maintains the shortest route to A for C, which is 6. Thus D(B, A) becomes 6 !

Distance Vector: Poisoned Reverse

- If C routes through B to get to A:
  - *C tells B its (C's) distance to A is infinite (so B won't route to A via C)*
  - *Will this completely solve count to infinity problem?*



NO! multipath loops

Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	**60**	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	**51**	C
C	1	B

D	C	N
A	51	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	**50**	A
B	1	B

D	C	N
A	50	A
B	1	B

D	C	N
A	50	A
B	1	B

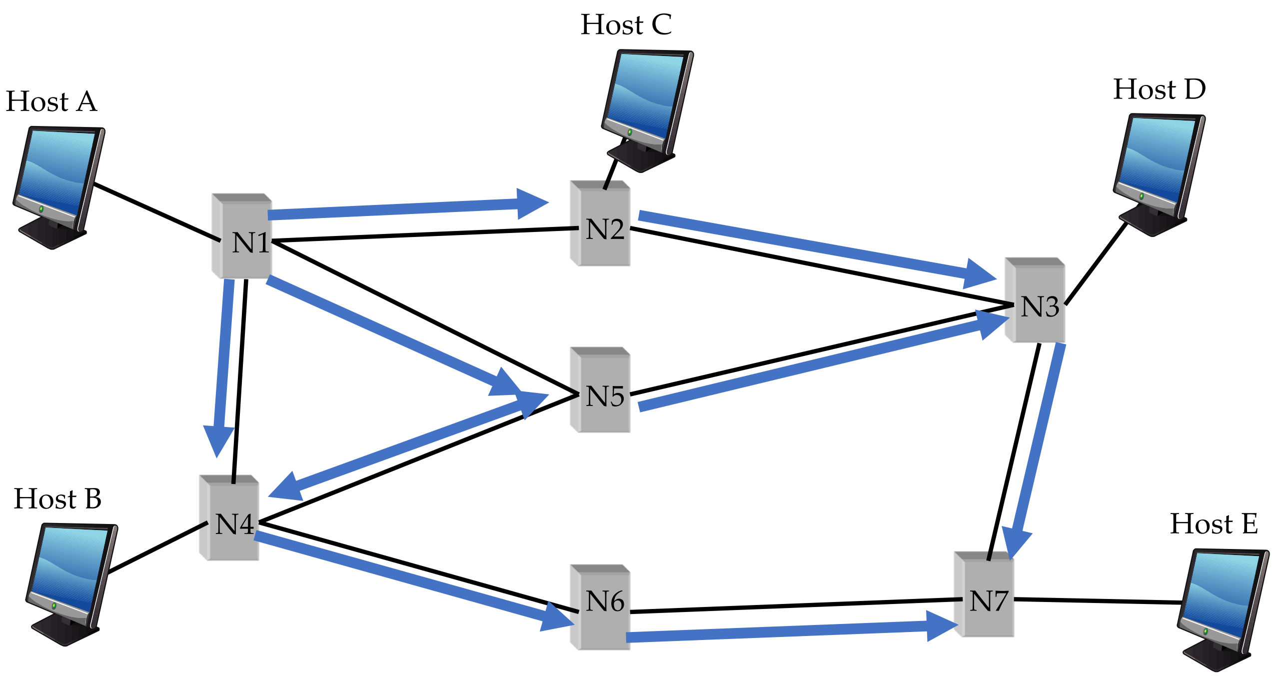
time

Link cost changes here; B updates D(B, A) = 60 as C has advertised D(C, A) = ∞

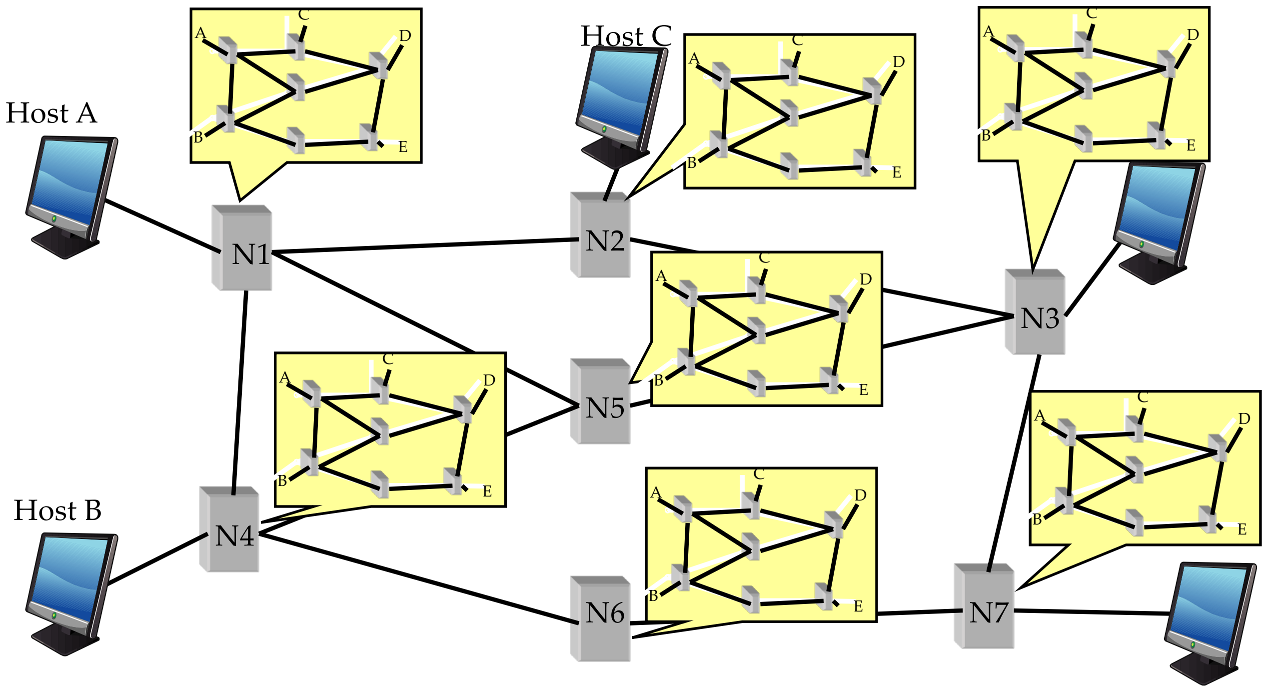
Algorithm terminates

Link State: Control Traffic

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

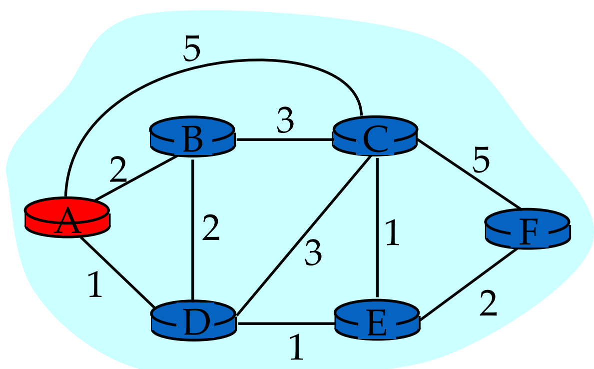


Link State: Node State



Example: Dijkstra's Algorithm

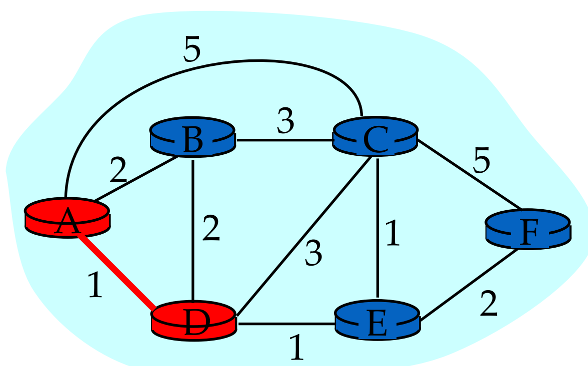
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						



1 ***Initialization:***
2 S = {A};
3 for all nodes *v*
4 if *v* adjacent to *A*
5 then D(v) = c(A,v);
6 else D(v) = ∞
…

Example: Dijkstra's Algorithm

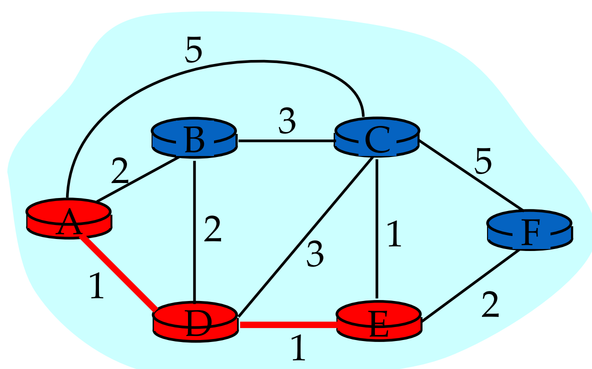
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						




```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
   update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

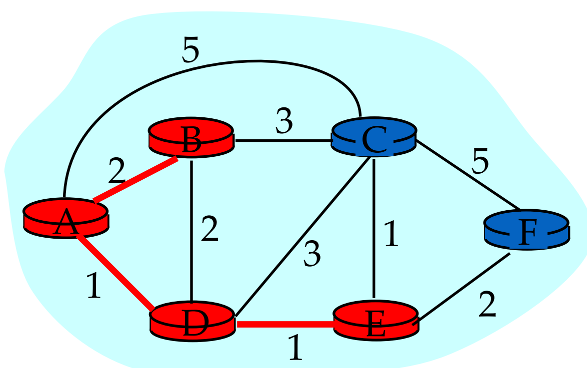
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						




```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
   update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

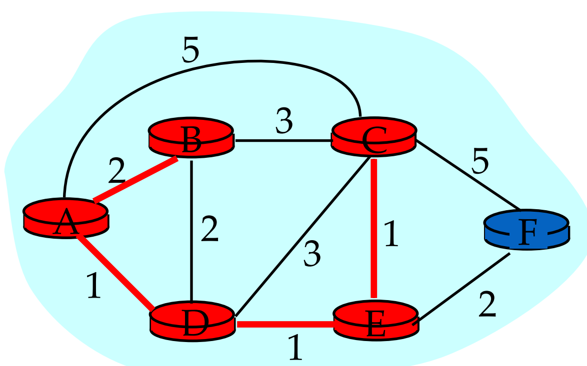
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						




```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
   update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
  until all nodes in S;
```

Example: Dijkstra's Algorithm

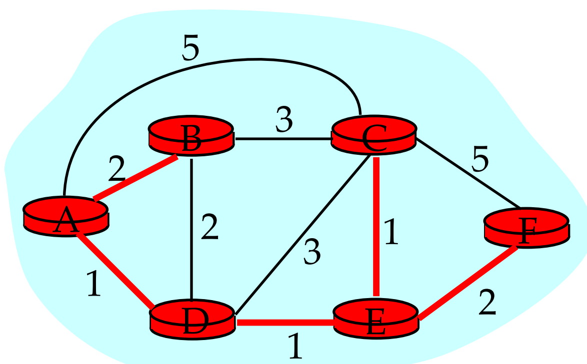
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						




```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
   update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
  until all nodes in S;
```

Example: Dijkstra's Algorithm

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					




```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
   update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Link State vs. Distance Vector

Message complexity

- LS: $O(n^2 * e)$ messages
  - n: number of nodes
  - e: number of edges
- DV: $O(d * n * k)$ messages
  - d: node's degree
  - k: number of rounds

Time complexity

- LS: $O(n * \log n)$
- DV: $O(n)$

Convergence time

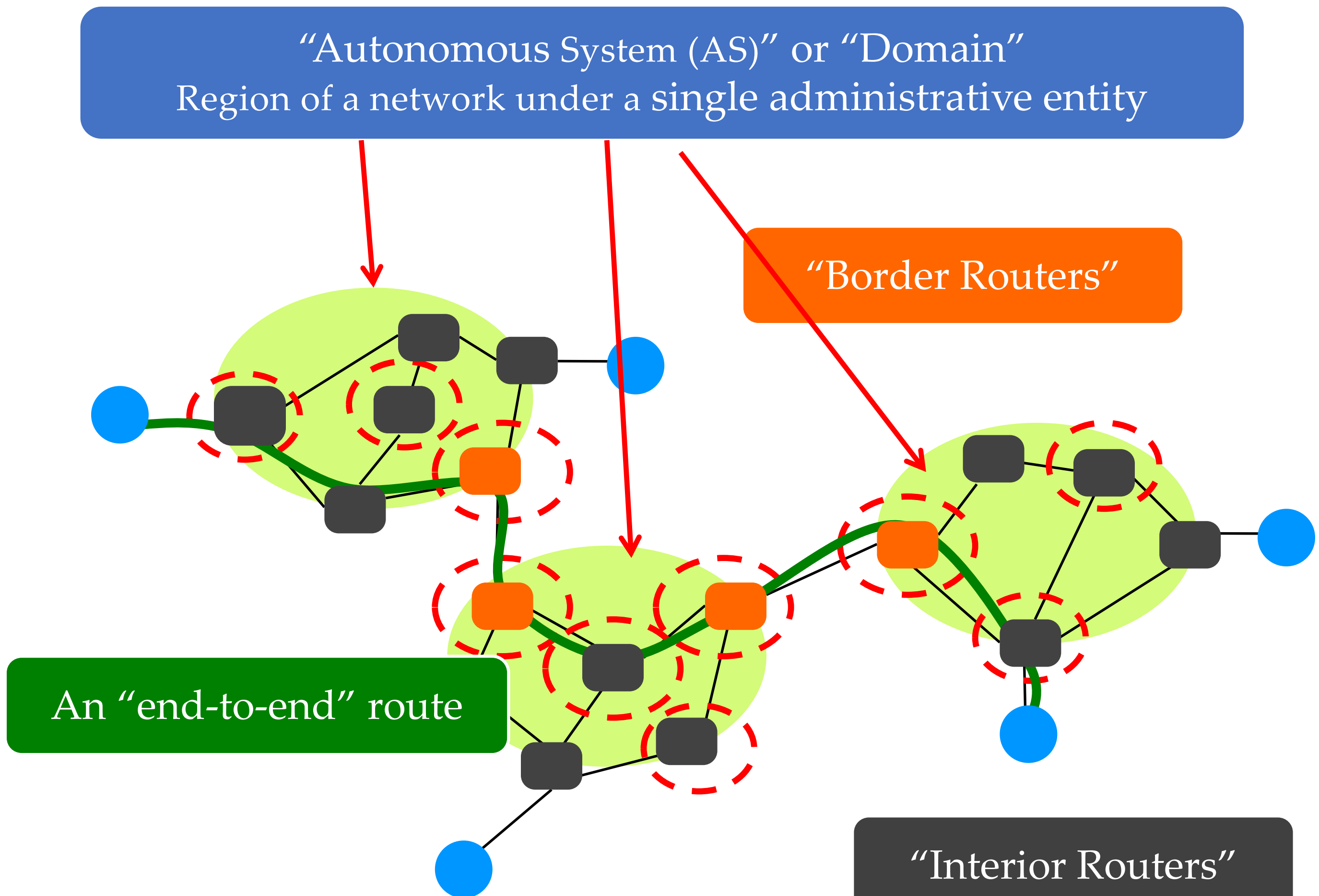
- LS: $O(1)$
- DV: $O(k)$

- Robustness: what happens if router malfunctions?
- LS:
  - node can advertise incorrect link cost
  - each node computes only its own table
- DV:
  - node can advertise incorrect path cost
  - each node's table used by others; error propagate through network

So who wins?
What matters the most?

Switching Gears…

- Distance vectors vs. Link State
  - A "local" routing problem
  - How do you route from A to B inside the same network (autonomous system)
- But what about routing across *Internet*, i.e. between networks?
  - Let's talk about how to assign addresses and route in the wide area!


"Autonomous System (AS)" or "Domain"
Region of a network under a single administrative entity
"Border Routers"
An "end-to-end" route
"Interior Routers"

Autonomous Systems (AS)

- AS is a network under a single administrative control
  - currently over 30,000 ASes
  - Think AT&T, France Telecom, Verizon, Level 3, etc.
- ASes are sometimes called "domains"
- Each AS is assigned a unique identifier
  - 16 bit AS Number (ASN)

"Intradomain" routing: within an AS

- Link-State (OSPF) and Distance-Vector (RIP, IGRP)
- Focus
  - "least cost" paths
  - convergence

“Interdomain” routing: between ASes

- Two key challenges
  - Scaling

  - Administrative structure
    - Issues of autonomy, policy, privacy

"Interdomain" routing: between ASes

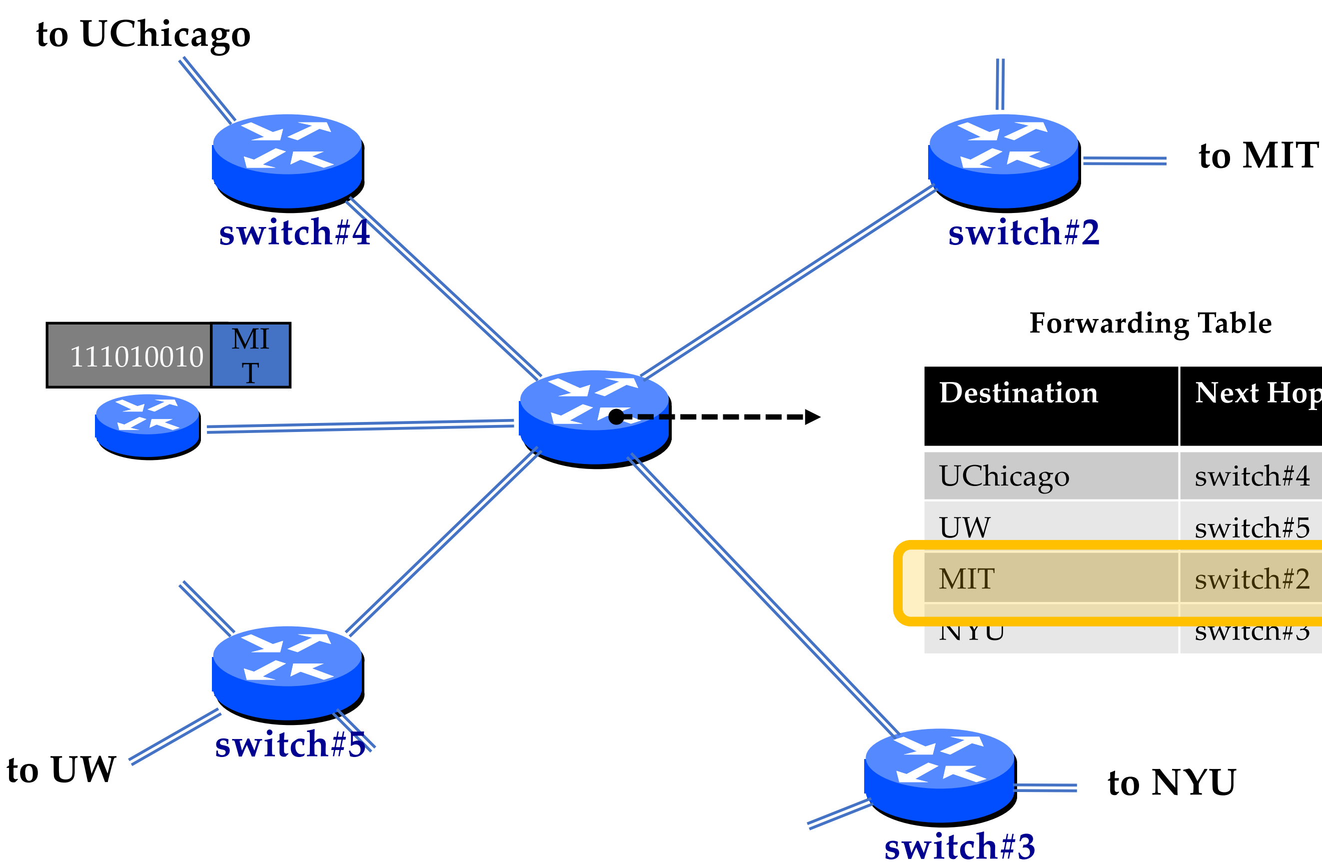
- Two key challenges
  - Scaling

  - Administrative structure
    - Issues of autonomy, policy, privacy

Recall From Last Lecture

- Assume each host has a unique ID
- No particular structure to those IDs

Recall Also…



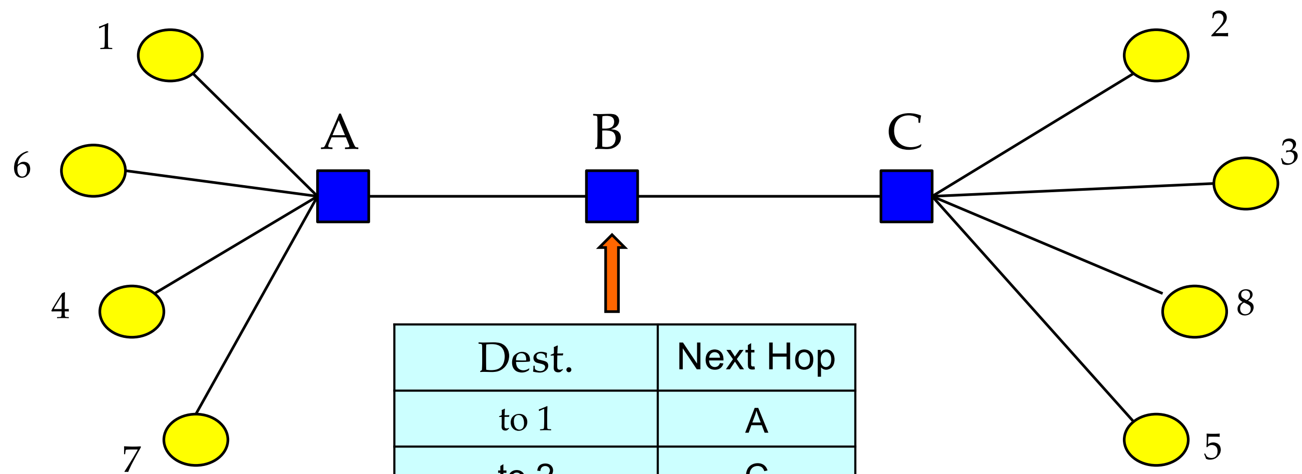
Forwarding Table

Destination	Next Hop
UChicago	switch#4
UW	switch#5
MIT	switch#2
NYU	switch#3

Scaling

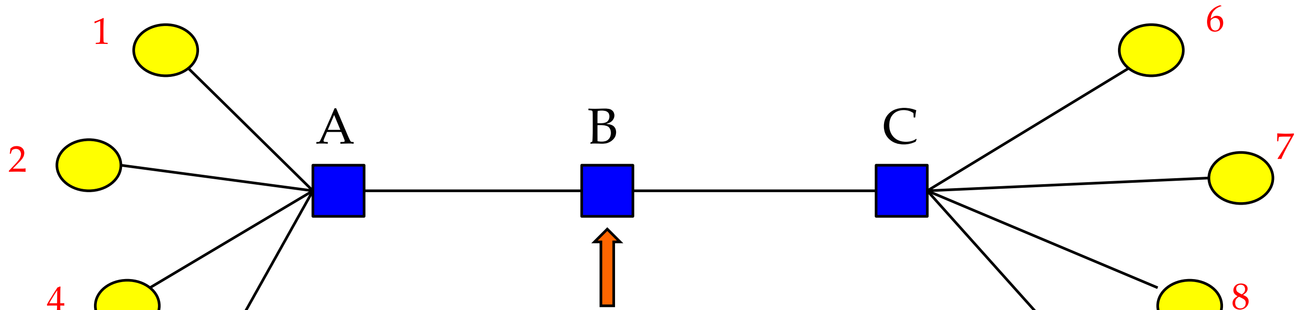
- A router must be able to reach any destination
  - Given packet's destination address, lookup "next hop"
- Naive: Have an entry for each destination
  - There would be over 2^32 entries!
  - And routing updates per destination!
- Any ideas on how to improve scalability?

A smaller table at node B?



Dest.	Next Hop
to 1	A
to 2	C
to 3	C
to 4	A
to 5	C
to 6	A
to 7	A
to 8	C

Re-number the end-systems?



Dest.	Next Hop
to [1-4]	A
to [5-8]	C

- careful address assignment → can *aggregate* multiple addresses into one range → scalability!
- akin to reducing the number of destinations

Scaling

- A router must be able to reach *any* destination
- Naive: Have an entry for each destination
- Better: Have an entry for a range of addresses
  - But can’t do this if addresses are assigned randomly!
- How addresses are allocated will matter!!

Host a**ddressing is key to scaling**

Two Key Challenges

- Scaling
- Administrative structure
  - Issues of autonomy, policy, privacy

Administrative structure shapes Interdomain routing

- ASes want freedom to pick routes based on policy
  - "My traffic can't be carried over my competitor's network"
  - "I don't want to carry A's traffic through my network"
  - Not expressible as Internet-wide "least cost"!

- ASes want autonomy
  - Want to choose their own internal routing protocol
  - Want to choose their own policy

- ASes want privacy
  - choice of network topology, routing policies, etc.

Choice of Routing Algorithm

Link State (LS) *vs.* Distance Vector (DV)?

- LS offers no privacy – broadcasts all network information
- LS limits autonomy -- need agreement on metric, algorithm

- DV is a decent starting point
  - Per-destination updates by intermediate nodes give us a hook
  - but wasn't designed to implement policy
  - and is vulnerable to loops if shortest paths not taken

The "Border Gateway Protocol" (BGP) extends distance-vector ideas to accommodate policy

Today

- Addressing

- BGP
  - context and basic ideas: today
  - details and issues: next lecture

Addressing Goal: Scalable Routing

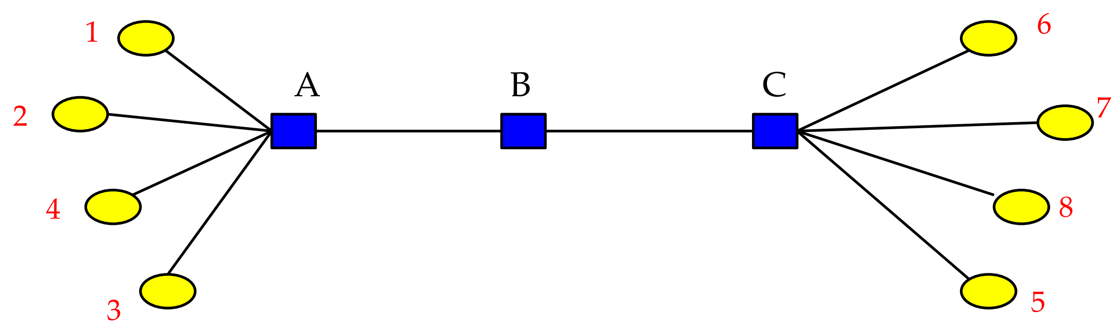
- State: Small forwarding tables at routers
  - Much less than the number of hosts

- Churn: Limited rate of change in routing tables

Ability to aggregate addresses is crucial for both

(one entry to *summarize* many addresses)

Aggregation only works if….



- Groups of destinations reached via the same path
- These groups are assigned contiguous addresses
- These groups are relatively stable
- Few enough groups to make forwarding easy

Hence, IP Addressing: Hierarchical

- Hierarchical address structure
- Hierarchical address allocation
- Hierarchical addresses and routing scalability

IP Addresses (IPv4)

- Unique 32-bit number associated with a host

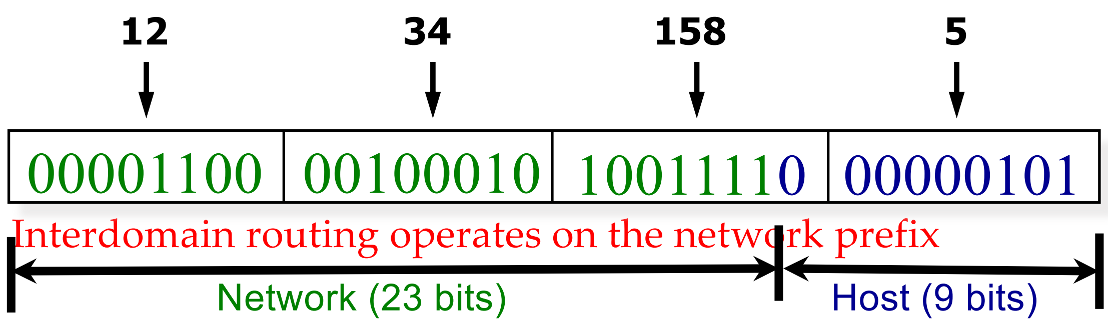
  00001100 00100010 10011110 00000101

- Represented with the “dotted quad” notation
  - e.g., 12.34.158.5

12	**34**	**158**	**5**
00001100	00100010	10011110	00000101

Hierarchy in IP Addressing

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



History of Internet Addressing

- Always dotted-quad notation
- Always network/host address split
- But nature of that split has changed over time

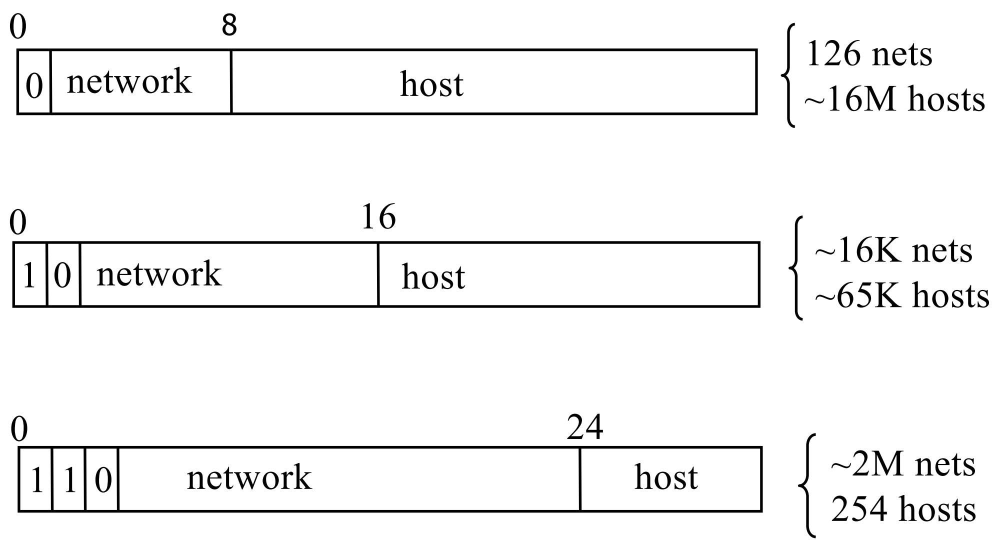
Original Internet Addresses

- First eight bits: network component
- Last 24 bits: host component

- Assumed 256 networks were more than enough!

Next 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
- Motivation: offer a better tradeoff between size of the routing table and efficient use of the IP address space

CIDR (example)

- Suppose a network has fifty 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"

Classful vs. Classless addresses

- Example: an organization needs 500 addresses.
  - A single class C address not enough (254 hosts).
  - Instead a class B address is allocated. (~65K hosts)
  - That's overkill, a huge waste!

- CIDR allows an arbitrary prefix-suffix boundary
  - Hence, organization allocated a single /23 address (equivalent of 2 class C's)

- Maximum waste: 50%

Hence, IP Addressing: Hierarchical

- Hierarchical address structure
- Hierarchical address allocation
- Hierarchical addresses and routing scalability

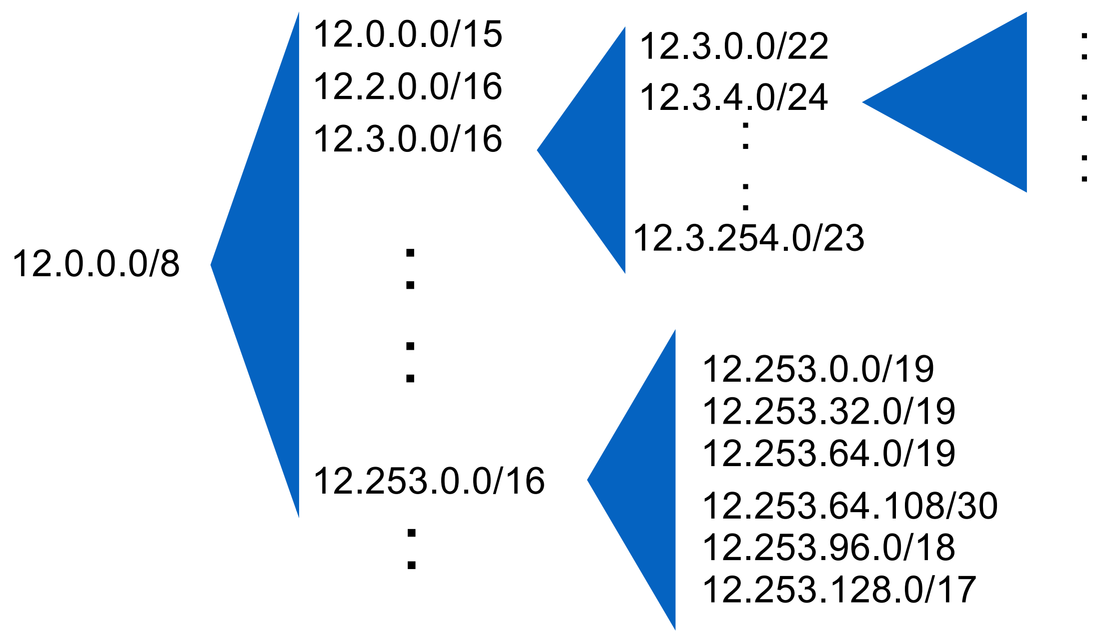
Allocation Done Hierarchically

- Internet Corporation for Assigned Names and Numbers (ICANN) gives large blocks to…
- Regional Internet Registries, such as the American Registry for Internet Names (ARIN), which give blocks to…
- Large institutions (ISPs), which give addresses to…
- Individuals and smaller institutions
- FAKE Example: UChicago actually triple homed

ICANN → ARIN → Qwest → UChicago → CS

CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as get closer to host



FAKE 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

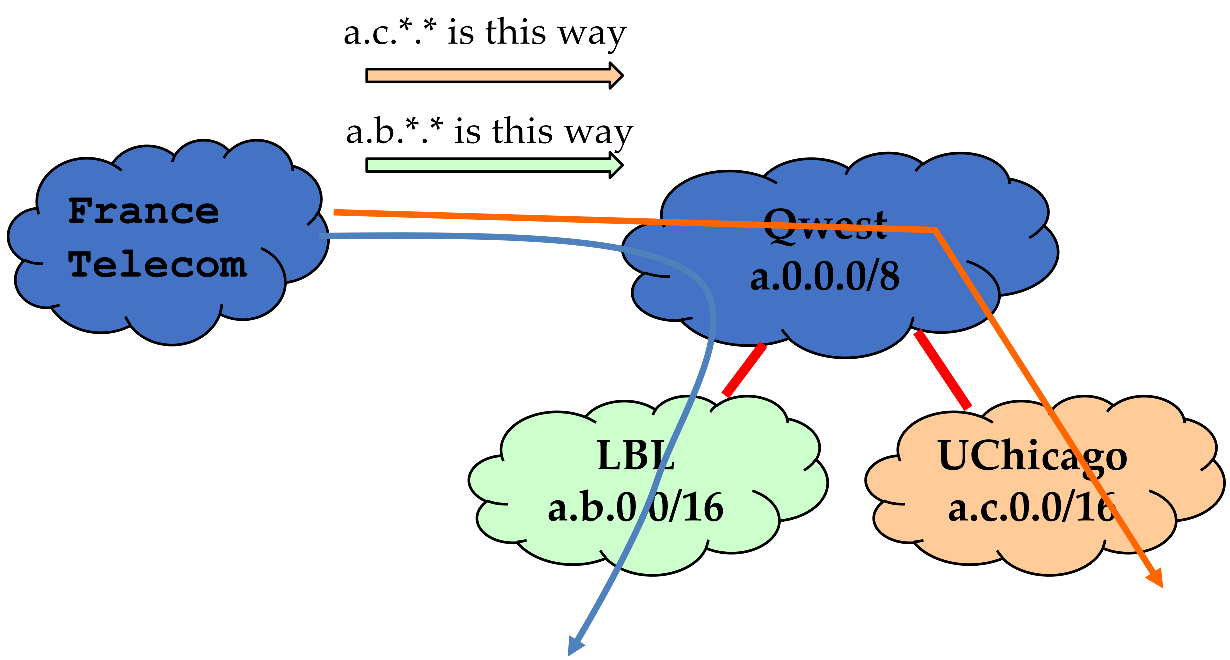
Hence, IP Addressing: Hierarchical

- Hierarchical address structure
- Hierarchical address allocation
- Hierarchical addresses and routing scalability

IP addressing → scalable routing?

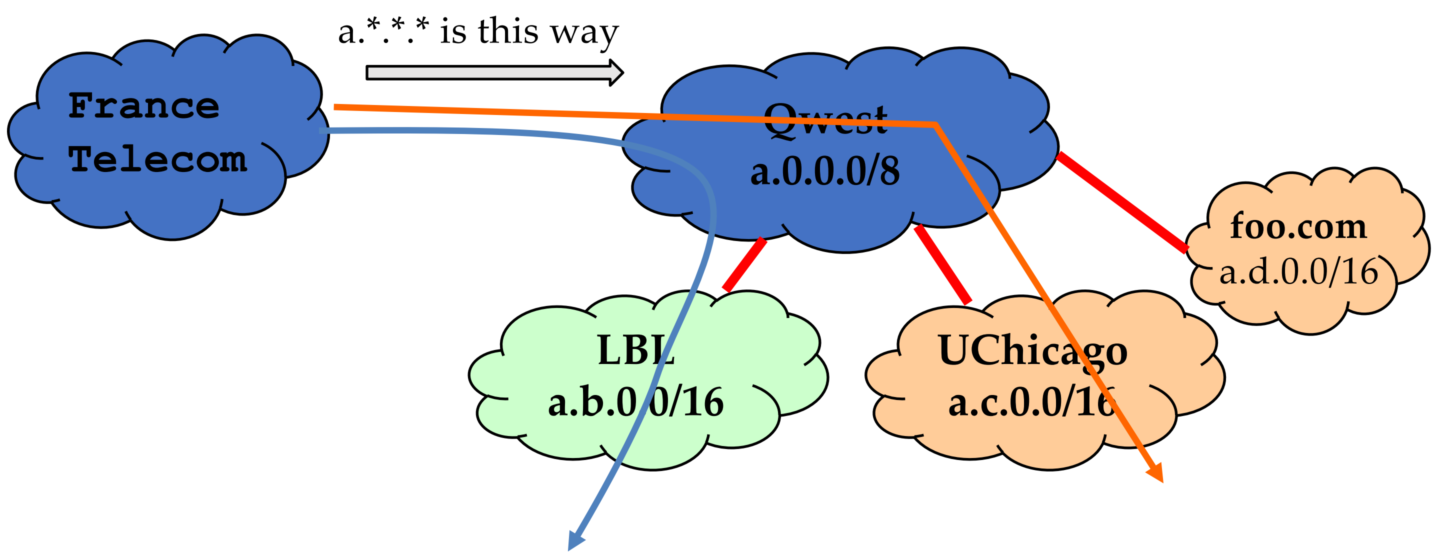
Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy

IP addressing → scalable routing?



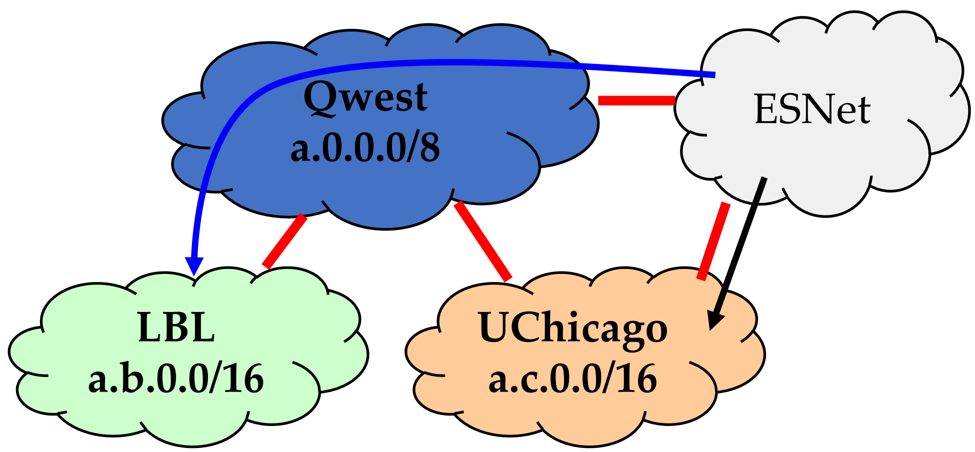
IP addressing → scalable routing?

Can add new hosts/networks without updating the routing entries at France Telecom



IP addressing → scalable routing?

ESNet must maintain routing entries for both a.*.*.* and a.c.*.*



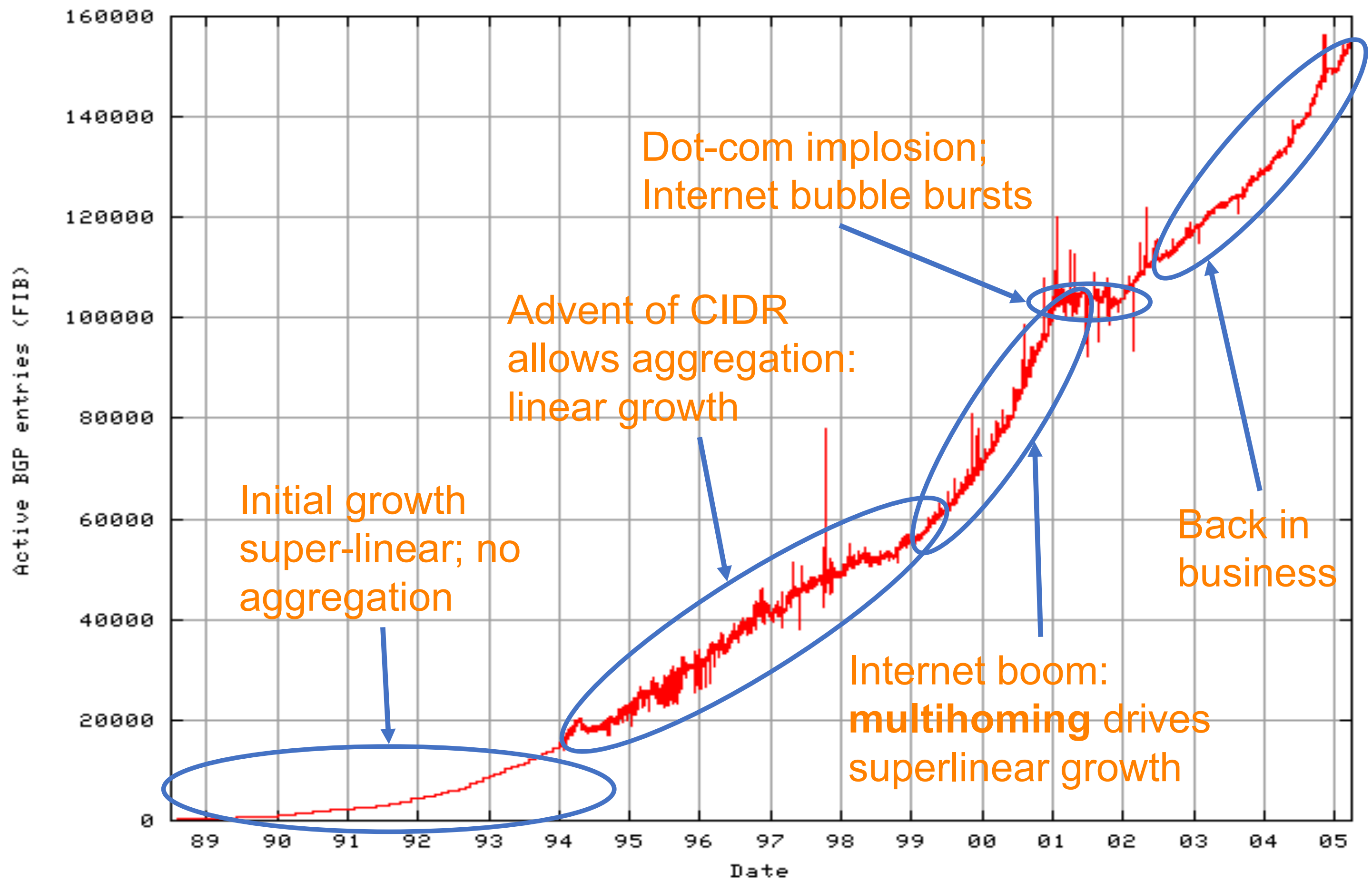
IP addressing → scalable routing?

- Hierarchical address allocation helps routing scalability if allocation matches topological hierarchy

- Problem: may not be able to aggregate addresses for "multi-homed" networks

- Two competing forces in scalable routing
  - aggregation reduces number of routing entries
  - multi-homing increases number of entries

Growth in Routed Prefixes (1989-2005)



Summary of Addressing

- Hierarchical addressing
  - Critical for scalable system
  - Don't require everyone to know everyone else
  - Reduces amount of updating when something changes
- Non-uniform hierarchy
  - Useful for heterogeneous networks of different sizes
  - Class-based addressing was far too coarse
  - Classless InterDomain Routing (CIDR) more flexible
- A later lecture: impact of CIDR on router designs

Outline

- Addressing

- Border Gateway Protocol (BGP)
  - today: context and key ideas
  - next lecture: details and issues

BGP (Today)

- The role of policy
  - what we mean by it
  - why we need it

- Overall approach
  - four non-trivial changes to DV
  - how policy is implemented (detail-free version)

Administrative structure shapes Interdomain routing

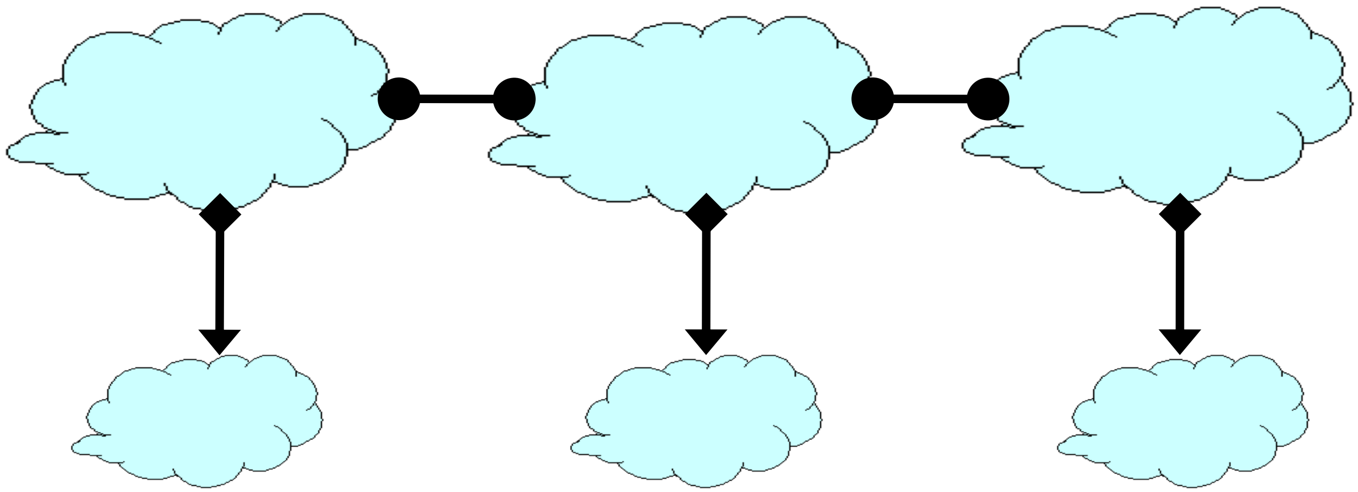
- ASes want freedom to pick routes based on policy
- ASes want autonomy
- ASes want privacy

Topology and policy is shaped by the business relationships between ASes

- Three basic kinds of relationships between ASes
  - AS A can be AS B's *customer*
  - AS A can be AS B's *provider*
  - AS A can be AS B's *peer*

- Business implications
  - Customer pays provider
  - Peers don't pay each other
    - Exchange roughly equal traffic

Business Relationships



Relations between ASes

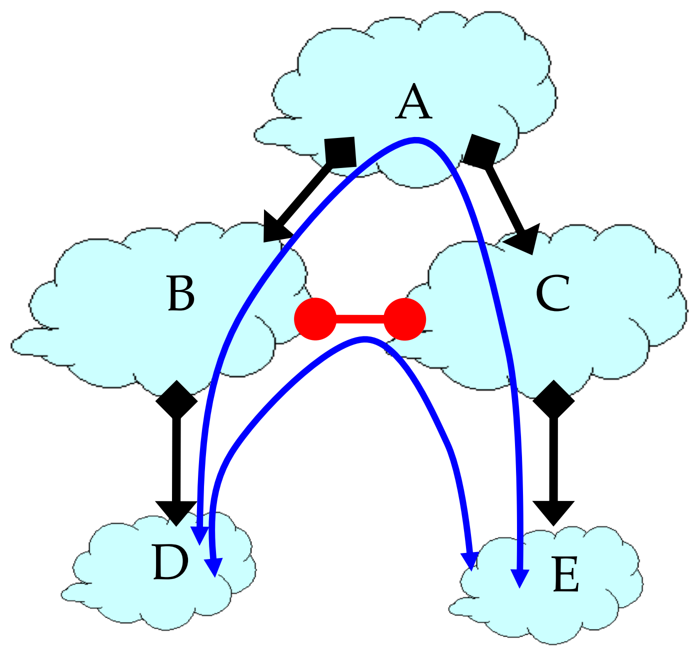
provider ◆——▶ customer

peer ●——● peer

Business Implications

- Customers pay provider
- Peers don't pay each other

Why peer?



E.g., D and E talk a lot

Peering saves B *and* C money

Relations between ASes

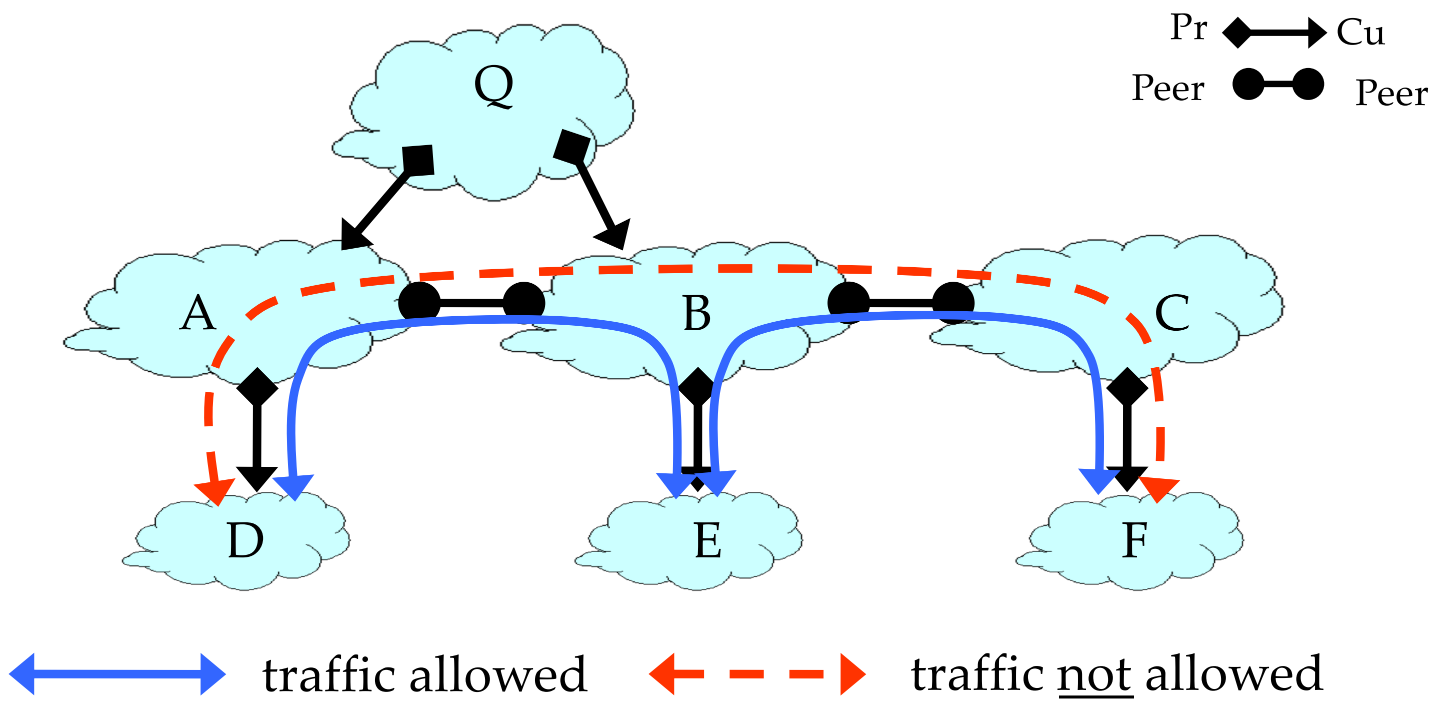
provider ◆——▶ customer

peer ●——● peer

Business Implications

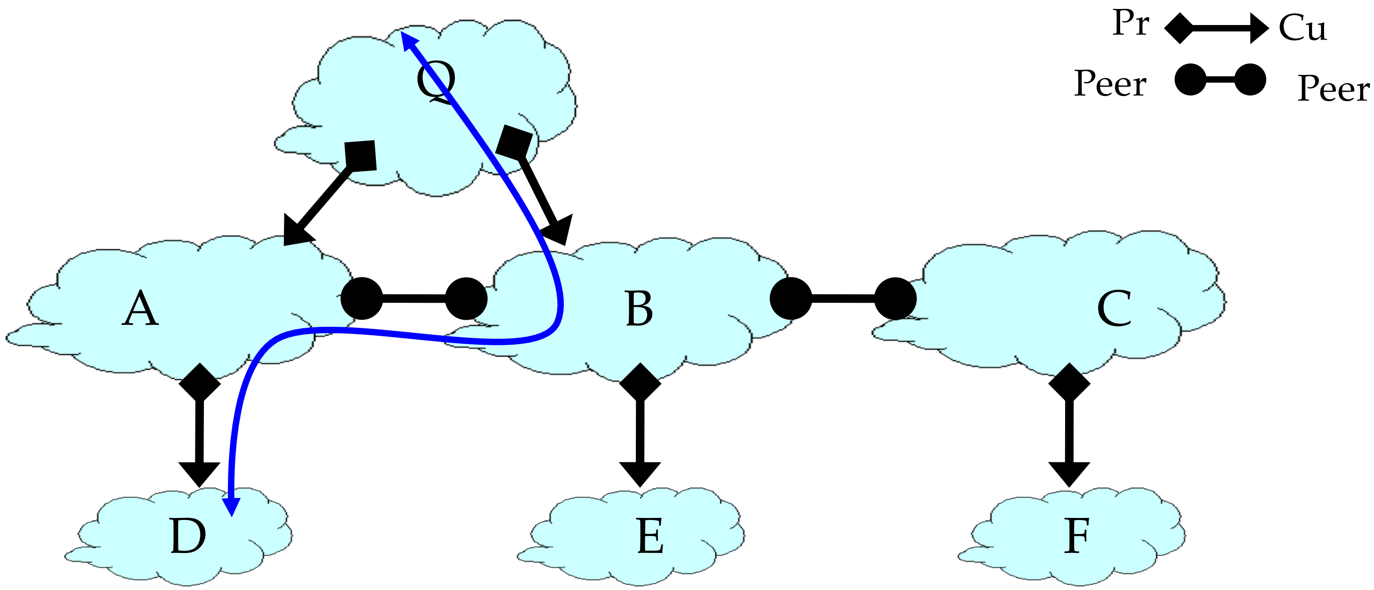
- Customers pay provider
- Peers don't pay each other

Routing Follows the Money!



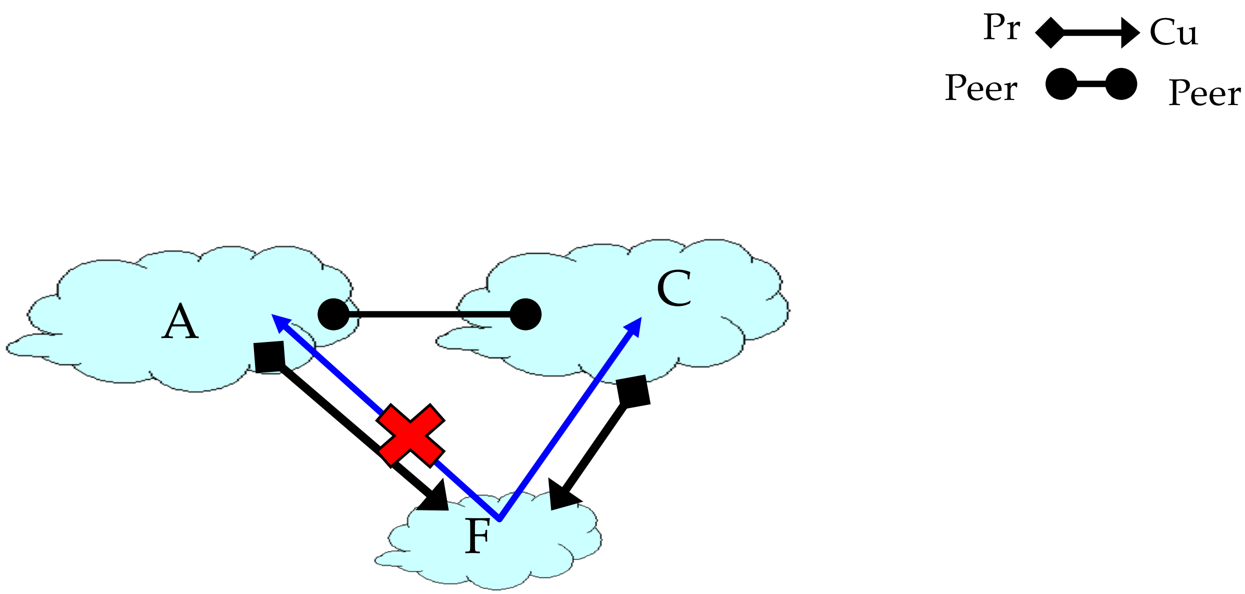
- ASes provide “transit” between their customers
- Peers do not provide transit between other peers

Routing Follows the Money!



- An AS only carries traffic to/from its own customers over a peering link

Routing Follows the Money!



- Routes are “valley free” (will return to this later)

In Short

- AS topology reflects business relationships between Ases
- Business relationships between ASes impact which routes are acceptable
- BGP Policy: Protocol design that allows ASes to control which routes are used
- Next lecture: more formal analysis of the impact of policy on reachability and route stability