Routing

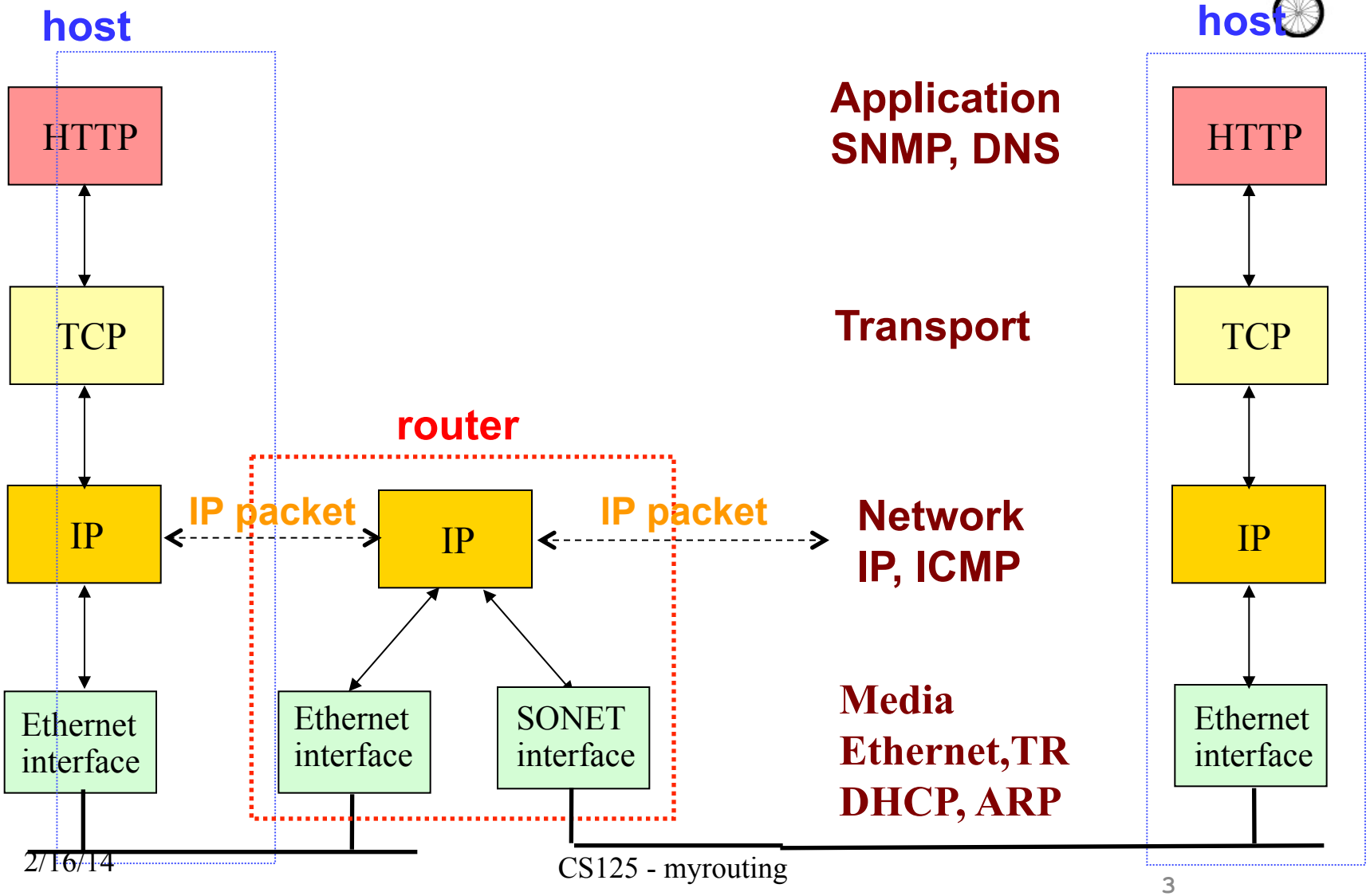
Reading: Chapter 3

Overview: Books on Routing, BGP is a life/career

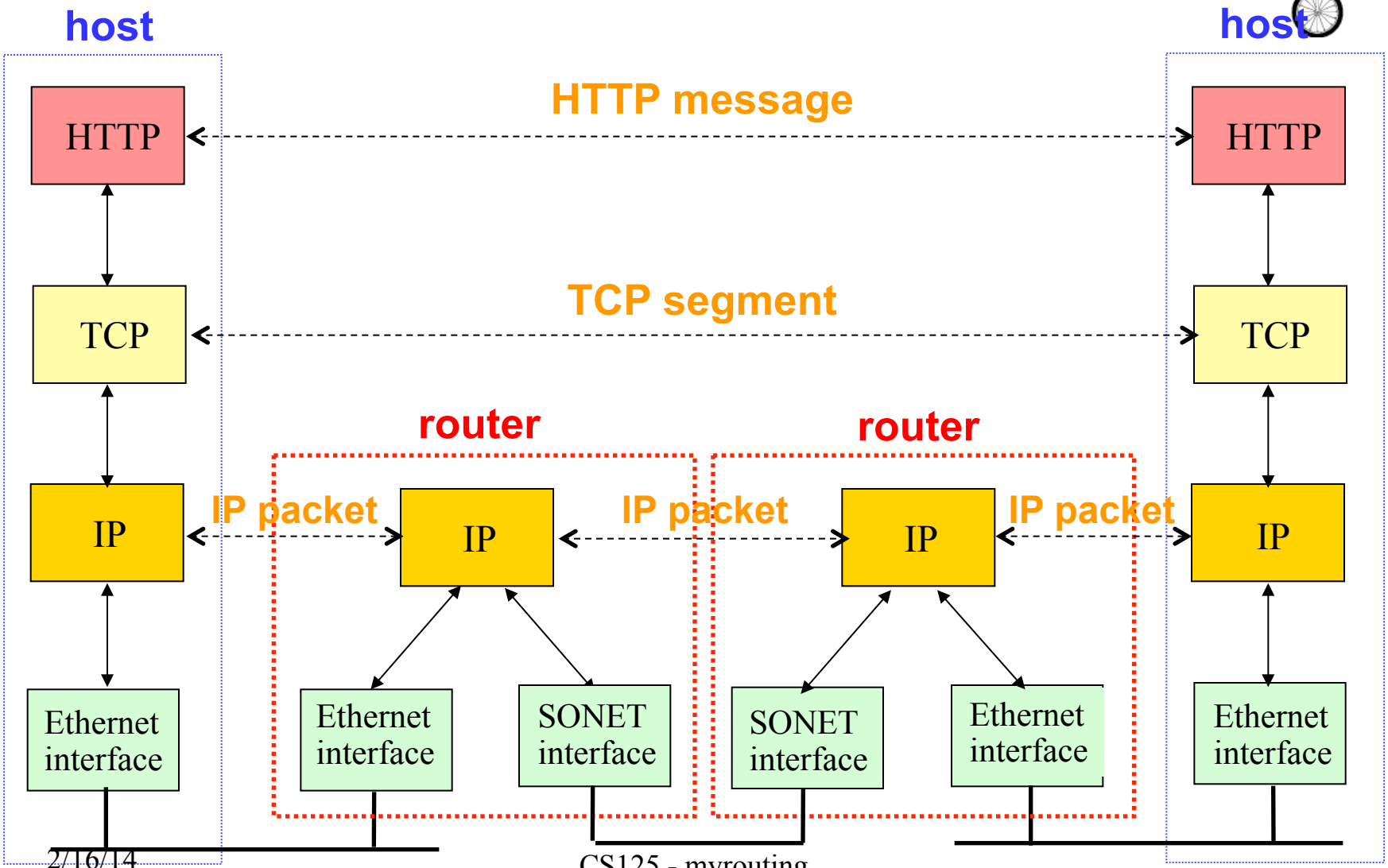
Goals of Lecture

- Path selection
  - Minimum-hop and shortest-path routing
- Routing protocols
  - Link state: OSPF, Open Shortest Path First
  - Distance vector: RIP, Routing Information Protocol
- Metrics

IP Suite In Action: Where We Are!!



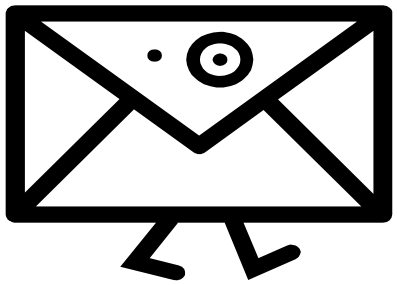
IP Suite In Action: End Hosts vs. Routers



What is Routing?

- A famous quotation from RFC 791

  "A *name* indicates what we seek.
  An *address* indicates where it is.
  A *route* indicates how we get there."
  -- Jon Postel



IP Routing

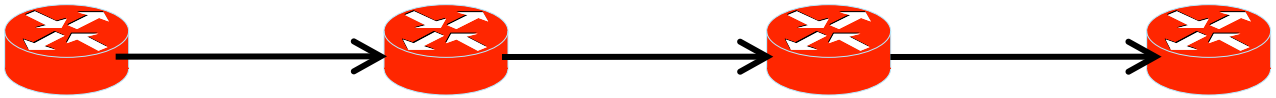
- Routing - operational aspects of Internet (NANOG)
- Routing - process of choosing/creating a path
- Router - network node making the choice on path
- IP Routing Factors
  - Network load
  - Datagram length
  - Requested service
  - Physical network characteristics
- Truth - shortest paths usually based on hop counts???
  - Net neutrality

IP Routing, cont

- Direct Routing
  - Host to host attached to same physical network/media
- Indirect Routing
  - Source/Destination not on same physical network, Router needed
  - Sender must know Router/Gateway IP, then MAC address
  - Default Gateway, always available or screwed (DHCP)

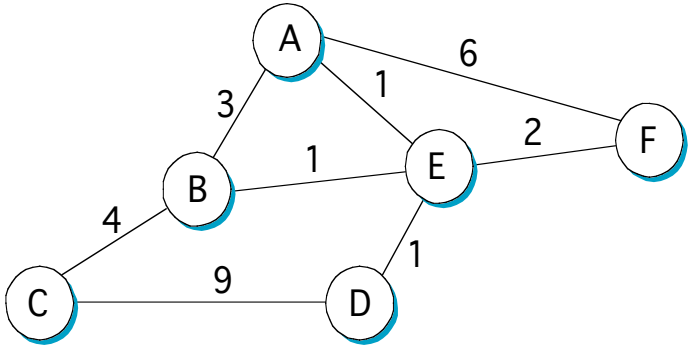
Forwarding vs. Routing

- Forwarding: data plane –moving packet
  - Directing a data packet to an outgoing link
  - Individual router *using* a forwarding table
- Routing: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Individual router *creating* a forwarding table



Overview

- Network as a Graph



- Problem: Find lowest cost path between two nodes
- Factors
  - Static: topology
  - Dynamic: load
  - Static: link speed
  - Dynamic/Static: Weights and costs

Routing Worlds

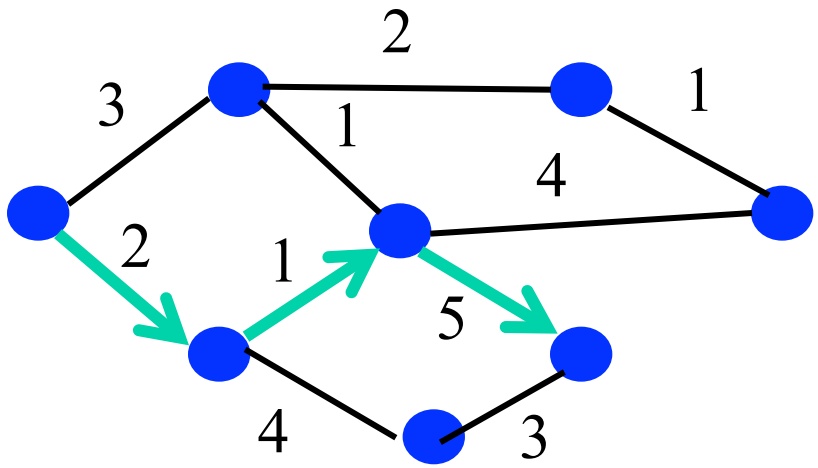
- IGP
  - Interior Gateway protocols – AS, Autonomous System
  - Intradomain Routing protocols
  - Need to define ‘Domain’, e.g., Claremont Colleges
- EGP
  - Exterior Gateway protocols
  - Interdomain Routing protocols – Between ASs

Why Does Routing Matter?

- End-to-end performance
  - Quality of the path affects user performance
  - Propagation delay, throughput, and packet loss
- Use of network resources
  - Balance of the traffic over the routers and links
  - Avoiding congestion by directing traffic to lightly-loaded links – always an issue
- Transient disruptions during changes
  - Failures, maintenance, and load balancing
  - Limiting packet loss and delay during changes

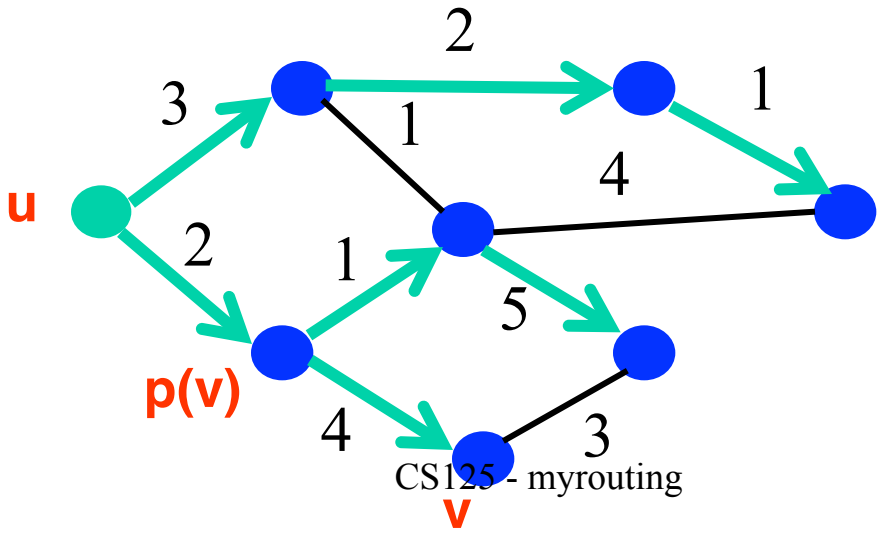
Shortest-Path Routing

- Path-selection model
  - Destination-based
  - Load-insensitive (e.g., static link weights)
  - Minimum hop count or sum of link weights



Shortest-Path Problem

- Given: network topology with link costs
  - **c(x,y)**: i.e., link cost from node x to node y
  - Infinity if x and y are not direct neighbors
- Compute: least-cost paths to all nodes
  - From a given source U to all other nodes
  - **p(v)**: predecessor node along path from source to V



Dijkstra's Shortest-Path Algorithm

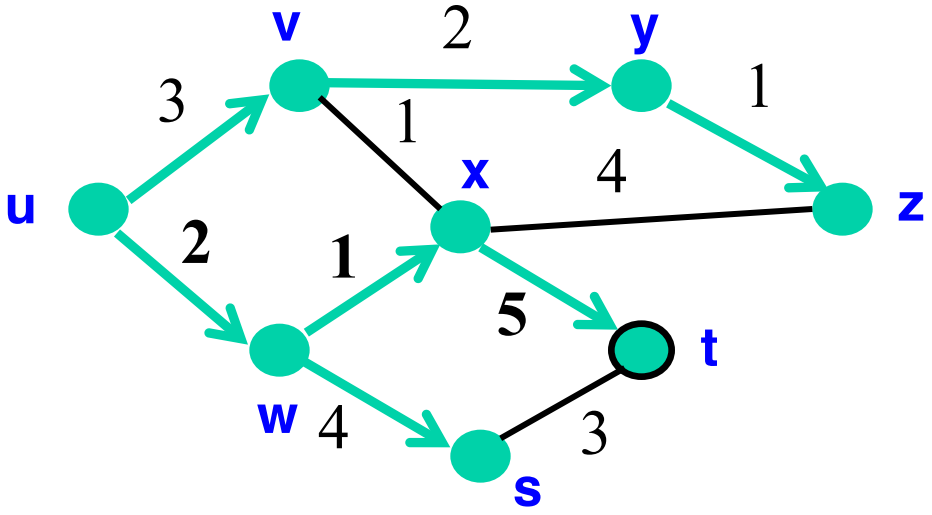
- Iterative algorithm
  - After k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
  - Initially, **S = {u}** where u is the source node
  - Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
  - Initially, **D(v) = c(u,v)** for all nodes v adjacent to u
  - … and **D(v)** = ∞ for all other nodes v
  - Continually update D(v) as shorter paths are learned
- How to learn paths?

Dijsktra's Algorithm


```
1  Initialization:
2    S = {u}
3    for all nodes v
4      if v adjacent to u {
5          D(v) = c(u,v)
6      else D(v) = ∞
7
8    Loop
9      find w not in S with the smallest D(w)
10     add w to S
11     update D(v) for all v adjacent to w and not in S:
12        D(v) = min{D(v), D(w) + c(w,v)}
13   until all nodes in S
```

Shortest-Path Tree for Node u Built by Dijkstra's Alg.

- Shortest-path tree from u
- Forwarding table at u



	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Telling 'u' the next hop to reach any other node

Link-State Routing

- Each router keeps track of its incident links
  - Whether the link is up or down
  - The cost on the link
- Each router broadcasts the link state
  - To give every router a complete view of the graph
- Each router runs Dijkstra's algorithm
  - To compute the shortest paths
  - … and constructs the forwarding table
- Example protocols
  - Open Shortest Path First (OSPF)
  - Intermediate System – Intermediate System (IS-IS)

Link State

- Strategy
  - send to all nodes (not just neighbors) information about directly connected links (not entire routing table)
  - Routers can then build their own map
- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet

Link State (cont)

- Reliable flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new (my) LSP periodically
    - increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP
    - discard when TTL=0, aging

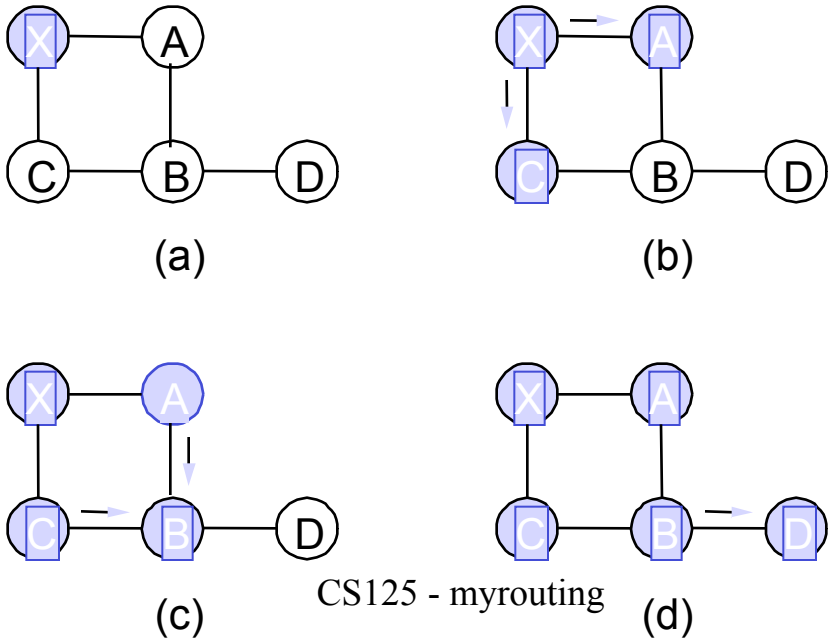
Detecting Topology Changes

- Beaconing
  - Periodic "hello" messages in both directions
  - Detect a failure after a few missed "hellos", usually k of n

- Performance trade-offs
  - Detection speed
  - Overhead on link bandwidth and CPU
  - Likelihood of false detection

Broadcasting the Link State

- Flooding
  - Node sends link-state information out its links
  - And then the next node sends out all of its links
  - … except the one where the information arrived
  - note race and time stamp needed at node B



Broadcasting the Link State

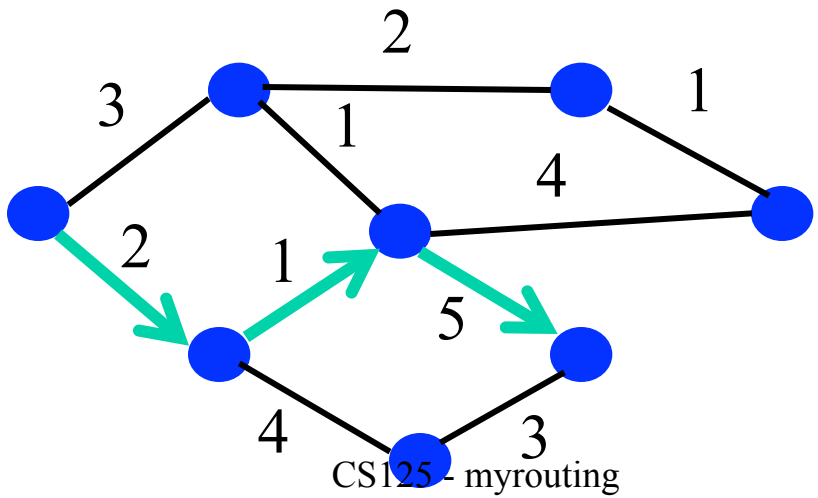
- Reliable flooding
  - Ensure all nodes receive link-state information
  - … and that they use the latest version
- Challenges
  - Packet loss
  - Out-of-order arrival
- Solutions
  - Acknowledgments and retransmissions (in routing protocol)
  - Sequence numbers
  - Time-to-live for each packet

When to Initiate Flooding?

- Topology change
  - Link or node failure
  - Link or node recovery
- Configuration change
  - Link cost change
- Periodically
  - Refresh the link-state information
  - Typically (say) 30 minutes
  - Corrects for possible corruption of the data

Convergence

- Getting consistent routing information to all nodes
  - E.g., all nodes having the same link-state database
- Consistent forwarding after convergence
  - All nodes have the same link-state database
  - All nodes forward packets on shortest paths
  - The next router on the path forwards to the next hop



Convergence Delay

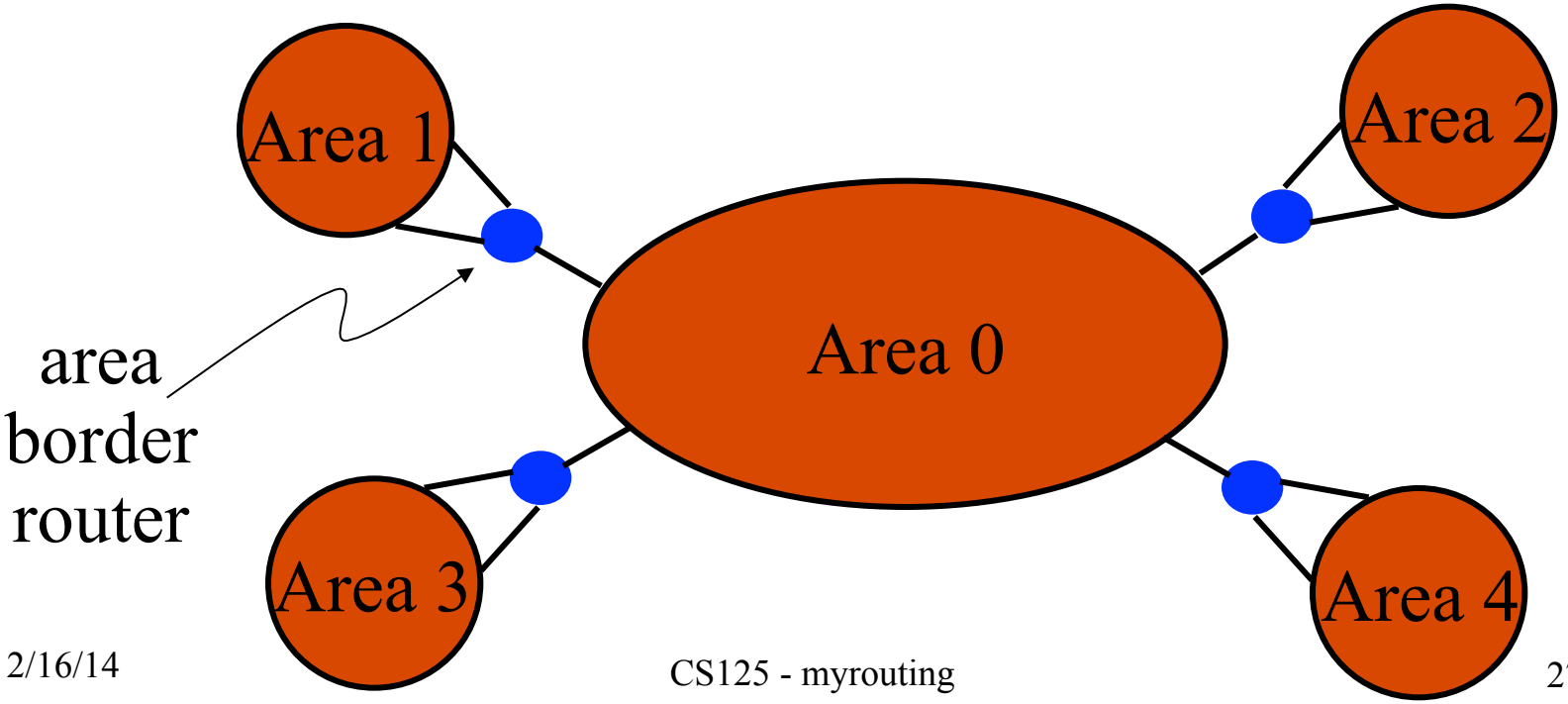
- Sources of convergence delay
  - Detection latency of failed links
    - Forwarding incorrect info
    - Inconsistent link-state database
  - Flooding of link-state information
  - Shortest-path computation
  - Creating the forwarding table
- Performance during convergence period
  - Lost packets due to black holes and TTL expiring
  - Looping packets consuming resources
  - Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video

Reducing Convergence Delay

- Faster detection
  - Smaller hello timers
  - Link-layer technologies that can detect failures
- Faster flooding
  - Flooding immediately
  - Sending link-state packets with high-priority
- Faster computation
  - Faster processors on the routers
  - Incremental Dijkstra algorithm (can rebuild for part of graph
- Faster forwarding-table update
  - Data structures supporting incremental updates

Scaling Link-State Routing

- Overhead of link-state routing
  - Flooding link-state packets throughout the network
  - Running Dijkstra's shortest-path algorithm on each router
- Introduce hierarchy through "areas"



Another Idea to Routing Distance Vector

- Each node maintains a set of triples
  - **`(Destination, Cost, NextHop)`**
- Directly connected neighbors exchange updates
  - periodically (on the order of several seconds)
  - whenever table changes (called *triggered* update)
- Each update is a list of pairs:
  - **`(Destination, Cost)`**
- Update local table if receive a "better" route
  - smaller cost
  - came from next-hop
- Refresh existing routes; delete if they time out
- Depend on neighbor for correct info
- Update = what a neighbor knows

Distance Vector Algorithm

- c(x,v) = cost for direct link from x to v
  - Node x maintains costs of direct links c(x,v)
- $D_x(y)$ = estimate of least cost from x to y
  - Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$
- Each node v periodically sends D_v to its neighbors
  - And neighbors update their own distance vectors
  - $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ for each node $y \in N$
- Over time, the distance vector D_x converges

Distance Vector Algorithm

Iterative, asynchronous:

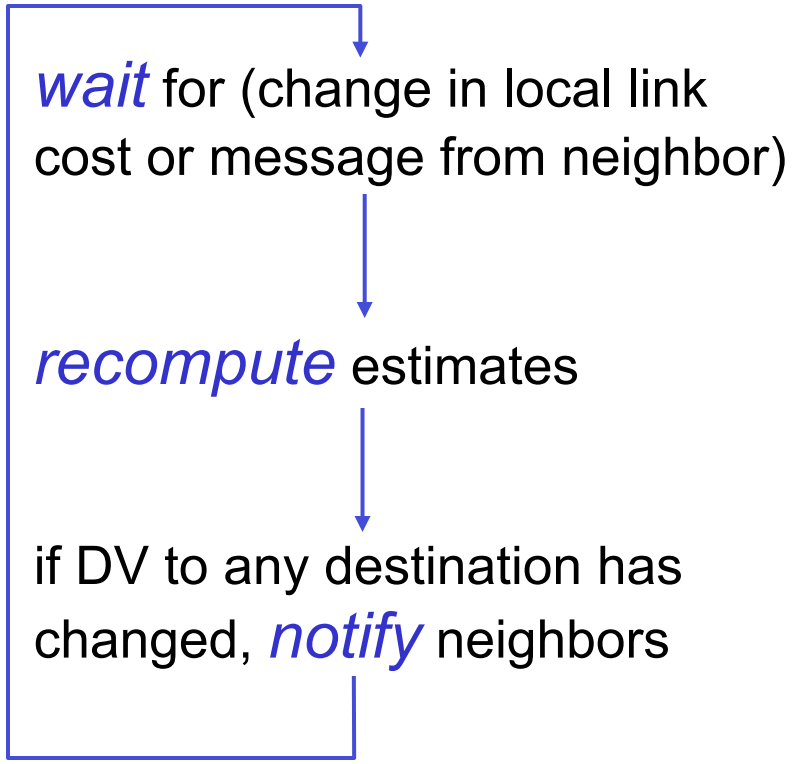
each local iteration caused by:

- Local link cost change
- Distance vector update message from neighbor

Distributed:

- Each node notifies neighbors *only* when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:



Distance Vector Example: Step 0

Optimum 1-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

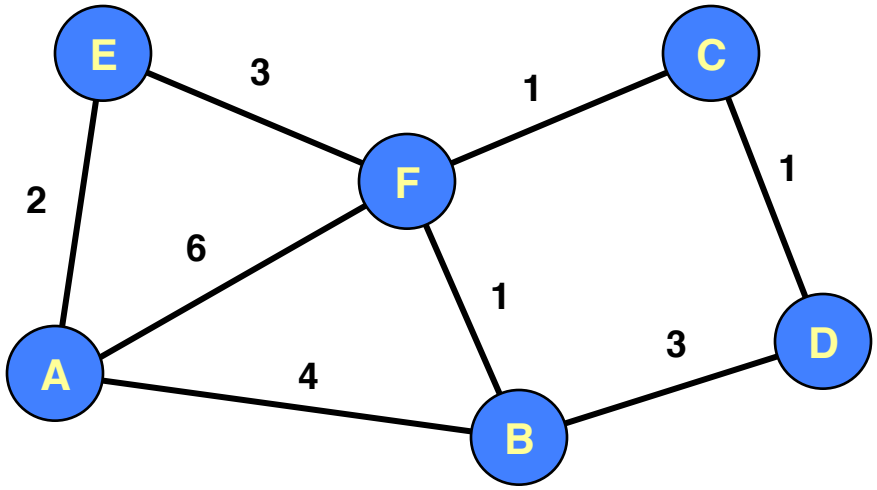


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

Distance Vector Example: Step 2

Optimum 2-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	7	F	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

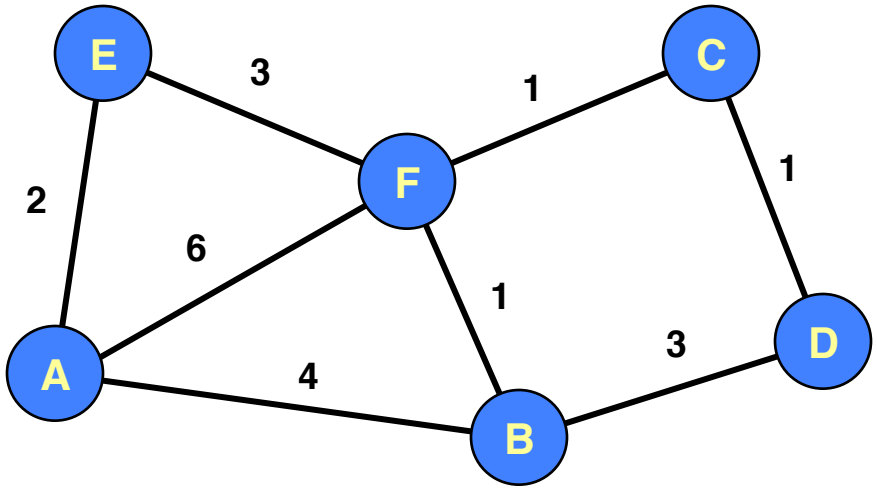


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	–	D	2	C
E	4	F	E	∞	–	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

Distance Vector Example: Step 3

Optimum 3-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

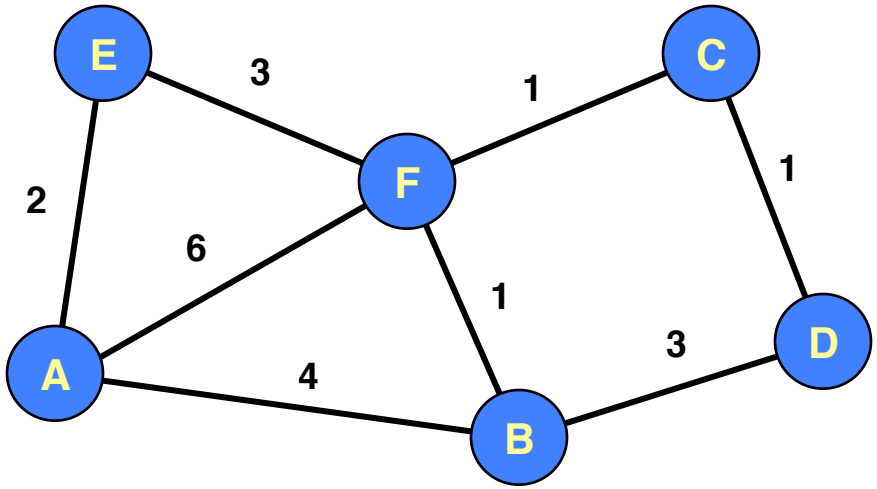


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

Distance Vector Failures: Set cost to infinity: &

Optimum 3-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

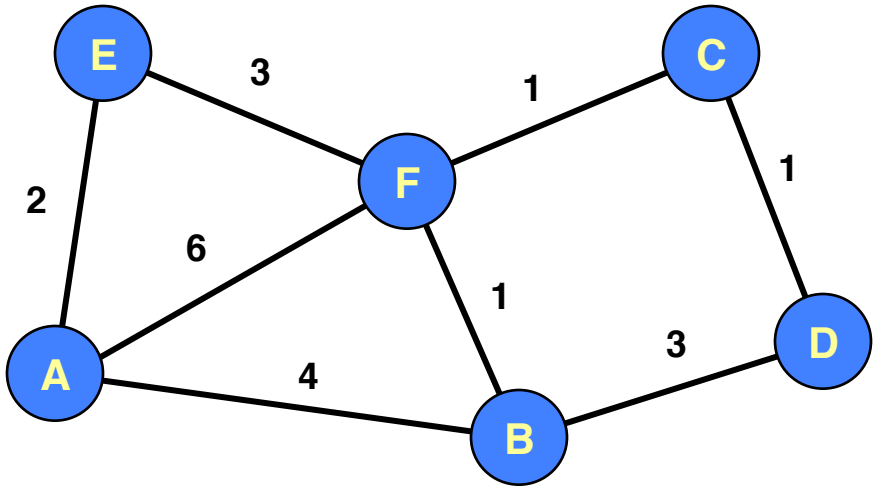


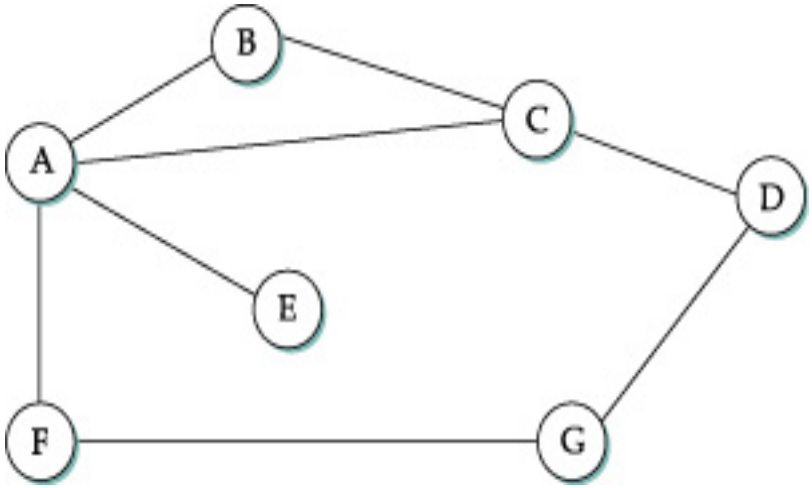
Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	&	
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

What about D's table

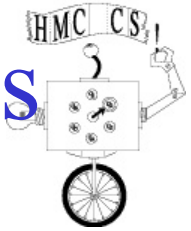
Distance Vector: Link Cost Changes, an Issue!!

Link cost changes:

- Node detects local link cost change
- Updates the distance table
- If cost change in least cost path, notify neighbors

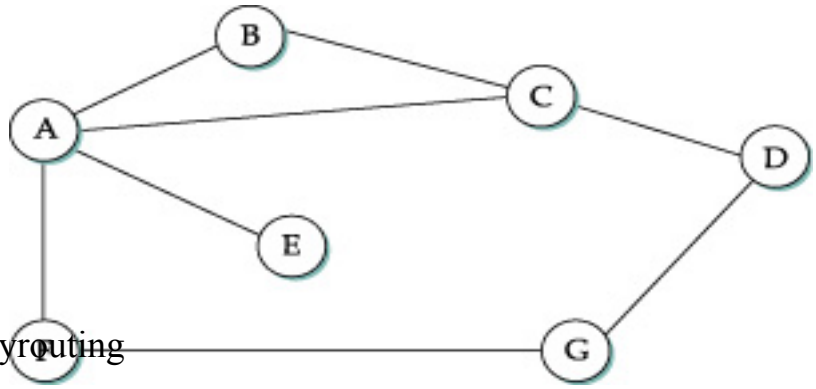


Distance Vector: Link Cost Changes



Count to Infinity:

- Link from A to E goes down
- Next advertisements:
  - A advertises infinity to E, but B and C advertise a distance of 2 to E
- Depending on timing:
  - B hearing that E can be reached in 2 hops via C, concludes it (B) can reach E in 3 hops, hearing this A concludes it can get to E in 4 hops via B, hearing this C concludes it can get to E in 5 hops, etc.



Loop-Breaking Heuristics

- Set infinity to 16
- Split horizon - Don't send routes back to source of update - need to keep that info in the table
- Split horizon with poison reverse - send routes back to source, but set distance to infinity

Example Routing Protocol
Routing Information Protocol (RIP)

- Distance vector protocol
  - Nodes send distance vectors every 30 seconds
  - … or, when an update causes a change in routing
- Link costs in RIP
  - All links have cost 1
  - Valid distances of 1 through 15
  - … with 16 representing infinity
  - Small "infinity" → smaller "counting to infinity" problem
- RIP is limited to fairly small networks
  - E.g., used in the Princeton campus network

```
void
mergeRoute (Route *new)
{
   int i;

   for (i = 0; i < numRoutes; ++i)
   {
      if (new->Dest == rt[i].Dest)
      {
         if (new->Cost + 1 < rt[i].Cost)
            break;
         else if (new->NextHop == rt[i].NextHop)
            break;
         else
            return;
   } }
   rt[i] = *new;
   rt[i].TTL = MAX_TTL;
   ++rt[i].Cost;
   if (i == numRoutes)
      ++numRoutes;
}
```

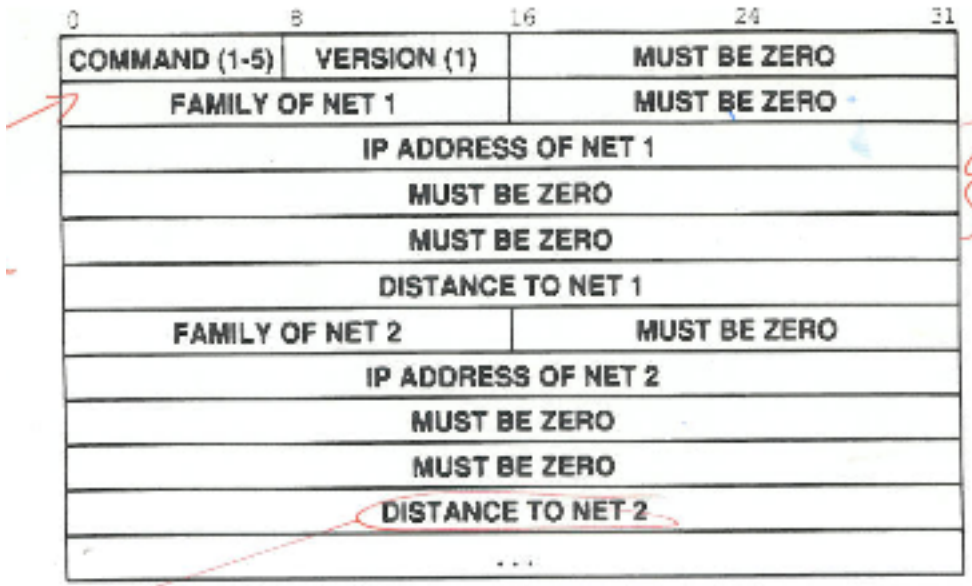


Figure 15.5 The format of a RIP message. After the 32-bit header, the message contains a sequence of pairs, where each pair consists of a network IP address and an integer distance to that network.

In the figure, field *COMMAND* specifies an operation according to the following table:

Command	Meaning
1	Request for partial or full routing information
2	Response containing network-distance pairs from sender's routing table
3	Turn on trace mode (obsolete)
4	Turn off trace mode (obsolete)
5	Reserved for Sun Microsystems internal use

Summary of LS and DV

- Distance Vector -- each node talks to its directly connected neighbors. It tells them everything that it knows about connectivity
- Link State - each node talks to all other nodes, but it tells them only what it knows for sure -- directly connected links. Nodes then build their own network graph.

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) messages sent
- DV: exchange between neighbors only
  - Convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) messages
- DV: convergence time varies
  - May be routing loops
  - Count-to-infinity problem

Robustness: what happens if router malfunctions?

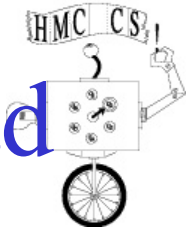
LS:

- Node can advertise incorrect *link* cost
- Each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- Each node's table used by others (error propagates)

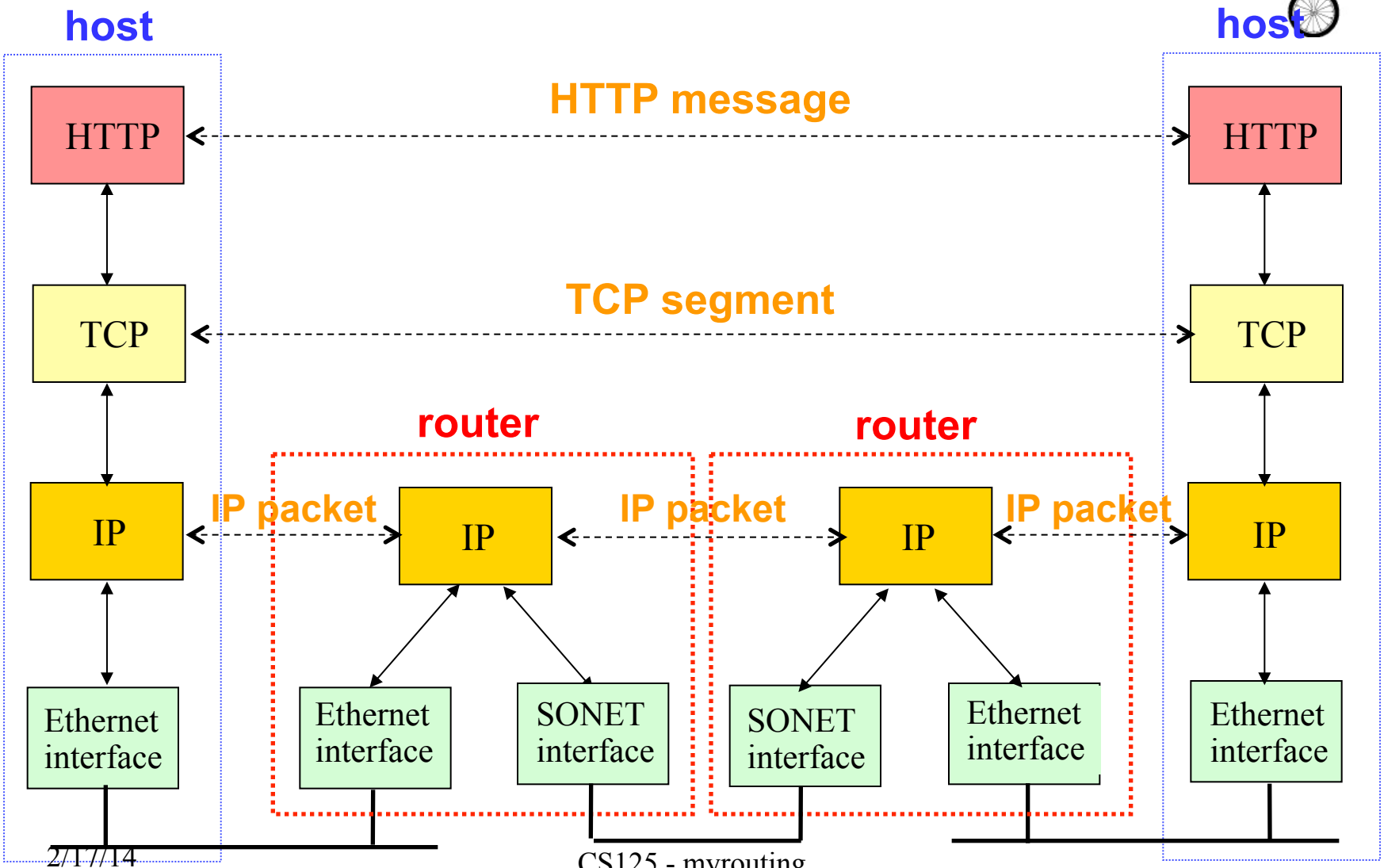
Metrics - how can we measure link cost..Latency, Capacity, Current Load



- Original ARPANET metric
  - measures number of packets queued on each link
  - took neither latency nor bandwidth into consideration
  - Move packets to shortest queue, not destination
- New ARPANET metric
  - stamp each incoming packet with its arrival time (**AT**)
  - record departure time (**DT**)
  - when link-level ACK arrives, compute

    Delay = (DT - AT) + Transmit + Latency
  - if timeout, reset **DT** to departure time for retransmission
  - link cost = average delay over some time period
  - Note: latency is statically defined for a link

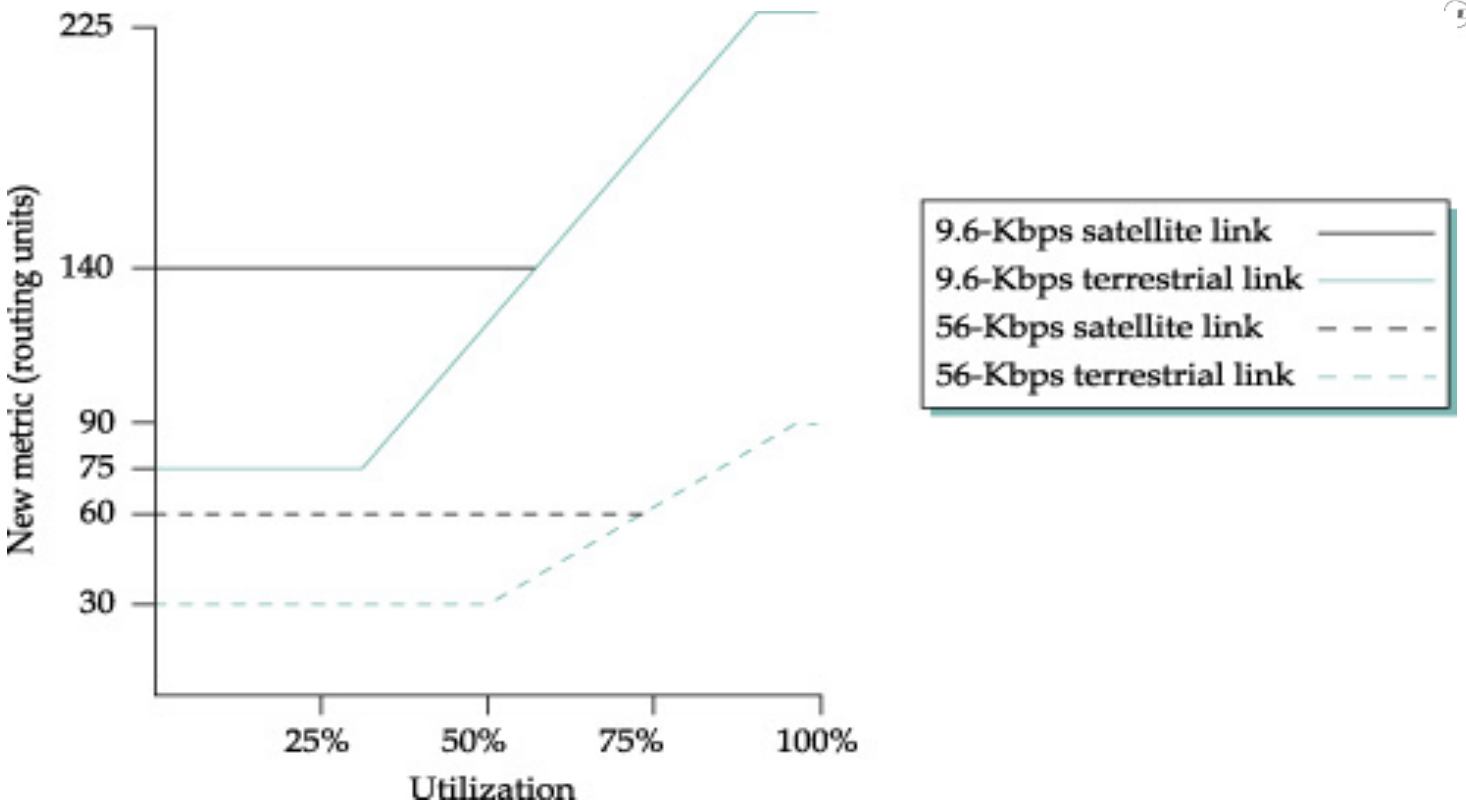
IP Suite In Action: End Hosts vs. Routers



Metrics -cont

- Problems with New metric
  - Under low load, static factors dominated cost
  - Under high load, congested links had very high costs; packets oscillated between congested and idle links
  - Range of costs too large: prefered path of 126 lightly loaded 56Kbps links to a 1 hop 9.6 Kbps path.
  - Fine Tuning
- Revised
  - Replace delay measurement with link utilization
  - Compressed dynamic range
  - Highly loaded link never > 3 times idle cost
  - Most expensive link only 7 times least
  - Cost is a function of link utilization only at moderate to high loads
  - Less likely to universally abandon a link.

Metrics -cont



Use of Metrics

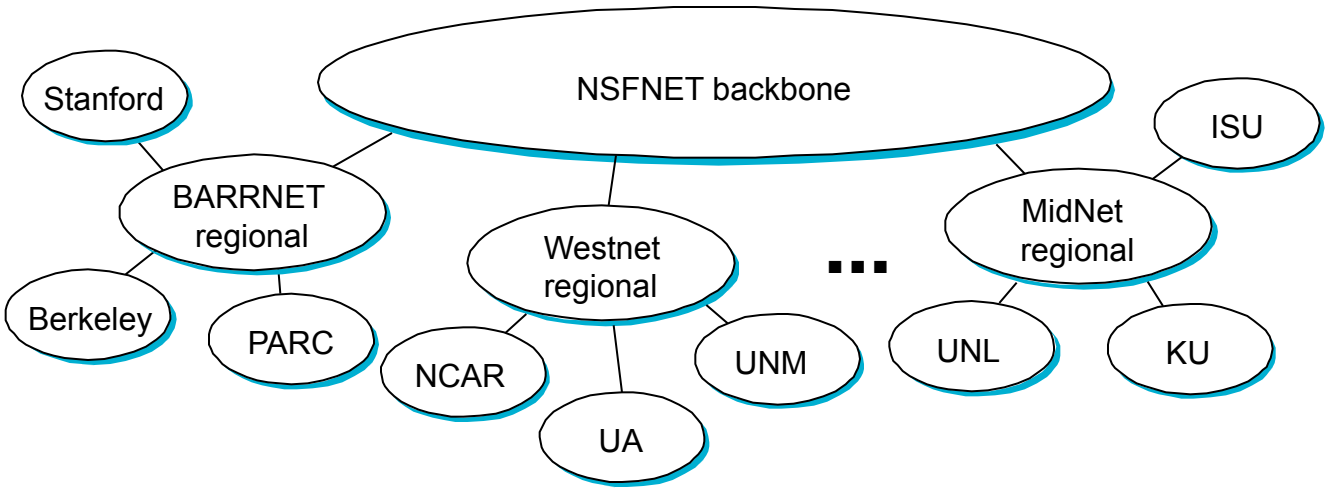
1. Not instantaneous, average over time
2. Updates only sent when > some threshold

How to Make Routing Scale

- Flat versus Hierarchical Addresses
- Inefficient use of Hierarchical Address Space
  - class C with 2 hosts (2/255 = 0.78% efficient)
  - class B with 256 hosts (256/65535 = 0.39% efficient)
- Still Too Many Networks
  - routing tables do not scale
  - route propagation protocols do not scale

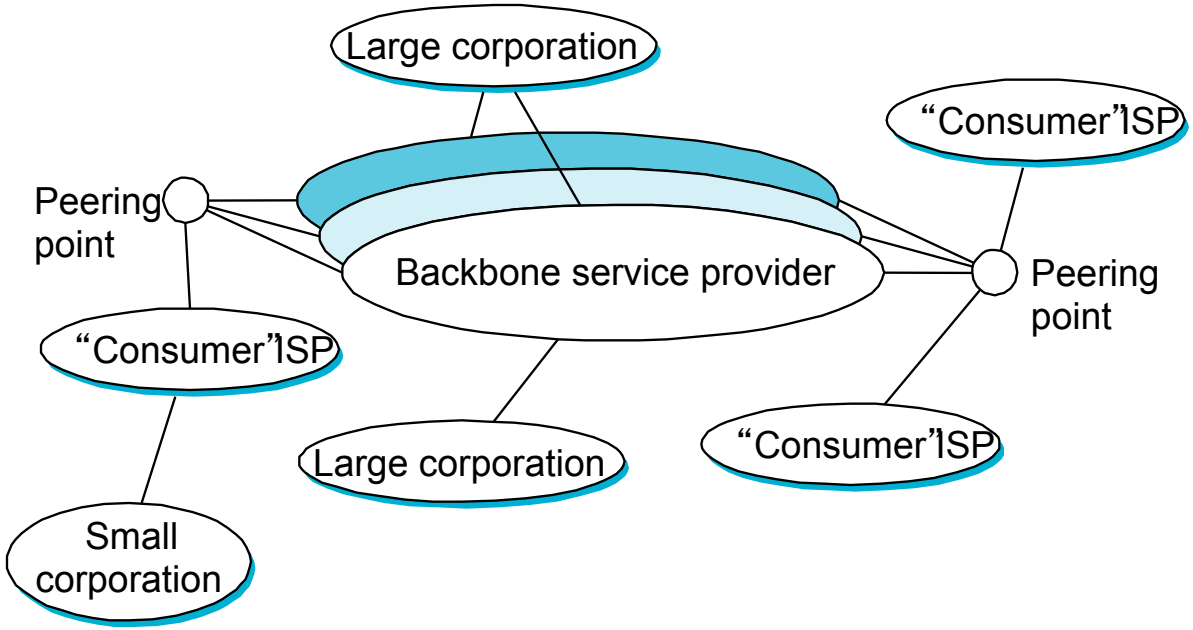
Internet Structure

Recent Past



Internet Structure

Today



Forwarding Algorithm


```
D = destination IP address
for each entry (SubnetNum, SubnetMask, NextHop)
    D1 = SubnetMask & D
    if D1 = SubnetNum
        if NextHop is an interface
            deliver datagram directly to D
        else
            deliver datagram to NextHop
```

- Use a default router if nothing matches
- Not necessary for all 1s in subnet mask to be contiguous: be liberal in what you accept, conservative in what you send
- Can put multiple subnets on one physical network
- Subnets not visible from the rest of the Internet

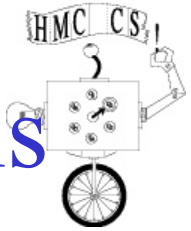
Route Propagation

- Know a smarter router
  - hosts know local router
  - local routers know site routers
  - site routers know core router
  - core routers know everything
- Autonomous System (AS)
  - corresponds to an administrative domain
  - examples: University, company, backbone network
  - assign each AS a 16-bit number
- Two-level route propagation hierarchy
  - interior gateway protocol (each AS selects its own)
  - exterior gateway protocol (Internet-wide standard)

Reduce the Route Problem

- Participating Nodes = # of ASs
- Interdomain Route = Path to AS Border Route
- Therefore Complexity of Interdomain routing == order of number of ASs
- Complexity of Intradomain routing = order of number of networks in an AS

Popular Interior Gateway Protocols

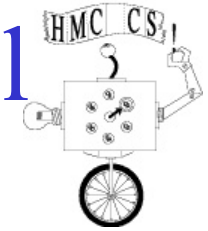


- RIP: Route Information Protocol
  - developed for XNS
  - distributed with Unix
  - distance-vector algorithm
  - based on hop-count
- OSPF: Open Shortest Path First
  - recent Internet standard
  - uses link-state algorithm
  - supports load balancing
  - supports authentication

EGP: Exterior Gateway Protocol Historical

- Overview
  - designed for tree-structured Internet
  - concerned with *reachability*, not optimal routes
- Protocol messages
  - neighbor acquisition: one router requests that another be its peer; peers exchange reachability information
  - neighbor reachability: one router periodically tests if the another is still reachable; exchange HELLO/ACK messages; uses a k-out-of-n rule
  - routing updates: peers periodically exchange their routing tables (distance-vector)

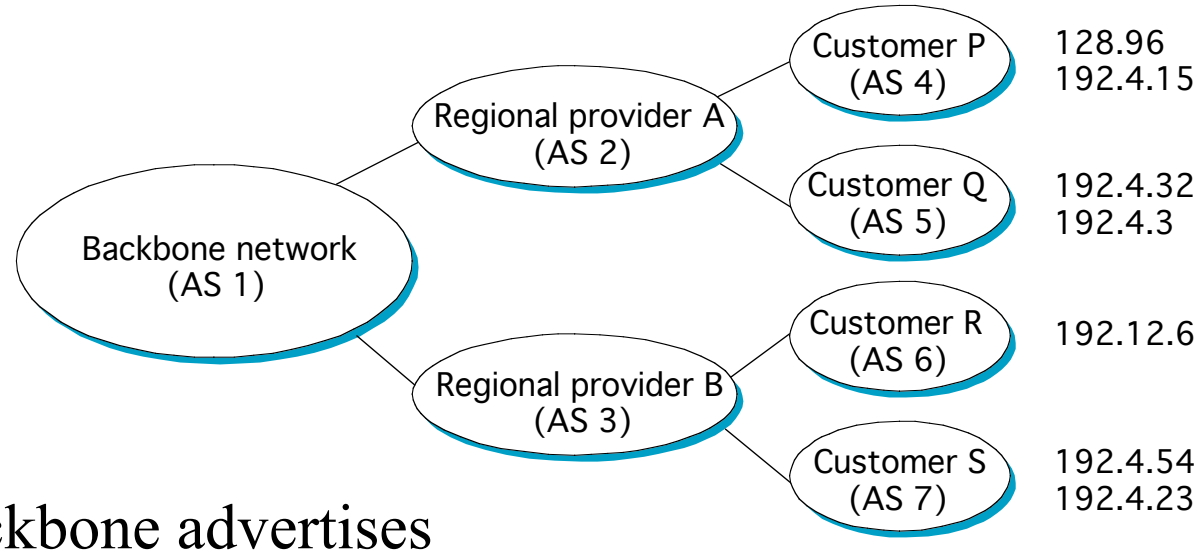
BGP-4: Border Gateway Protocol



- AS Types
  - stub AS: has a single connection to one other AS
    - carries local traffic only (HMC)
  - multihomed AS: has connections to more than one AS
    - refuses to carry transit traffic (Boeing)
  - transit AS: has connections to more than one AS
    - carries both transit and local traffic (ISP)
- Each AS has:
  - one or more border routers
  - one BGP *speaker* that advertises:
    - local networks
    - other reachable networks (transit AS only)
    - gives *path* information – list of ASs
    - Reachability over optimality
- Internet = interconnected set of ASs

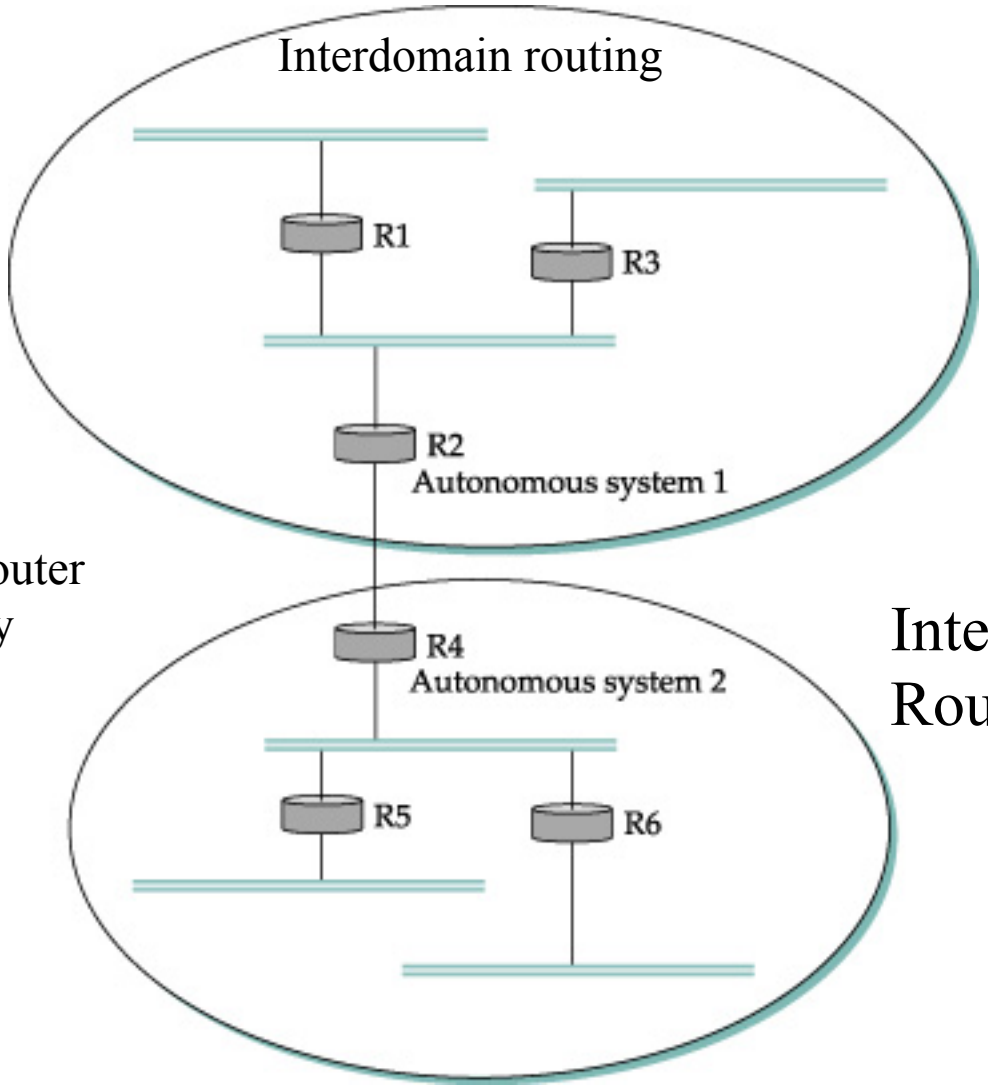
BGP Example

- Speaker for AS2 advertises reachability to P and Q
  - network 128.96, 192.4.153, 192.4.32, and 192.4.3, can be reached directly from AS2



- Speaker for backbone advertises
  - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2).
- Speaker can cancel previously advertised paths

AS = Routing Domain



R2, R4 = Border Router
EGP and IGP ability

Interdomain Routing

Conclusions

- Routing is a distributed algorithm
  - React to changes in the topology
  - Compute the shortest paths
- Two main shortest-path algorithms
  - Dijkstra → link-state routing (e.g., OSPF and IS-IS)
  - Bellman-Ford → distance vector routing (e.g., RIP)
- Convergence process
  - Changing from one topology to another
  - Transient periods of inconsistency across routers